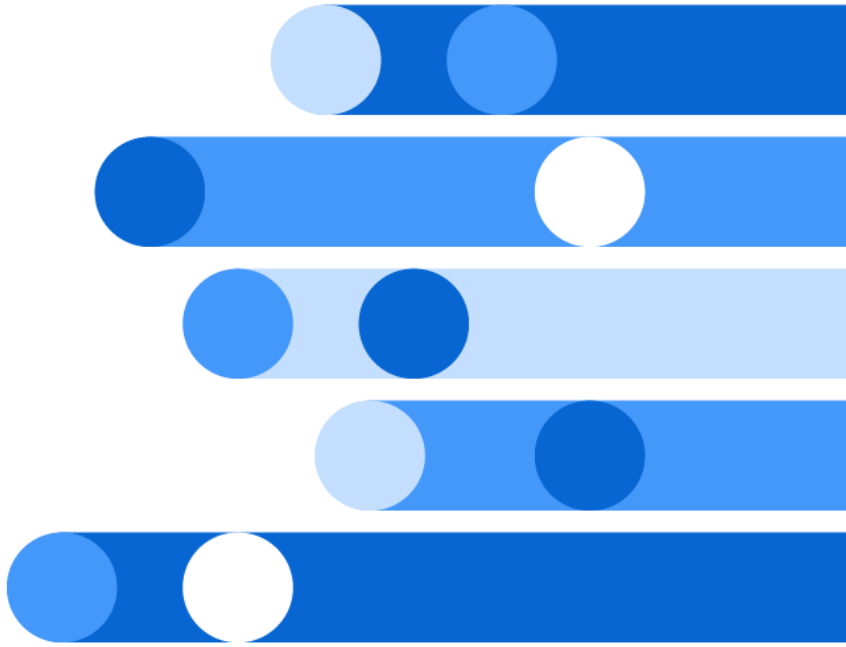
SAS® Visual Statistics Procedures

2026.07*



* This document might apply to additional versions of the software. Open this document in SAS Help Center and click on the version in the banner to see all available versions.



SAS® Documentation
July 16, 2026

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2026. *SAS® Visual Statistics: Procedures*. Cary, NC: SAS Institute Inc.

SAS® Visual Statistics: Procedures

Copyright © 2026, SAS Institute Inc., Cary, NC, USA

All Rights Reserved. Produced in the United States of America.

For a hard-copy book: No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

For a web download or e-book: Your use of this publication shall be governed by the terms established by the vendor at the time you acquire this publication.

The scanning, uploading, and distribution of this book via the internet or any other means without the permission of the publisher is illegal and punishable by law. Please purchase only authorized electronic editions and do not participate in or encourage electronic piracy of copyrighted materials. Your support of others' rights is appreciated.

July 2026

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.

SAS software may be provided with certain third-party software, including but not limited to open source software, which is licensed under its applicable third-party software license agreement. For license information about third-party software distributed with SAS software, refer to Third-Party Software Reference | SAS Support.

v_054-P1:casstat

Contents

Chapter 1. Introduction . . . 1
Chapter 2. Shared Concepts . . . 5
Chapter 3. ASSESS Procedure . . . 111
Chapter 4. BART Procedure . . . 131
Chapter 5. BINNING Procedure . . . 171
Chapter 6. CAEFFECT Procedure . . . 199
Chapter 7. CARDINALITY Procedure . . . 245
Chapter 8. CATTRANSFORM Procedure . . . 265
Chapter 9. CORRELATION Procedure . . . 287
Chapter 10. EFA Procedure . . . 311
Chapter 11. FREQTAB Procedure . . . 379
Chapter 12. GAMMOD Procedure . . . 579
Chapter 13. GAMSELECT Procedure . . . 641
Chapter 14. GENSELECT Procedure . . . 703
Chapter 15. ICA Procedure . . . 795
Chapter 16. KCLUS Procedure . . . 821
Chapter 17. LMIXED Procedure . . . 859
Chapter 18. LOGSELECT Procedure . . . 959
Chapter 19. MBC Procedure . . . 1091
Chapter 20. MODELMATRIX Procedure . . . 1129
Chapter 21. MVOUTLIER Procedure . . . 1151
Chapter 22. NLMOD Procedure . . . 1201
Chapter 23. NMF Procedure . . . 1233
Chapter 24. PARTITION Procedure . . . 1265
Chapter 25. PCA Procedure . . . 1289
Chapter 26. PHSELECT Procedure . . . 1331
Chapter 27. PLSMOD Procedure . . . 1377
Chapter 28. QTRSELECT Procedure . . . 1419
Chapter 29. REGSELECT Procedure . . . 1475
Chapter 30. SANDWICH Procedure . . . 1553
Chapter 31. SIMSYSTEM Procedure . . . 1577
Chapter 32. SPC Procedure . . . 1641
Chapter 33. SUPERLEARNER Procedure . . . 1725
Chapter 34. TREESPLIT Procedure . . . 1781
Chapter 35. VARIMPUTE Procedure . . . 1875
Chapter 36. VARREDUCE Procedure . . . 1887

Chapter 1
Introduction

Contents

Overview of SAS Visual Statistics Procedures **1**
About This Book **2**
Chapter Organization 2
Typographical Conventions 2
Options Used in Examples 3
Where to Turn for More Information **3**
Online Documentation 3
SAS Technical Support Services 4

Overview of SAS Visual Statistics Procedures

SAS Visual Statistics procedures include predictive modeling tools that have been specially developed to take advantage of the distributed environment that SAS Viya provides. Methods include linear regression, logistic regression, quantile regression, generalized linear models, generalized additive models, reduced-rank models, nonlinear models, clustering, and decision trees. The procedures provide model selection, dimension reduction, and identification of important variables whenever this identification is appropriate for the analysis.

This book also describes utility procedures, which can be used together with any SAS Viya procedures.

NOTE: When you license SAS Visual Statistics, you also have access to SAS/STAT software. For more information about SAS/STAT procedures, see *SAS/STAT User's Guide*.

About This Book

This book assumes that you are familiar with Base SAS software and with the books *SAS Programmers Guide: Essentials* and *Base SAS Procedures Guide*. It also assumes that you are familiar with basic SAS System concepts, such as using the DATA step to create SAS data sets and using Base SAS procedures (such as the PRINT and SORT procedures) to manipulate SAS data sets.

Chapter Organization

Chapter 1, this chapter, provides an overview of SAS Visual Statistics procedures and summarizes related information, products, and services.

Chapter 2 provides information about topics that are common to multiple procedures. Topics include how to use SAS Cloud Analytic Services (CAS) sessions and how to load a SAS data set onto a CAS server. It also documents the following statements that are used across a number of procedures: CLASS, CODE, EFFECT, PARTITION, and SELECTION.

The remaining chapters for statistical and utility procedures appear in alphabetical order by procedure name and are organized as follows:

- The “Overview” section briefly describes the analysis provided by the procedure.
- The “Getting Started” section provides a quick introduction to the procedure through a simple example.
- The “Syntax” section describes the SAS statements and options that control the procedure.
- The “Details” section discusses methodology and other topics, such as ODS tables.
- The “Examples” section contains examples that use the procedure.
- The “References” section contains references for the methodology.

Typographical Conventions

This book uses several type styles for presenting information. The following list explains the meaning of the typographical conventions used in this book:

roman	is the standard type style used for most text.
UPPERCASE ROMAN	is used for SAS statements, options, and other SAS language elements when they appear in text. However, you can enter these elements in your own SAS programs in lowercase, uppercase, or a mixture of the two.
UPPERCASE BOLD	is used in the “Syntax” sections’ initial lists of SAS statements and options.
oblique	is used in the syntax definitions and in text to represent arguments for which you supply a value.

VariableName	is used for the names of variables and data sets when they appear in text.
bold	is used for matrices and vectors.
italic	is used for terms that are defined in text, for emphasis, and for references to publications.
`monospace bold`	is used for example code. In most cases, this book uses lowercase type for SAS code.

Options Used in Examples

The HTMLBLUE style is used to create the graphs and the HTML tables that appear in the online documentation. The PEARLJ style is used to create the PDF tables that appear in the documentation. A style template controls stylistic elements such as colors, fonts, and presentation attributes. You can specify a style template for an HTML ODS destination as follows:

```
ods html style=HTMLBlue;
```

You can also specify a style template for a PDF ODS destination as follows:

```
ods pdf style=PearlJ;
```

Most of the PDF tables are produced by using the following SAS System option:

```
options papersize=(6.5in 9in);
```

If you run the examples, you might get slightly different output. This is a function of the SAS System options that are used and the precision that your computer uses for floating-point calculations.

Where to Turn for More Information

Online Documentation

You can access the documentation by going to `http://support.sas.com/documentation`.

SAS Technical Support Services

The SAS Technical Support staff is available to respond to problems and answer technical questions regarding the use of procedures in this book. Go to `http://support.sas.com/techsup` for more information.

Chapter 2
Shared Concepts

Contents

Introduction to Shared Concepts **6**
Using CAS Sessions and CAS Engine Librefs 7
Loading a SAS Data Set onto a CAS Server 7
Syntax Common to SAS Visual Statistics Procedures **9**
CLASS Statement 9
CODE Statement 13
DISPLAY Statement 16
DISPLAYOUT Statement 17
EFFECT Statement 18
Collection Effects 20
Multimember Effects 21
Polynomial Effects 22
Spline Effects 25
Splines and Spline Bases 28
ODS Table Names 33
LSMEANS Statement 34
PARTITION Statement 41
SELECTION Statement 42
STORE Statement 56
Optimization Options 57
Details for SAS Visual Statistics Procedures **62**
Levelization of Classification Variables 62
Specification and Parameterization of Model Effects 64
Effect Operators 65
GLM Parameterization of Classification Variables and Effects 67
Nonsingular Parameterization 71
Class Variable Parameterization with Unbalanced Designs 74
Model Selection Methods 75
Full Model Fitted 75
Forward Selection 75
Backward Elimination 77
Stepwise Selection 78
Forward-Swap Selection 80
Least Angle Regression 81
LASSO Selection 81

Adaptive LASSO Selection . . . 82
Relaxed LASSO Selection . . . 83
Elastic Net Selection . . . 83
Adaptive Elastic Net Selection . . . 86
Group LASSO Selection . . . 87
Best-Subset Selection . . . 88
Folded Concave Penalized Selection . . . 89
Model Selection Plots . . . 93
Coefficient Panel . . . 94
Criterion Panel . . . 95
Fit by Role Plot . . . 95
Using Options to Customize Plots . . . 96
Examining Specific Step Ranges . . . 98
Informative Missingness . . . 99
Using Validation and Test Data . . . 101
Using the Validation Statistic as the CHOOSE= Criterion . . . 102
Using the Validation Statistic as the STOP= Criterion . . . 102
Multithreading . . . 102
Choosing an Optimization Algorithm . . . 103
First- or Second-Order Algorithms . . . 103
Algorithm Descriptions . . . 104
The APPLYROWORDER Option . . . 107
References . . . **108**

Introduction to Shared Concepts

SAS Visual Statistics procedures run on SAS Viya. One component of SAS Viya is SAS Cloud Analytic Services (CAS), which is the analytic server and associated cloud services. The following subsections describe how to set up and use CAS sessions.

The section "Syntax Common to SAS Visual Statistics Procedures" on page 9 describes the common syntax elements that are supported by some of the procedures in this book. In some cases, individual procedures implement these common elements in slightly different ways. When this occurs, the differences are described in the respective procedure chapters.

The section "Details for SAS Visual Statistics Procedures" on page 62 provides details that are common to some of the procedures in this book.

Using CAS Sessions and CAS Engine Librefs

SAS Cloud Analytic Services (CAS) is the analytic server and associated cloud services in SAS Viya. This section describes how to create a CAS session and set up a CAS engine libref that you can use to connect to the CAS session. It assumes that you have a CAS server already available; contact your system administrator if you need help starting and terminating a server. This CAS server is identified by specifying the host on which it runs and the port on which it listens for communications. To simplify your interactions with this CAS server, the host information and port information for the server are stored as SAS option values that are retrieved automatically whenever this CAS server needs to be accessed. You can examine the host and port values for the server at your site by using the following statements:

```
proc options option=(CASHOST CASPORT);
run;
```

In addition to starting a CAS server, your system administrator might also have created a CAS session and a CAS engine libref for your use. You can define your own sessions and CAS engine librefs that connect to the CAS server as shown in the following statements:

```
cas mysess;
libname mylib cas sessref=mysess;
```

The CAS statement creates the CAS session named mysess, and the LIBNAME statement creates the mylib CAS engine libref that you use to connect to this session. It is not necessary to explicitly name the CASHOST and CASPORT of the CAS server in the CAS statement, because these values are retrieved from the corresponding SAS option values.

If you have created the mysess session, you can terminate it by using the TERMINATE option in the CAS statement as follows:

```
cas mysess terminate;
```

For more information about the CAS statement and the LIBNAME statement, see *SAS Cloud Analytic Services: User's Guide*. For general information about CAS and CAS sessions, see *SAS Cloud Analytic Services: Fundamentals*.

Loading a SAS Data Set onto a CAS Server

Procedures in this book require the input data to reside on a CAS server. To work with a SAS data set, you must first load the data set onto the CAS server. Data loaded on the CAS server are called *data tables*. This section lists three methods of loading a SAS data set onto a CAS server. In this section, mylib is the name of the caslib that is connected to the mysess CAS session.

- You can use a single DATA step to create a data table on the CAS server as follows:

```
data mylib.Sample;
   input y x @@;
   datalines;
.46 1 .47 2 .57 3 .61 4 .62 5 .68 6 .69 7
;
```

Note that DATA step operations might not work as intended when you perform them on the CAS server instead of the SAS client.

- You can create a SAS data set first, and when it contains exactly what you want, you can use another DATA step to load it onto the CAS server as follows:

```
data Sample;
   input y x @@;
   datalines;
.46 1 .47 2 .57 3 .61 4 .62 5 .68 6 .69 7 .78 8
;
data mylib.Sample;
   set Sample;
run;
```

- You can use the CASUTIL procedure as follows:

```
proc casutil sessref=mysess;
   load data=Sample casout="Sample";
quit;
```

The CASUTIL procedure can load data onto a CAS server more efficiently than the DATA step. For more information about the CASUTIL procedure, see *SAS Cloud Analytic Services: User's Guide*.

The mylib caslib stores the Sample data table, which can be distributed across many machine nodes. You must use a caslib reference in procedures in this book to enable the SAS client machine to communicate with the CAS session. For example, the following REGSELECT procedure statements use a data table that resides in the mylib caslib:

```
proc regselect data = mylib.Sample;
  ...statements...;
run;
```

You can delete your data table by using the DELETE procedure as follows:

```
proc delete data = mylib.Sample;
run;
```

The Sample data table is accessible only in the mysess session. When you terminate the mysess session, the Sample data table is no longer accessible from the CAS server. If you want your Sample data table to be available to other CAS sessions, then you must *promote* your data table. For more information about data tables, see *SAS Cloud Analytic Services: User's Guide*.

Syntax Common to SAS Visual Statistics Procedures

CLASS Statement

CLASS *variable* <(*options*)>. . . <*variable* <(*options*)>> </ *global-options*> **;**

This section applies to the following procedures: GAMMOD, GAMSELECT, GENSELECT, LMIXED, LOGSELECT, MODELMATRIX, PHSELECT, PLSMOD, QTRSELECT, REGSELECT, SANDWICH, and TREESPLIT.

The CLASS statement names the classification variables to be used as explanatory variables in the analysis. These variables enter the analysis not through their values, but through levels to which the unique values are mapped. For more information about these mappings, see the section "Levelization of Classification Variables" on page 62.

If the procedure permits a classification variable as a response (dependent variable or target), the response does not need to be specified in the CLASS statement.

You can specify options either as individual variable *options*, by enclosing the options in parentheses after the variable name, or as *global-options*, by placing them after a slash (/). *Global-options* are applied to all variables that are specified in the CLASS statement. If you specify more than one CLASS statement, the *global-options* that are specified in any one CLASS statement apply to all CLASS statements. However, individual CLASS variable *options* override the *global-options*.

Table 2.1 summarizes the values you can use for either an *option* or a *global-option*. The options are described in detail in the list that follows Table 2.1.

Table 2.1 CLASS Statement Options

Option	Description
DESCENDING	Reverses the sort order
MISSING	Treats missing values as valid levels
ORDER=	Specifies the sort order for the levels
PARAM=	Specifies the parameterization of the variable
REF=	Specifies the reference level of the variable
SPLIT	Splits levels of CLASS variables into independent effects

DESCENDING

DESC

reverses the sort order of the classification variable. If both the DESCENDING and ORDER= options are specified, the procedure orders the categories according to the ORDER= option and then reverse that order.

MISSING

treats missing values (“.”, “.A”, . . . , “.Z” for numeric variables and blanks for character variables) as valid values for the CLASS variable.

If you do not specify the MISSING option, observations that have missing values for CLASS variables are removed from the analysis.

ORDER=< FORMATTED | FREQ | INTERNAL > < (*var1*='*level1*' '*level2*' . . . *var2*= . . .) >

ORDER= FORMATTED | FREQ | INTERNAL | '*level1*' '*level2*' . . .

specifies the sort order for the levels of classification variables. This order determines which parameters in the model correspond to each level in the data. The following table shows how to interpret values of the ORDER= keywords:

Value of ORDER=	Levels Sorted By
FORMATTED	External formatted values, except for numeric variables that have no explicit format, which are sorted by their unformatted (internal) values. The sort order is machine-dependent. For numeric variables for which you have supplied no explicit format, the levels are ordered by their internal values.
FREQ	Descending frequency count (levels that have more observations come earlier in the order)
INTERNAL	Unformatted value. The sort order is machine-dependent.

PROC LOGSELECT also enables you to specify a particular order for variables by listing the levels in quotes in the desired order. For example, in the following statement, the global specification (after the slash) specifies that the levels of all variables except the pet variable are to be sorted by their formatted value, and the individual specification of the food variable overrides the FORMATTED order:

```
class pet food(order='wet' 'dry') / order=formatted(pet='dog' 'cat');
```

Any level of the pet and food variables that you do not specify is added to the end of the variable list in formatted order.

The ORDER= option does not apply to missing values, which are always ordered first.

For more information about sort order, see the chapter about the SORT procedure in *Base SAS Procedures Guide* and the discussion of BY-group processing in the “Grouping Data” section of *SAS Programmers Guide: Essentials*. By default, ORDER=FORMATTED.

PARAM=*keyword*

specifies the parameterization method for the classification variable or variables. You can specify any of the *keywords* shown in the following table; design matrix columns are created from CLASS variables according to the corresponding coding schemes.

Table 2.2 Value of PARAM=

Value of PARAM=	Coding
EFFECT	Effect coding. The REF= option in the CLASS statement determines the reference level.
GLM	Less-than-full-rank reference cell coding. This *keyword* can be used only as a *global-option* and is applied to all CLASS variables; all other individual variable parameterization specifications are ignored. The REF= option in the CLASS statement indirectly determines the reference level through the order of levels.
ORDINAL | THERMOMETER	Cumulative parameterization for an ordinal CLASS variable
POLYNOMIAL | POLY	Polynomial coding. If the classification variable is numeric, then the ORDER= option in the CLASS statement is ignored, and the internal unformatted values are used.
REFERENCE | REF	Reference cell coding. The REF= option in the CLASS statement determines the reference level.
ORTHEFFECT	Orthogonalizes PARAM=EFFECT coding. The REF= option in the CLASS statement determines the reference level.
ORTHORDINAL | ORTHOTHERM	Orthogonalizes PARAM=ORDINAL coding
ORTHPOLY	Orthogonalizes PARAM=POLYNOMIAL coding. If the classification variable is numeric, then the ORDER= option in the CLASS statement is ignored, and the internal unformatted values are used.
ORTHREF	Orthogonalizes PARAM=REFERENCE coding. The REF= option in the CLASS statement determines the reference level.

All parameterizations are full rank, except for the GLM parameterization. If you specify a full rank parameterization for any CLASS variable, then every CLASS variable without a specified coding is given the EFFECT coding.

By default, PARAM=GLM. For more information about how parameterization of classification variables affects the construction and interpretation of model effects, see the section "Specification and Parameterization of Model Effects" on page 64.

REF='*level***' | *keyword***

REFERENCE='*level***' | *keyword***

specifies the reference level that is used when you specify a nonsingular parameterization. You can specify the following values:

'*level*' specifies the *level* of the variable to use as the reference level. Specify the formatted value of the variable if a format is assigned. You can specify this value only for an individual variable *option*.

FIRST designates the first ordered level as reference. You can specify this value either for an individual variable *option* or for a *global-option*.

LAST designates the last ordered level as reference. You can specify this value either for an individual variable *option* or for a *global-option*.

By default, REF=LAST.

SPLIT

specifies that design matrix columns that correspond to any effect that contains a split classification variable can be selected to enter or leave a model independently of the other design columns of that effect. This option applies to procedures that perform model selection.

Suppose that the variable temp has three levels (`'hot'`, `'warm'`, and `'cold'`), that the variable gender has two levels (`'M'` and `'F'`), and that the variables are used in a PROC REGSELECT run as follows:

```
proc regselect data=mylib.data;
   class temp gender / split;
   model y = gender gender*temp;
run;
```

The two effects in the MODEL statement are split into eight independent effects. The effect "gender" is split into two effects that are labeled "gender_M" and "gender_F". The effect "gender*temp" is split into six effects that are labeled "gender_M*temp_hot", "gender_F*temp_hot", "gender_M*temp_warm", "gender_F*temp_warm", "gender_M*temp_cold", and "gender_F*temp_cold". The previous PROC REGSELECT step is equivalent to the following:

```
proc regselect data=mylib.data;
   model y = gender_M gender_F
             gender_M*temp_hot  gender_F*temp_hot
             gender_M*temp_warm gender_F*temp_warm
             gender_M*temp_cold gender_F*temp_cold;
run;
```

The SPLIT option can be used on individual classification variables. For example, consider the following PROC REGSELECT step:

```
proc regselect data=mylib.data;
   class temp(split) gender;
   model y = gender gender*temp;
run;
```

In this case, the effect "gender" is not split and the effect "gender*temp" is split into three effects, which are labeled "gender*temp_hot", "gender*temp_warm", and "gender*temp_cold". Furthermore, each of these three split effects now has two parameters that correspond to the two levels of "gender." The PROC REGSELECT step is equivalent to the following:

```
proc regselect data=mylib.data;
   class gender;
   model y = gender gender*temp_hot gender*temp_warm gender*temp_cold;
run;
```

CODE Statement

CODE < *options* > **;**

This section applies to the following procedures: GENSELECT, KCLUS, LOGSELECT, PCA, PHSELECT, QTRSELECT, REGSELECT, TREESPLIT, and VARIMPUTE.

The CODE statement writes SAS DATA step code for computing predicted values of the fitted model to a file or to a table. For survival procedures, the predicted values include survival probabilities and cumulative hazards at specific time points. To score new data, you can then include the file in a DATA step, or you can specify the table in the **runCodeTable** action in the **dataStep** action set.

Table 2.3 summarizes the *options* that you can specify in the CODE statement.

Table 2.3 CODE Statement Options

Option	Description
COMMENT	Adds comments to the generated code
CUMHAZ	Generates cumulative hazard values at specified time points
FILE=	Names the file in which to save the generated code
FORMATWIDTH=	Specifies the numeric format width for the regression coefficients
INDENTSIZE=	Specifies the number of spaces to indent the generated code
IPROB	Computes individual predicted probabilities for ordinal response models
LABELID=	Specifies a number used to construct names and labels
LINESIZE=	Specifies the line size for the generated code
NOSURVIVAL	Suppresses the generation of survival probabilities
NOTRIM	Compares formatted values, including blank padding
OUT=	Names an output table in which to save the generated code
PCATALL	Generates probabilities for all levels of categorical response variables
SHOWTIME	Creates variables that contain the time points at which predictions are made
TIMEPOINT=	Specifies the time points at which survival probabilities or cumulative hazards are predicted

If you do not specify the FILE= option or the OUT= option and if your SAS client has a default path, then the SAS scoring code is written to an external file named *_code_*. You can specify the following *options* in the CODE statement.

COMMENT
adds comments to the generated code.

CUMHAZ
generates SAS code to predict the cumulative hazard function at the time points that you specify in the TIMEPOINT= option. This option applies to the PHSELECT procedure.

FILE=*filename*
names the external file that saves the generated code. When enclosed in a quoted string (for example, FILE="c:\mydir\scorecode.sas"), this option specifies the path and filename for writing the code to an external file. If you do not specify a path but your SAS client has a default path, then the code is written to an external file named *filename* at that location. You can also specify an unquoted *filename* of no more than eight characters. If the *filename* is assigned as a fileref in a Base SAS FILENAME statement, the file that is specified in the FILENAME statement is opened; otherwise, if your SAS client has a default path, an external file named *filename* is created. This option cannot be specified together with the OUT= option.

FORMATWIDTH=*width*
specifies the width to use in formatting derived numbers such as parameter estimates. You can specify a value in the range 12 to 32; the default is 20.

INDENTSIZE=*n*
specifies the number of spaces to indent the generated code. You can specify a value in the range 0 to 10; the default is 3.

IPROB
computes the individual predicted probabilities for ordinal response models. This option applies to the following procedures: GENSELECT and LOGSELECT. For a response variable Y with three levels, 1, 2, and 3, the individual probabilities are $\Pr(Y = 1)$, $\Pr(Y = 2)$, and $\Pr(Y = 3)$.

LABELID=*value*
specifies a number used to construct array names and statement labels in the generated code. You can specify a value in the range 0 to 1024; the default is randomly chosen.

LINESIZE=*value*

LS=*value*
specifies the line size for the generated code. You can specify a value in the range 64 to 254; the default is 120.

NOSURVIVAL
suppresses SAS code for survival probabilities prediction. This option applies to the PHSELECT procedure. If you do not specify this option, survival procedures generate SAS code to predict survival probabilities at the time points that you specify in the TIMEPOINT= option.

NOTRIM
bases comparisons of formatted values on the full format width, including blank padding. By default, blanks at the beginning and end of strings are ignored.

OUT=*libref.data-table*

creates a table that contains the score code. *libref.data-table* is a two-level name, where *libref* refers to the library, and *data-table* specifies the name of the output data table. For more information about this two-level name, see the section "Using CAS Sessions and CAS Engine Librefs" on page 7. You can specify this table in the **runCodeTable** action in the **dataStep** action set to score another data table; for more information, see *SAS Viya: System Programming Guide*. This option cannot be specified with the FILE= option.

PCATALL

generates probabilities for all levels of categorical response variables. This option applies to the following procedures: GENSELECT and LOGSELECT.

SHOWTIME

creates variables that contain the time points at which predictions are made. This option applies to the PHSELECT procedure. If T is the name of the failure time variable, these time point variables are named T_1, T_2, and so on.

TIMEPOINT=*list* | **QUANTILE(***probability-list***)**

TIME=*list* | **QUANTILE(***probability-list***)**

specifies the time points at which survival probabilities or cumulative hazards are predicted. This option applies to the PHSELECT procedure. You can specify a list of numbers that represent exact time points. For example:

```
code cumhaz timepoint = 40 to 60 by 10;
```

If T is the name of the failure time variable, the preceding specification requests the predicted cumulative hazard and survival probability at the time points T=40, T=50, and T=60. If you also specify the SHOWTIME option, these time points are saved in the variables T_1, T_2, and T_3, respectively. The predicted variables are named and labeled as shown in Table 2.4.

Table 2.4 Predicted Variables

T	Name	Label
40	C_T_1	Cumulative Hazard at T_1
50	C_T_2	Cumulative Hazard at T_2
60	C_T_3	Cumulative Hazard at T_3
40	S_T_1	Survival Probability at T_1
50	S_T_2	Survival Probability at T_2
60	S_T_3	Survival Probability at T_3

Alternatively, you can use the keyword QUANTILE to specify a list of quantile probabilities. For example:

```
code timepoint = quantile(.2, .5, .8);
```

Survival procedures compute the 20th, 50th, and 80th percentiles from the Kaplan-Meier curve (or the Breslow curve if you also specify the ENTRY= option in the MODEL statement) and use them as the time points for the prediction.

By default, TIMEPOINT=QUANTILE(0.25, 0.50, 0.75).

DISPLAY Statement

DISPLAY < *table-list* > < / *options* > **;**

This section applies to the following procedures: CORRELATION, GAMMOD, GAMSELECT, GENSELECT, ICA, KCLUS, LMIXED, LOGSELECT, MBC, NLMOD, NMF, PARTITION, PCA, PHSELECT, PLSMOD, QTRSELECT, REGSELECT, SANDWICH, and VARREDUCE.

The DISPLAY statement enables you to specify a list of display tables to display or exclude. This statement is similar to the ODS SELECT, ODS EXCLUDE, and ODS TRACE statements. However, the DISPLAY statement can improve performance when a large number of tables could be generated (such as in BY-group processing). The procedure processes the DISPLAY statement on a CAS server and thus sends only a subset of ODS tables to the SAS client. Because ODS statements are processed on a SAS client, first all the generated display tables are sent to the client, and then the client creates a subset.

If you use both DISPLAY and ODS statements together, the DISPLAY statement takes precedence over the ODS statements. Note that the ODS EXCLUDE statement processes tables that are sent to the client after they have been filtered by the DISPLAY statement. In some cases, it might appear that the ODS EXCLUDE statement is taking precedence because it can further filter the tables. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

You can specify the *table-list* as a list of table names, paths, partial pathnames, and regular expressions.

The table names that you can specify are listed in the specific procedure chapters. A path is a table name that is prefixed with dot-separated grouping information. For example, a SelectionSummary table that a procedure produces during a selection routine might have the path *Bygroup1.Summary.SelectionSummary*. A partial pathname does not include all groups; for example, *SelectionSummary* and *Summary.SelectionSummary* are partial pathnames for *Bygroup1.Summary.SelectionSummary*.

When you specify a table name or partial pathname, all display tables whose paths end in the specified name are selected for display or exclusion. For example, both *SelectionSummary* and *Summary.SelectionSummary* select *Bygroup1.Summary.SelectionSummary*.

A regular expression is enclosed in forward slashes (/). For example, specifying "/tions/" selects all pathnames that contain the substring "tions"; in particular, the *Bygroup1.Summary.SelectionSummary* table is selected. Specifying "!/tions/" selects all pathnames that do not contain the substring "tions"; in particular, the *Bygroup1.Summary.SelectionSummary* table is not selected.

You can specify the following *options* after a slash (/):

CASESENSITIVE
performs a case-sensitive comparison of table names in the *table-list* to display table names when tables are subsetted for display. To preserve case, you must enclose table names in the *table-list* in quotation marks.

EXCLUDE
displays all display tables except those that you specify in the *table-list*.

EXCLUDEALL
suppresses display of all tables. This option takes precedence over the other options.

TRACE
displays the display table names, labels, and paths.

DISPLAYOUT Statement

DISPLAYOUT *table-spec-list* < / *options* > **;**

This section applies to the following procedures: CORRELATION, GAMMOD, GAMSELECT, GENSELECT, ICA, KCLUS, LMIXED, LOGSELECT, MBC, NLMOD, NMF, PARTITION, PCA, PHSELECT, PLSMOD, QTRSELECT, REGSELECT, SANDWICH, and VARREDUCE.

The DISPLAYOUT statement enables you to create output tables from your displayed output. This statement is similar to the ODS OUTPUT statement. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

The *table-spec-list* specifies a list of output tables to create. Each entry in the list has either a *key*=*value* format or a *key* format:

key=*value*	specifies *key* as the ODS table name, path, or partial pathname, and specifies *value* as the output table name.
key	specifies *key* as the ODS table name and also as the output table name.

The table names that you can specify are listed in the specific procedure chapters.

Table names and partial pathnames are discussed under the DISPLAY statement. The DISPLAYOUT statement does not support regular expressions.

You can specify the following *options* after a slash (/):

INCLUDEALL
creates output tables for all display tables. The name of the created output table is the same as the name of the corresponding display table. If you specify this option, the *table-spec-list* specification is ignored.

NOREPLACE
does not replace any existing output table of the same name.

REPEATED
replicates all output tables on all nodes.

EFFECT Statement

EFFECT *effect-name* = *effect-type* **(***var-list* < / *effect-options*>**) ;**

This section applies to the following procedures: GENSELECT, LMIXED, LOGSELECT, MODELMATRIX, PHSELECT, PLSMOD, QTRSELECT, and REGSELECT.

The EFFECT statement enables you to construct special collections of columns for design matrices. These collections are called *constructed effects* to distinguish them from the usual model effects that are formed from continuous or classification variables, as discussed in the section "GLM Parameterization of Classification Variables and Effects" on page 67. For example, the terms A, B, x, A*x, and A*B in the following statements define fixed effects of the usual type in a generalized linear model:

```
proc genselect;
   class A B;
   model y = A B x A*x;
run;
```

A constructed effect, on the other hand, is assigned through the EFFECT statement. For example, in the following program, the EFFECT statement defines a constructed effect named spl:

```
proc genselect;
   class A B;
   effect spl = spline(x);
   model y = A B A*spl;
run;
```

The columns of spl are formed from the data set variable x as a cubic B-spline basis with three equally spaced interior knots.

Each constructed effect corresponds to a collection of columns that are referred to by the name you supply. You can specify multiple EFFECT statements, and all EFFECT statements must precede the MODEL statement.

You must specify the following arguments:

effect-name
names the effect. This name can appear in only one EFFECT statement and cannot be the name of a variable in the input data set.

effect-type
specifies the type of effect. You can specify the following *effect-types*:

COLLECTION specifies a collection effect that defines one or more variables as a single effect that has multiple degrees of freedom. The variables in a collection are considered as a unit for purposes of estimation and inference. For more information, see the section "Collection Effects" on page 20.

MULTIMEMBER | MM specifies a multimember classification effect whose levels are determined by one or more variables that appear in a CLASS statement. For more information, see the section "Multimember Effects" on page 21.

POLYNOMIAL | POLY specifies a multivariate polynomial effect in the specified numeric variables. For more information, see the section "Polynomial Effects" on page 22.

SPLINE specifies a regression spline effect whose columns are univariate spline expansions of one or more variables. A spline expansion replaces the original variable with an expanded or larger set of new variables. For more information, see the section "Spline Effects" on page 25.

var-list
specifies a list of variables that are used in constructing the effect. You can also specify any of the *effect-options* that are shown in Table 2.5 after a slash following the *var-list*.

Table 2.5 EFFECT Statement Options

Option	Description
Collection Effects Options	
DETAILS	Displays the constituents of the collection effect
Multimember Effects Options	
DETAILS	Displays the levels of the multimember effect
NOEFFECT	Specifies that observations whose levels are all missing for the multimember variables should have 0 values in the corresponding design matrix columns
STDIZE	Standardizes the design matrix entries so that each observation has a sum of 1
WEIGHT=	Specifies the weight variable for the contributions of each classification effect

Table 2.5 *continued*

Option	Description
Polynomial Effects Options	
DEGREE=	Specifies the degree of the polynomial
DETAILS	Displays details of the specified polynomial
MDEGREE=	Specifies the maximum degree of any variable in a term of the polynomial
NOSEPARATE	Treats the polynomial as a single effect with multiple degrees of freedom
STANDARDIZE=	Specifies centering and scaling suboptions for the variables that define the polynomial
Spline Effects Options	
BASIS=	Specifies the type of basis (B-spline basis or truncated power function basis) for the spline effect
DATABOUNDARY	Uses the extremes of the data as boundary knots for a B-spline basis
DEGREE=	Specifies the degree of the spline effect
DETAILS	Displays the knots and locations for each spline basis function
KNOTMAX=	Requests equally spaced right-side boundary knots starting at the variables' maximum and ending at the KNOTMAX= value
KNOTMETHOD=	Specifies how to construct the knots for the spline effect
KNOTMIN=	Requests equally spaced left-side boundary knots starting at the KNOTMIN= value and ending at the variables' minimum value
NATURALCUBIC	Specifies a natural cubic spline basis for the spline effect
SEPARATE	Treats the spline basis for each variable as a separate effect when multiple variables are specified
SPLIT	Treats each design matrix column as a separate effect for selection methods

Collection Effects

EFFECT *effect-name*=**COLLECTION (***var-list* < / **DETAILS** >**) ;**

You use a collection effect to define a set of variables that are treated as a single effect that has multiple degrees of freedom. The variables in *var-list* can be continuous or classification variables. Columns in the design matrix that are contributed by a collection effect are the design columns of its constituent variables in the order in which they appear in the definition of the collection effect. If you specify the DETAILS option, then a table that shows the constituents of the collection effect is displayed.

Multimember Effects

EFFECT *effect-name*=**MULTIMEMBER (***var-list* < / *mm-options*>**) ;**

EFFECT *effect-name*=**MM (***var-list* < / *mm-options*>**) ;**

A multimember effect is formed from one or more classification variables in such a way that each observation can be associated with one or more levels of the union of the levels of the classification variables. In other words, a multimember effect is a classification-type effect with possibly more than one nonzero column entry for each observation. Multimember effects are useful, for example, in modeling the following:

- nurses' effects on patient recovery in hospitals
- teachers' effects on student scores
- lineage effects in genetic studies

The levels of a multimember effect consist of the union of formatted values of the variables that define this effect. Each such level contributes one column to the design matrix. For each observation, the value that corresponds to each level of the multimember effect in the design matrix is the number of times that this level occurs for the observation.

For example, the following data provide teacher information and end-of-year test scores for students after two semesters:

Student	Score	Teacher1	Teacher2
Mary	87	Tobias	Cohen
Tom	89	Rodriguez	Tobias
Fred	82	Cohen	Cohen
Jane	88	Tobias	.
Jack	99	.	.

For example, Mary had different teachers in the two semesters, Fred had the same teacher in both semesters, and Jane received instruction only in the first semester.

You can model the effect of the teachers on student performance by specifying a multimember effect as follows:

```
CLASS teacher1 teacher2;
EFFECT teacher = MM(teacher1 teacher2);
```

The levels of the teacher effect are Cohen, Rodriguez, and Tobias, and the associated design matrix columns are as follows:

Student	Cohen	Rodriguez	Tobias
Mary	1	0	1
Tom	0	1	1
Fred	2	0	0
Jane	0	0	1
Jack	.	.	.

You can specify the following *mm-options* after a slash (/):

DETAILS
displays a table that shows the levels of the multimember effect.

NOEFFECT
specifies that, for observations of the multimember variables whose levels are all missing, the values in the corresponding design matrix columns be set to 0. If, in the preceding example, the teacher effect is defined by the following statement, then the associated design matrix columns' values for Jack are all 0:

```
EFFECT teacher = MM(teacher1 teacher2 / noeffect);
```

This statement enables you to include Jack in the analysis even though there is no effect of teachers on his performance.

A situation where it is important to designate observations as having no effect that can be attributed to a classification variable is the analysis of crossover designs, where lagged treatment levels are used to model the carryover effects of treatments between periods. Because there is no carryover effect for the first period, the treatment lag effect in a crossover design can be modeled by using a multimember effect that consists of a single classification variable and the NOEFFECT option, as in the following statements:

```
CLASS Treatment lagTreatment;
EFFECT Carryover = MM(lagTreatment / noeffect);
```

The lagTreatment variable contains a missing value for the first period. Otherwise, it contains the value of the treatment variable for the preceding period.

STDIZE
specifies that for each observation, the entries in the design matrix that corresponds to the multimember effect be scaled to have a sum of 1.

WEIGHT=*weight-list*
specifies numeric variables used to weigh the contributions of each of the classification effects that define the constructed multimember effect. The number of variables in *weight-list* must match the number of classification variables that define the effect.

Polynomial Effects

EFFECT *effect-name*=**POLYNOMIAL (***var-list* < / *polynomial-options* >**) ;**

EFFECT *effect-name*=**POLY (***var-list* < / *polynomial-options* >**) ;**

The variables in *var-list* must be numeric. A design matrix column is generated for each term of the specified polynomial. By default, each of these terms is treated as a separate effect for the purpose of model building. For example, the following two PROC genselect,upper() steps yield the identical analysis:

```
proc genselect;
   effect MyPoly = polynomial(x1-x3/degree=2);
   model y = MyPoly;
run;
```

```
proc genselect;
   model y = x1 x2 x3 x1*x1 x1*x2 x1*x3 x2*x2 x2*x3 x3*x3;
run;
```

You can specify the following *polynomial-options* after a slash (/):

DEGREE=*n*
: specifies the degree of the polynomial, where *n* must be a positive integer and is typically a small integer, such as 1, 2, or 3. By default, DEGREE=1.

DETAILS
: displays a table that shows the details of the specified polynomial, including the number of terms generated. If you also specify the STANDARDIZE option, then a table that shows the standardization details is also produced.

MDEGREE=*n*
: specifies the maximum degree of any variable in a term of the polynomial, where *n* must be a positive integer. The default is the degree of the specified polynomial. For example, the following statement generates the terms x_1, x_2, x_1^2, x_1x_2, x_2^2, $x_1^2x_2$, $x_1x_2^2$ and $x_1^2x_2^2$:

```
EFFECT MyPoly=POLYNOMIAL(x1 x2/degree=4 MDEGREE=2);
```

NOSEPARATE
: treats the polynomial as a single effect that has multiple degrees of freedom. The *effect-name* that you specify is used as the constructed effect name, and the labels of the terms are used as labels of the corresponding parameters.

STANDARDIZE < (*centerscale-opts***) > < = *standardize-opt* >**
: standardizes the variables that define the polynomial. By default, the standardized variables receive prefix "s_" in the variable names.

  You can use the following *centerscale-opts* to specify how the center and scale are estimated:

  METHOD=MOMENTS
  : estimates the center by the variable mean and the scale by the standard deviation. If a weight variable is specified using a WEIGHT statement, the weights are not used in the computation of the mean and standard deviation, and observations that have invalid weights are ignored. Only observations that are used in performing the analysis are used for the standardization.

  METHOD=RANGE
  : estimates the center by the midpoint of the variable range and the scale as half the variable range. Any observation that has a missing value for any regressor used in the model is ignored when the range of variables in a polynomial effect is computed. Observations that have valid regressor values but missing or invalid values of frequency variables, weight variables, or dependent variables are used in computing variable ranges. By default, METHOD=RANGE.

METHOD=WMOMENTS
is the same as METHOD=MOMENTS except that weighted means and weighted standard deviations are used.

PREFIX=NONE | *quoted-string*
specifies the prefix that is appended to standardized variables when forming the term labels. You can specify the following values:

quoted-string	specifies the prefix
NONE	does not apply a prefix

By default, PREFIX="s_".

Let

$$
\begin{aligned}
n &= \text{number of observations used in the analysis} \\
w &= \text{weight variable} \\
f &= \text{frequency variable} \\
x &= \text{variable to be standardized} \\
x_{(n)} &= \text{Max}_{i=1}^{n}(x_i) \\
x_{(1)} &= \text{Min}_{i=1}^{n}(x_i) \\
F &= \text{sum of frequencies} \\
&= \Sigma_{i=1}^{n} f_i \\
\text{WF} &= \text{sum of weighted frequencies} \\
&= \Sigma_{i=1}^{n} w_i f_i
\end{aligned}
$$

Table 2.6 shows how the center and scale are computed for each of the supported methods.

Table 2.6 Center and Scale Estimates by Method

Method	Center	Scale
RANGE	$(x_{(n)} + x_{(1)})/2$	$(x_{(n)} - x_{(1)})/2$
MOMENTS	$\bar{x} = \Sigma_{i=1}^{n} f_i x_i / F$	$\sqrt{\Sigma_{i=1}^{n} f_i (x_i - \bar{x})^2/(F-1)}$
WMOMENTS	$\bar{x}_w = \Sigma_{i=1}^{n} w_i f_i x_i / \text{WF}$	$\sqrt{\Sigma_{i=1}^{n} w_i f_i (x_i - \bar{x}_w)^2/(F-1)}$

You can control whether the standardization is to center, scale, or both center and scale by specifying one of the following *standardize-opts*:

CENTER
centers but does not scale the variables. For a variable x,

$$\text{s\_x} = x - \text{center}$$

CENTERSCALE
centers and scales the variables. This is the default if you do not specify a *standardization-opt*. For a variable x,

$$\text{s\_x} = \frac{x - \text{center}}{\text{scale}}$$

NONE
performs no standardization.

SCALE
scales but does not center the variables. For a variable x,

$$\text{s\_x} = \frac{x}{\text{scale}}$$

Spline Effects

This section discusses the construction of spline effects through the EFFECT statement. A spline effect expands variables into spline bases whose form depends on the options that you specify. For more information about regression splines and spline bases, see the section "Splines and Spline Bases" on page 28. You request a spline effect with the syntax

EFFECT *name*=**SPLINE (***var-list* < / *spline-options* >**) ;**

The variables in *var-list* must be numeric. Design matrix columns are generated separately for each of these variables, and the set of columns is collectively referred to with the specified name. By default, the spline basis that is generated for each variable is a cubic B-spline basis that has three equally spaced knots positioned between the minimum and maximum values of that variable. This yields by default seven design matrix columns for each of the variables in the SPLINE effect.

You can specify the following *spline-options* after a slash (/):

BASIS=BSPLINE
uses a B-spline basis for the spline expansion. For splines of degree d that are defined with n knots, this basis consists of $n + d + 1$ columns. In order to completely specify the B-spline basis, d left-side boundary knots and $\max\{d, 1\}$ right-side boundary knots are also required. For information about how to specify the positions of both the internal and boundary knots, see the KNOTMETHOD=, DATABOUNDARY, KNOTMIN=, and KNOTMAX= suboptions.

BASIS=TPF(*options***)**
specifies a truncated power function basis for the spline expansion. For splines of degree d that are defined with n knots for a variable x, this basis consists of an intercept, polynomials $x, x^2, \ldots, x^d$, and one truncated power function for each of the n knots. Unlike the B-spline basis, no boundary knots are required. For information about how you can specify the position of the internal knots, see the KNOTMETHOD= suboption.

You can modify the number of columns when you request BASIS=TPF by specifying the following *options*:

NOINT excludes the intercept column.

NOPOWERS excludes the intercept and polynomial columns.

By default, BASIS=BSPLINE.

DATABOUNDARY
uses the extremes of the data as boundary knots when building a B-spline basis.

DEGREE=*n*
specifies the degree of the spline transformation, where *n* must be a nonnegative integer and is typically a small integer, such as 0, 1, 2, or 3. By default, DEGREE=3.

DETAILS
displays tables that show the knot locations and the knots associated with each spline basis function.

KNOTMAX=*value*
requests that, for each variable in the EFFECT statement, the right-side boundary knots be equally spaced starting at the maximum of the variable and ending at the specified value. This option is ignored for variables whose maximum value is greater than the specified value or if the DATABOUNDARY option is also specified.

KNOTMETHOD=*knot-method***<(***knot-options***)>**
specifies how to construct the knots for spline effects. You can choose from the following *knot-methods* and affect the knot construction further with the method-specific *knot-options*:

EQUAL<(*n***)>**
specifies that *n* equally spaced knots be positioned between the extremes of the data. By default, *n* = 3. For a B-spline basis, any needed boundary knots continue to be equally spaced unless the DATABOUNDARY option has also been specified. By default, KNOTMETHOD=EQUAL(3).

LIST(*number-list***)**
specifies the list of internal knots to be used in forming the spline basis columns. For a B-spline basis, the data extremes are used as boundary knots.

LISTWITHBOUNDARY(*number-list***)**
specifies the list of all knots that are used in forming the spline basis columns. When you use a truncated power function basis, this list is interpreted as the list of internal knots. When you use a B-spline basis of degree *d*, then the first *d* entries are used as left-side boundary knots and the last $\max(d, 1)$ entries in the list are used as right-side boundary knots.

MULTISCALE<(*multiscale-options***)>**
generates multiple B-spline bases, which correspond to sets that have an increasing number of internal knots. As the number of internal knots increases, the spline basis that is generated can approximate features of the data at finer scales. So generating bases at multiple scales facilitates the modeling of both coarse- and fine-grained features of the data. For scale *i*, the spline basis corresponds to 2^i equally spaced internal knots. By default, the bases for scales 0–7 are generated. For each scale, a separate spline effect is generated. The name of the constructed spline effect at scale *i* is formed by appending _S*i* to the effect name that you specify in the EFFECT statement. If you specify multiple variables in the EFFECT statement, then spline bases are generated separately for each variable at each scale and the name of the corresponding effect is obtained

by appending the variable name followed by _S*i* to the name in the EFFECT statement. For example, the following statement generates effects named spl_x1_S0, spl_x1_S1, spl_x1_S2, . . ., spl_x1_S7 and spl_x2_S1, spl_x2_S2, . . ., spl_x2_S7:

```
EFFECT spl = spline(x1 x2 / knotmethod=multiscale);
```

The MULTISCALE option is ignored if you specify the BASIS=TPF *spline-option*.

The following *multiscale-options* control which scales are included:

STARTSCALE=*n* specifies the start scale, where *n* is a positive integer. By default, STARTSCALE=0.

ENDSCALE=*n* specifies the end scale, where *n* is a positive integer. By default, ENDSCALE=7.

PERCENTILES(*n***)**

requests that internal knots be placed at *n* equally spaced percentiles of the variable or variables named in the EFFECT statement. For example, the following statement positions internal knots at the deciles of the variable x. For a B-spline basis, the extremes of the data are used as boundary knots:

```
EFFECT spl = spline(x / knotmethod=percentiles(9));
```

RANGEFRACTIONS(*fraction-list***)**

places internal knots at each fraction of the ranges of the variables in the EFFECT statement. For example, if variable x1 ranges between 1 and 3 and variable x2 ranges between 0 and 20, then the following EFFECT statement uses internal knots 1.2, 2, and 2.5 for variable x1 and internal knots 2, 10, and 15 for variable x2:

```
EFFECT spl = spline(x1 x2 / knotmethod=rangefractions(.1 .5 .75));
```

For a B-spline basis, the data extremes are used as boundary knots.

KNOTMIN=*value*

requests that, for each variable in the EFFECT statement, the left-side boundary knots be equally spaced starting at the specified value and ending at the minimum of the variable. This option is ignored for variables whose minimum value is less than the specified value or if the DATABOUNDARY option is also specified.

NATURALCUBIC

uses a natural cubic spline basis for the spline expansion. Natural cubic splines, also known as restricted cubic splines, are cubic splines that are constrained to be linear beyond the extreme knots. The natural cubic spline basis that is produced by the EFFECT statement is obtained by starting from the unrestricted truncated power function cubic spline basis that is defined with n distinct knots and imposes the linearity constraints beyond the extreme knots. This basis consists of an intercept, the polynomial x, and $n - 2$ functions that are all linear beyond the largest knot. The ith function, $i = 1, 2, \ldots, n - 2$, is 0 to the left of the ith knot, which is called the "break knot." For more information about this basis, see the section "Splines and Spline Bases" on page 28. You can use the NOINT and NOPOWERS suboptions of the BASIS=TPF option to suppress the intercept and

polynomial x when the columns of the natural cubic spline basis are formed. When you specify the NATURALCUBIC option, the options BASIS=BSPLINE, DATABOUNDARY, DEGREE=, and KNOTMETHOD=MULTISCALE are not applicable.

SEPARATE

requests that, when multiple variables are specified in the EFFECT statement, the spline basis for each variable be treated as a separate effect. The names of these separated effects are formed by appending an underscore followed by the name of the variable to the name that you specify in the EFFECT statement. For example, the following statement generates the effect names spl_x1 and spl_x2:

```
EFFECT spl = spline(x1 x2 / separate);
```

In procedures that support variable selection, such as the GENSELECT procedure, these two effects can enter or leave the model independently during the selection process.

SPLIT

treats each individual column in the design matrix that corresponds to the spline effect as a separate effect that can enter or leave the model independently. Names for these split effects are generated by appending the variable name and an index for each column to the name that you specify in the EFFECT statement. For example, the effects generated for the spline effect in the following statement are spl_x1:1, spl_x1:2, ..., spl_x1:7 and spl_x2:1, spl_x2:2, ..., spl_x2:7:

```
EFFECT spl = spline(x1 x2 / split);
```

Splines and Spline Bases

This section provides details about how the EFFECT statement constructs spline bases. A spline function is a piecewise polynomial function in which the individual polynomials have the same degree and connect smoothly at join points whose abscissa values, called knots, are prespecified. You can use spline functions to fit curves to a wide variety of data.

A spline of degree 0 is a step function with steps located at the knots. A spline of degree 1 is a piecewise linear function where the lines connect at the knots. A spline of degree 2 is a piecewise quadratic curve whose values and slopes coincide at the knots. A spline of degree 3 is a piecewise cubic curve whose values, slopes, and curvature coincide at the knots. Visually, a cubic spline is a smooth curve, and it is the most commonly used spline when a smooth fit is desired. Note that when no knots are used, splines of degree d are simply polynomials of degree d.

More formally, suppose you specify knots $k_1 < k_2 < k_3 < \cdots < k_n$. Then a spline of degree $d \geq 0$ is a function $S(x)$ with d–1 continuous derivatives such that

$$S(x) = \begin{cases} P_0(x) & x < k_1 \\ P_i(x) & k_i \leq x < k_{i+1};\ i = 1, 2, \ldots, n-1 \\ P_n(x) & x \geq k_n \end{cases}$$

where each $P_i(x)$ is a polynomial of degree d. The requirement that $S(x)$ has $d - 1$ continuous derivatives is satisfied by requiring that the function values and all derivatives up to order $d - 1$ of the adjacent polynomials at each knot match.

A counting argument yields the number of parameters that define a spline with n knots. There are $n + 1$ polynomials of degree d, producing $(n + 1)(d + 1)$ coefficients. However, there are d restrictions at each of the n knots, so the number of free parameters is $(n + 1)(d + 1) - nd = n + d + 1$. In mathematical terminology this says that the dimension of the vector space of splines of degree d on n distinct knots is $n + d + 1$. If you have $n + d + 1$ basis vectors, then you can fit a curve to your data by regressing your dependent variable by using this basis for the corresponding design matrix columns. In this context, such a spline is known as a regression spline. The EFFECT statement provides a simple mechanism for obtaining such a basis.

If you remove the restriction that the knots of a spline must be distinct and allow repeated knots, then you can obtain functions that have less smoothness and even discontinuities at the repeated knot location. For a spline of degree d and a repeated knot that has multiplicity $m \leq d$, the piecewise polynomials that join such a knot are required to have only $d - m$ matching derivatives. Note that this increases the number of free parameters by $m - 1$ but also decreases the number of distinct knots by $m - 1$. Hence the dimension of the vector space of splines of degree d with n knots is still $n + d + 1$, provided that any repeated knot has a multiplicity less than or equal to d.

The EFFECT statement supports the commonly used *truncated power function* basis and *B-spline* basis. With exact arithmetic and by using the complete basis, you obtain the same fit with either of these bases. The following subsections provide details about constructing spline bases for the space of splines of degree d with n knots that satisfies $k_1 \leq k_2 \leq k_3 < \cdots \leq k_n$.

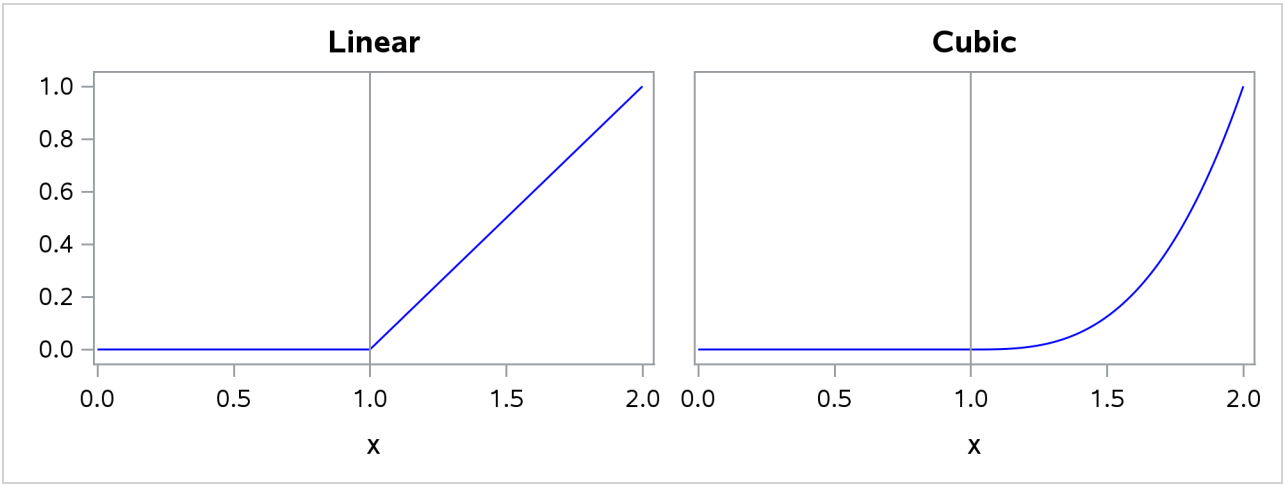
Truncated Power Function Basis

A truncated power function for a knot k_i is a function defined by

$$t_i(x) = \begin{cases} 0 & x < k_i \\ (x - k_i)^d & x \geq k_i \end{cases}$$

Figure 2.1 shows such functions for $d = 1$ and $d = 3$ with a knot at $x = 1$.

Figure 2.1 Truncated Power Functions with Knot at $x = 1$



The name is derived from the fact that these functions are shifted power functions that are truncated to 0 to the left of the knot. These functions are piecewise polynomial functions that have two pieces whose function values and derivatives of all orders up to d–1 are 0 at the defining knot. Hence these functions are splines of

degree d. It is easy to see that these n functions are linearly independent. However, they do not form a basis, because such a basis requires $n+d-1$ functions. The usual way to add $d+1$ additional basis functions is to use the polynomials $1, x, x^2, \ldots, x^d$. These $d+1$ functions together with the n truncated power functions $t_i(x), i = 1, 2, \ldots, n$ form the truncated power basis.

Note that each time a knot is repeated, the associated exponent used in the corresponding basis function is reduced by 1. For example, for splines of degree d with three repeated knots $k_i = k_{i+1} = k_{i+2}$, the corresponding basis functions are $t_i(x) = (x - k_i)_+^d$, $t_{i+1}(x) = (x - k_i)_+^{d-1}$, and $t_{i+2}(x) = (x - k_i)_+^{d-2}$. Provided that the multiplicity of each repeated knot is less than or equal to the degree, this construction continues to yield a basis for the associated space of splines.

The main advantage of the truncated power function basis is the simplicity of its construction and the ease of interpreting the parameters in a model that corresponds to these basis functions. However, there are two weaknesses when you use this basis for regression. These functions grow rapidly without bound as x increases, resulting in numerical precision problems when the x data span a wide range. Furthermore, many or even all of these basis functions can be nonzero when evaluated at some x value, resulting in a design matrix that has few zeros and precludes the use of sparse matrix technology to speed up computation. This weakness can be addressed by using a B-spline basis.

B-Spline Basis

A B-spline basis can be built by starting with a set of Haar basis functions, which are functions that are 1 between adjacent knots and 0 elsewhere, and then applying a simple linear recursion relationship d times, yielding the $n + d + 1$ needed basis functions. For the purpose of building the B-spline basis, the n prespecified knots are called internal knots. This construction requires d additional knots, known as boundary knots, to be positioned to the left of the internal knots, and $\max(d, 1)$ boundary knots to be positioned to the right of the internal knots. The actual values of these boundary knots can be arbitrary. The EFFECT statement provides several methods for placing the necessary boundary knots, including the common method of using repeated values of the data extremes as the boundary knots. The boundary knot placement affects the precise form of the basis functions that are generated, but it does not affect the following two desirable properties:

- The B-spline basis functions are nonzero over an interval that spans at most $d + 2$ knots. This yields design matrix columns each of whose rows contain at most $d + 2$ adjacent nonzero entries.
- The computation of the basis functions at any x value is numerically stable and does not require evaluating powers of this value.

The following figures show the B-spline bases that are defined on $[0, 1]$ with four equally spaced internal knots at 0.2, 0.4, 0.6, and 0.8.

Figure 2.2 shows a linear B-spline basis. This basis consists of six functions, each of which is nonzero over an interval that spans at most three knots.

Figure 2.2 Linear B-Spline Basis with Four Equally Spaced Interior Knots

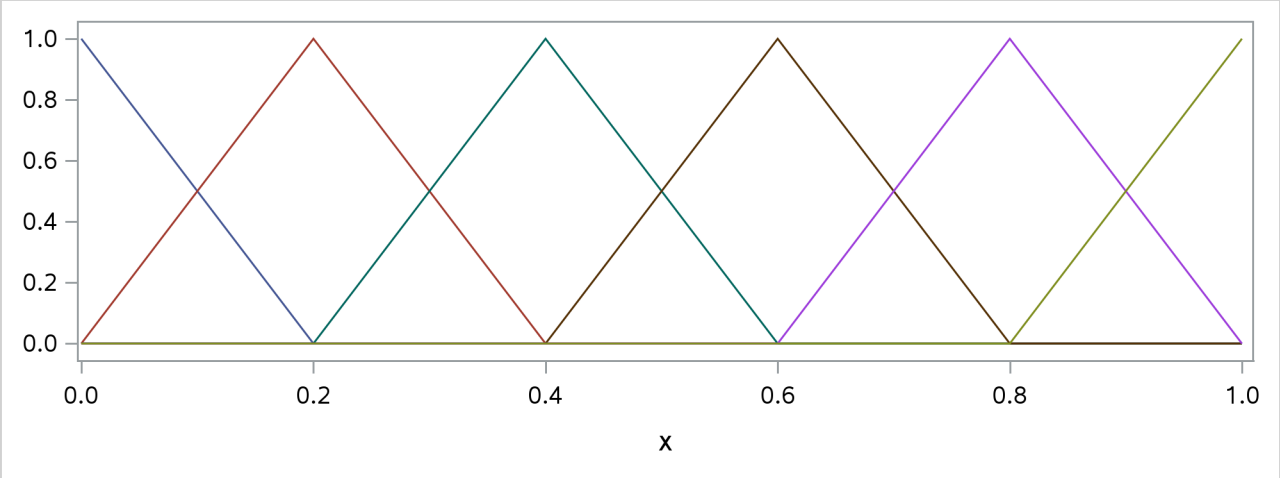


Figure 2.3 shows a cubic B-spline basis where the necessary boundary knots are positioned at $x = 0$ and $x = 1$. This basis consists of eight functions, each of which is nonzero over an interval that spans at most five knots.

Figure 2.3 Cubic B-Spline Basis with Four Equally Spaced Interior Knots

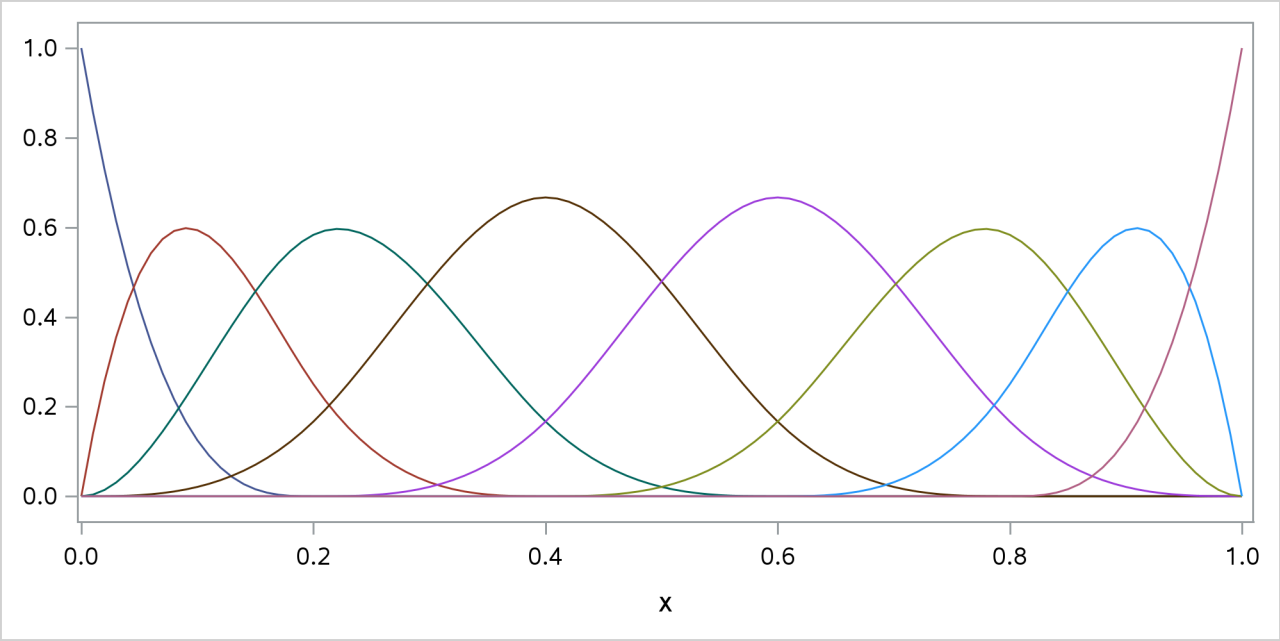
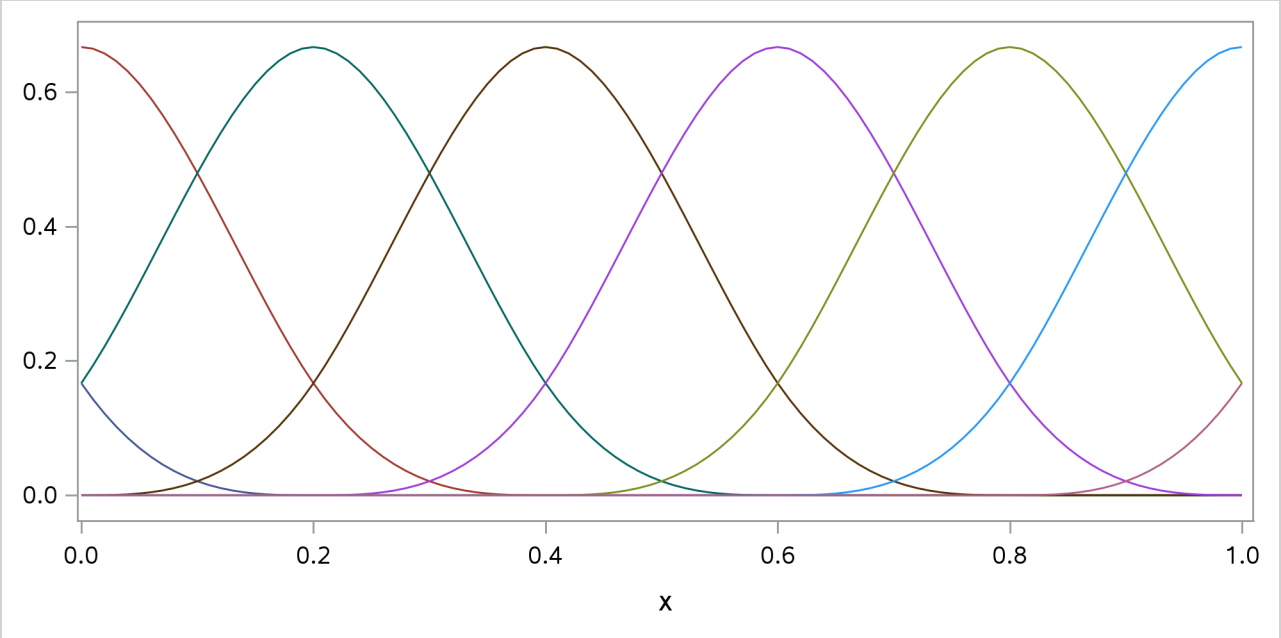


Figure 2.4 shows a different cubic B-spline basis where the necessary left-side boundary knots are positioned at –0.6, –0.4, –0.2, and 0. The right-side boundary knots are positioned at 1, 1.2, 1.4, and 1.6. As in the basis shown in Figure 2.3, this basis consists of eight functions, each of which is nonzero over an interval that spans at most five knots. The different positioning of the boundary knots has merely changed the shape of the individual basis functions.

Figure 2.4 Cubic B-Spline Basis with Equally Spaced Boundary and Interior Knots

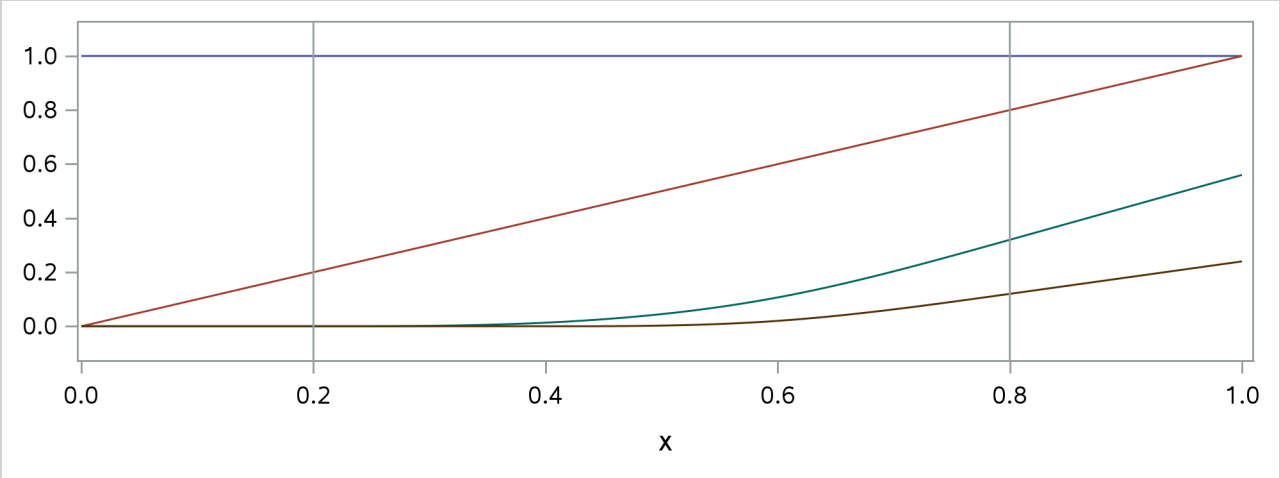


For more information about this construction, see Hastie, Tibshirani, and Friedman (2001).

Natural Cubic Spline Basis

Natural cubic splines are cubic splines with the additional restriction that the splines are required to be linear beyond the extreme knots. Some authors prefer the terminology "restricted cubic splines" to "natural cubic splines." The space of unrestricted cubic splines on n knots has the dimension $n + 4$. Imposing the restrictions that the cubic polynomials beyond the first and last knot reduce to linear polynomials reduces the number of degrees of freedom by 4, so a basis for the natural cubic splines consists of n functions. Starting from the truncated power function basis for the unrestricted cubic splines, you can obtain a reduced basis by imposing linearity constraints. For more information about this construction, see Hastie, Tibshirani, and Friedman (2001). Figure 2.5 shows this natural cubic spline basis defined on $[0, 1]$ with four equally spaced internal knots at 0.2, 0.4, 0.6, and 0.8. This basis consists of four basis functions that are all linear beyond the extreme knots at 0.2 and 0.8.

Figure 2.5 Natural Cubic Spline Basis with Four Equally Spaced Knots



ODS Table Names

Each constructed effect produces one or more informative tables when the DETAILS option is also specified. Each table has a name associated with it. This name is used to refer to the table by procedures that support a DISPLAY or DISPLAYOUT statement. You also use this name to refer to the table in ODS statements. These names are listed in Table 2.7.

Table 2.7 ODS Tables Produced by the EFFECT Statement

Table Name	Description	Option
BSplineDetails	B-spline basis details	BASIS=BSPLINE
CollectionLevelInfo	Levels of collection effects	
MMLevelInfo	Levels of multimember effects	
PolyDetails	Number of variables and columns, polynomial degree, and standardization method	
PolyScaling	Centering and scaling details	STANDARDIZE
SplineKnots	knot and boundary knot values	
TPFSplineDetails	Truncated power function spline basis details	BASIS=TPF or NATURALCUBIC

LSMEANS Statement

LSMEANS *model-effects* < / *options* > **;**

LSMEANS EFFECT=*model-effects* < / *options* > **;**

This section applies to the following procedures: GENSELECT, LMIXED, and LOGSELECT.

The LSMEANS statement computes least squares means (LS-means) of classification fixed effects. As in the GLM procedure, LS-means are *predicted population margins*—that is, they estimate the marginal means over a balanced population. In a sense, LS-means are to unbalanced designs as classification and subclassification arithmetic means are to balanced designs. You can view LS-means as linear combinations of the parameter estimates that are constructed in such a way that they correspond to average predicted values in a population where the levels of classification variables are balanced.

Each LS-mean is computed as $\mathbf{L}\widehat{\boldsymbol{\beta}}$, where $\mathbf{L}$ is the coefficient matrix that is associated with the least squares mean and $\widehat{\boldsymbol{\beta}}$ is the estimate of the fixed-effects parameter vector. This $\mathbf{L}$ matrix is the same as the $\mathbf{L}$ matrix that is constructed in PROC GLM; however, the standard errors are adjusted for the covariance parameters in the model. The approximate standard errors for the LS-mean are computed as the square root of $\mathbf{L}(\mathbf{X}'\widehat{\mathbf{V}}^{-1}\mathbf{X})^{-}\mathbf{L}'$. The approximate variance matrix of the fixed-effects estimates depends on the estimation method.

LS-means can be computed for any effect in the procedure's MODEL statement that involves only classification variables. You can specify multiple *model-effects* in one LSMEANS statement or in multiple LSMEANS statements, and all LSMEANS statements must appear after the MODEL statement. When you restore a model from an item store by using PROC GENSELECT or PROC LOGSELECT, then you must specify the desired *model-effects* in the EFFECT= option before the slash (/).

The $\mathbf{L}$ matrix is tested for estimability, and if this test fails, the procedure displays "Non-est" for the LS-means entries. Note that linear functions of LS-means, such as differences, can be estimable even when the means themselves are not estimable. Estimability checks for differences are thus applied separately from checks for the means. Assuming that the LS-mean is estimable, the procedure constructs an approximate normal test or t test (depending on the procedure) to test the null hypothesis that the associated population quantity equals zero. By default, the denominator degrees of freedom for the t test are the residual degrees of freedom.

Table 2.8 summarizes the *options* that are available in the LSMEANS statement. All *options* are subsequently discussed in alphabetical order. Note that the DF= and SLICE= options are available only with the LMIXED procedure.

Table 2.8 LSMEANS Statement Options

Option	Description
Construction and Computation of LS-Means	
AT	Modifies covariate values used to compute LS-means
ATVAR	Modifies covariate values for restored models
DIFF	Computes differences of LS-means
SINGULAR=	Tunes estimability checking
SLICE=	Partitions interaction LS-means effects

Table 2.8 *continued*

Option	Description
Degrees of Freedom and *p*-Values	
ADJUST=	Specifies the multiple comparison adjustment method for LS-means differences
ALPHA=α	Specifies the confidence level ($1-\alpha$)
DF=	Assigns a specific value to the degrees of freedom for tests and confidence limits
Statistical Output	
CL	Constructs confidence limits for means and mean differences
CORR	Displays the correlation matrix of LS-means
COV	Displays the covariance matrix of LS-means
E	Displays the **L** matrix

You can specify the following options in the LSMEANS statement after a slash (/):

ADJUST=*method*

computes a multiple comparison adjustment for the *p*-values and confidence limits for the differences of LS-means. The adjusted quantities are produced along with the unadjusted quantities. The ADJUST= option works together with the DIFF option. If you omit the DIFF option and specify a *method* other than NONE, then the ADJUST= option invokes the DIFF option.

You must specify one of the following *methods* for the ADJUST= option:

BON

performs Bonferroni *t* tests of differences between LS-means. The method involves correction factors described in Chapter 53, "The GLM Procedure" (*SAS/STAT User's Guide*), and Chapter 86, "The MULTTEST Procedure" (*SAS/STAT User's Guide*); also see Westfall and Young (1993) and Westfall et al. (1999).

DUNNETT

performs Dunnett's *t* test, which tests whether any treatments are significantly different from a single control for the effects in the LSMEANS statement. When the LS-means are correlated, the procedure uses the factor-analytic covariance approximation described in Hsu (1992) and identifies the adjustment as "Dunnett-Hsu" in the results. The approximation derives approximate "effective sample sizes" for which exact critical values are computed.

NELSON

performs Nelson's *t* test of each LS-mean against an average of the LS-means (Ott 1967; Nelson 1982, 1991, 1993). When the LS-means are correlated, the procedure uses the factor-analytic covariance approximation described in Hsu (1992) and identifies the adjustment as "Nelson-Hsu" in the results. The approximation derives approximate "effective sample sizes" for which exact critical values are computed.

NONE | T
performs no adjustment for multiple comparisons.

SCHEFFE
performs Scheffé's multiple comparison procedure.

SIDAK
performs pairwise t tests on differences between LS-means with levels adjusted according to Šidák's inequality. The method involves correction factors described in Chapter 53, "The GLM Procedure" (*SAS/STAT User's Guide*), and Chapter 86, "The MULTTEST Procedure" (*SAS/STAT User's Guide*); also see Westfall and Young (1993) and Westfall et al. (1999).

SIMULATE < (*simopts*) >
performs the simulation-based multiple comparison procedure. This method computes adjusted p-values and confidence limits from the simulated distribution of the maximum or maximum absolute value of a multivariate t random vector. All covariance parameters, except the residual scale parameter, are fixed at their estimated values throughout the simulation, potentially resulting in some underdispersion. The simulation estimates q, the true $(1-\alpha)$ quantile, where $1-\alpha$ is the confidence coefficient. The default value of α is 0.05; you can change this value by specifying the ALPHA= option in the LSMEANS statement.

The number of samples is set so that the tail area for the simulated q is within γ of $1-\alpha$ with $100(1-\epsilon)\%$ confidence. In equation form,

$$\Pr(|F(\widehat{q}) - (1-\alpha)| \leq \gamma) = 1-\epsilon$$

where $\hat{q}$ is the simulated q and F is the true distribution function of the maximum; see Edwards and Berry (1987) for details. By default, $\gamma = 0.005$ and $\epsilon = 0.01$, placing the tail area of $\hat{q}$ within 0.005 of 0.95 with 99% confidence.

You can specify the following *simopts* options in parentheses after the ADJUST=SIMULATE option:

ACC=*value*
specifies the target accuracy radius γ of a $100(1-\epsilon)\%$ confidence interval for the true probability content of the estimated $(1-\alpha)$ quantile. By default, ACC=0.005. Note that if you also specify the CVADJUST *simoption*, then the actual accuracy radius will probably be substantially less than this target.

CVADJUST
estimates the quantile by the control variate adjustment method of Hsu and Nelson (1998) instead of simply as the quantile of the simulated sample. Specifying this option usually has the effect of significantly reducing the accuracy radius γ of a $100 \times (1-\epsilon)\%$ confidence interval for the true probability content of the estimated $(1-\alpha)$ quantile. The control-variate-adjusted quantile estimate takes approximately twice as long to compute, but it is typically much more accurate than the sample quantile.

EPS=*value*
specifies the value ϵ for a $100 \times (1 - \epsilon)\%$ confidence interval for the true probability content of the estimated $(1 - \alpha)$ quantile. The default value of the accuracy confidence is 99%, corresponding to EPS=0.01.

NSAMP=*n*
specifies the sample size for the simulation. By default, *n* is set by a formula that is based on the values of the target accuracy radius γ and accuracy confidence $100 \times (1 - \epsilon)\%$ for an interval for the true probability content of the estimated $(1 - \alpha)$ quantile. When the default values are used for γ, ϵ, and α (0.005, 0.01, and 0.05, respectively), NSAMP=12605. If you omit the NSAMP= option or specify NSAMP=0, the default sample size is used.

REPORT
displays a report on the simulation, including a listing of the parameters, such as γ, ϵ, and α, as well as an analysis of various methods of estimating or approximating the quantile.

SEED=*number*
specifies an integer to be used to start the pseudorandom number generator for the simulation. If you do not specify a seed, or if you specify a *number* less than or equal to 0, the seed is generated by reading the time of day from the computer's clock.

SMM | GT2
performs pairwise comparisons based on the studentized maximum modulus and Šidák's uncorrelated-*t* inequality, yielding Hochberg's GT2 method when sample sizes are unequal.

T | NONE
performs no adjustment for multiple comparisons.

TUKEY
performs Tukey's studentized range test on LS-means. When your data are unbalanced, the procedure uses the approximation described in Kramer (1956) and identifies the adjustment as "Tukey-Kramer" in the results.

By default, ADJUST=NONE.

If you specify ADJUST=DUNNETT, the procedure analyzes all differences with the first control level unless you have specified a different control level. If you specify ADJUST=NELSON, then DIFF=ANOM is assumed.

If you specify ADJUST=TUKEY, then DIFF=ALL is assumed. If you specify other *methods*, then the procedure performs all pairwise differences unless you specify the DIFF option.

Note that computing the exact adjusted *p*-values and critical values for unbalanced designs can be computationally intensive, especially for ADJUST=NELSON.

ALPHA=*number*
specifies that a confidence interval be constructed for each of the LS-means with confidence level $1 -$ *number*. The value of *number* is between 0 and 1; the default is 0.05.

AT *at-specification*

enables you to modify the values of the covariates that are used in computing LS-means. By default, all covariates are set equal to their mean values for computing standard LS-means. The AT option enables you to assign arbitrary values to the covariates. Additional columns in the output table indicate the values of the covariates.

You can specify the following *at-specifications*:

MEANS

sets covariates equal to their mean values (as with standard LS-means) and applies this adjustment to crossproducts of covariates.

variable=*value*

sets the covariate *variable* equal to *value*.

(*variable-list*)=(*value-list*)

sets covariates in the *variable-list* equal to values in the *value-list*.

Consider the following example:

```
class A B;
model Y = A*B x1 x2 x1*x2;
LSMEANS A*B;
LSMEANS A*B / at means;
LSMEANS A*B / at x1=1.2;
LSMEANS A*B / at (x1 x2)=(1.2 0.3);
```

For the first two LSMEANS statements, the LS-means coefficient of x1 is $\overline{x}_1$ (the mean of x1) and of x2 is $\overline{x}_2$ (the mean of x2). For the first LSMEANS statement, the coefficient of x1*x2 is $\overline{x_1 x_2}$. However, for the second LSMEANS statement, the coefficient is $\overline{x}_1 \times \overline{x}_2$. The third LSMEANS statement sets the coefficient of x1 equal to 1.2 and leaves it at $\overline{x}_2$ for x2, and the final LSMEANS statement sets these values to 1.2 and 0.3, respectively.

Even if you specify a weight variable, the unweighted covariate means are used for the covariate coefficients if you omit the AT option. If you specify the AT option, then weight and frequency variables are taken into account as follows. The weighted covariate means are used for the covariate coefficients for which no explicit AT option values are given, or if you specify the AT MEANS option. Observations that do not contribute to the analysis because of a missing dependent variable are used in computing the covariate means. You should use the E option in conjunction with the AT option to verify that the modified LS-means coefficients are the ones that you want.

ATVAR *at-specification*

is the same as the AT option, except that it is required when you restore a model from an item store but do not provide a data table. This option is available with the GENSELECT and LOGSELECT procedures.

CL

constructs confidence limits for each of the LS-means. The LMIXED procedure computes *t*-type limits, whereas other procedures compute normal (z) intervals. If you specify the DDFM=NONE option in the MODEL statement in PROC LMIXED, then infinite degrees of freedom are used for this test, which essentially computes a z interval. The confidence level is 0.95 by default; you can change the level by specifying the ALPHA= option.

CORR

displays the estimated correlation matrix of the least squares means as part of the "Least Squares Means" table.

COV

displays the estimated covariance matrix of the least squares means as part of the "Least Squares Means" table.

DF=*number*

specifies the degrees of freedom for the *t* test and confidence limits. The default is the residual degrees of freedom that you define by specifying the DDFM=RESIDUAL option in the MODEL statement in the LMIXED procedure. This option is available only for PROC LMIXED.

DIFF<=*difftype***>**

displays differences of the LS-means in the "Differences of Least Squares Means" table. You can specify the following values for the optional *difftype*:

ALL

displays all pairwise differences.

ANOM

displays differences between each LS-mean and the average LS-mean, as in the analysis of means (Ott 1967). The average is computed as a weighted mean of the LS-means, with the weights being inversely proportional to the diagonal entries of the $\mathbf{L}\left(\mathbf{X}'\mathbf{X}\right)^{-}\mathbf{L}'$ matrix. When you specify a WEIGHT statement, the preceding matrix is replaced with $\mathbf{L}\left(\mathbf{X}'\mathbf{W}\mathbf{X}\right)^{-}\mathbf{L}'$, where $\mathbf{W}$ is the diagonal matrix that contains the weights. If LS-means are nonestimable, this design-based weighted mean is replaced with an equally weighted mean. Note that the ANOM procedure in SAS/QC software implements both tables and graphics for the analysis of means with a variety of response types. For one-way designs and normally distributed data, the ANOM computations are equivalent to the results of PROC ANOM.

CONTROL<('level1'<,'level2',...>)>

produces two-tailed tests and confidence limits for differences with a control; by default, the control is the first level of each specified LS-mean effect. To specify which levels of the effects are the controls, list the quoted formatted values in parentheses after the CONTROL keyword. For example, if the effects A, B, and C are classification variables, each having two levels, 1 and 2, the following LSMEANS statement specifies the (1,2) level of A*B and the (2,1) level of B*C as controls:

```
lsmeans A*B B*C / diff=control('1' '2', '2' '1');
```

For multiple effects, the results depend on the order of the list, so you should check the output to make sure that the controls are correct.

CONTROLL
produces one-tailed results and tests whether the noncontrol levels are significantly smaller than the control. The upper confidence limits for the control minus the noncontrol levels are considered to be infinity and are displayed as missing.

CONTROLU
produces one-tailed results and tests whether the noncontrol levels are significantly larger than the control. The upper confidence limits for the noncontrol levels minus the control are considered to be infinity and are displayed as missing.

E
displays the matrix coefficients for all LS-mean effects in the "Matrix Coefficients" table.

PDIFF<=*difftype*>
is the same as the DIFF option.

SINGULAR=*number*
tunes the estimability checking. If **v** is a vector, define ABS(**v**) to be the largest absolute value of the elements of **v**. If ABS($\mathbf{L} - \mathbf{LT}$) is greater than *c***number* for any row of **L** in the contrast, then $\mathbf{L}\boldsymbol{\beta}$ is declared nonestimable. Here, **T** is the Hermite form matrix $(\mathbf{X}'\mathbf{X})^{-}\mathbf{X}'\mathbf{X}$, and *c* is ABS(**L**), except when it equals 0, and then *c* is 1. The value of *number* is between 0 and 1; the default is 1E–4.

SLICE=*effect1* <,*effect2*, ... >}
specifies effects by which to partition interaction LS-mean effects. This option is available only for the LMIXED procedure. This partition can produce what are known as tests of simple effects (Winer 1971). For example, suppose that A*B is significant, and you want to test the effect of A for each level of B. The appropriate LSMEANS statement is

```
lsmeans A*B / slice=B;
```

This statement tests for the simple main effects of A for B, which are calculated by extracting the appropriate rows from the coefficient matrix for the A*B LS-means and by using them to form an *F* test.

The SLICE= option produces *F* tests that test the simultaneous equality of cell means at a fixed level of the slice effect (Schabenberger, Gregoire, and Kong 2000). It produces a table titled "Tests of Effect Slices."

ODS Table Names: LSMEANS Statement

Each table that the LSMEANS statement creates has a name associated with it. You can use this name to refer to the table when using the Output Delivery System (ODS) to select tables and create output data sets. The table names are listed in Table 2.9. For more information about ODS, see Chapter 23, "Using the Output Delivery System" (*SAS/STAT User's Guide*).

Table 2.9 ODS Tables Produced by the LSMEANS Statement

Table Name	Description	Required Option
Coef	**L** matrix coefficients	E
Diffs	Differences of least squares means	DIFF or ADJUST=
LSMeans	Least squares means	Default
SimDetails	Details for difference quantile simulation	ADJUST=SIMULATE(REPORT)
SimResults	Evaluation of difference quantile simulation	ADJUST=SIMULATE(REPORT)
Slices	Tests of LS-means effect slices	SLICE=

PARTITION Statement

PARTITION *partition-option* **;**

This section applies to the following procedures: GAMSELECT, GENSELECT, LOGSELECT, PHSELECT, QTRSELECT, REGSELECT, and TREESPLIT.

The PARTITION statement specifies how observations in the input data set are logically partitioned into disjoint subsets for model training, validation, and testing. For more information, see the section "Using Validation and Test Data" on page 101. Either you can designate a variable in the input data table and a set of formatted values of that variable to determine the role of each observation, or you can specify proportions to use for randomly assigning observations to each role.

You must specify exactly one of the following *partition-options*:

FRACTION(< TEST=*fraction* **> < VALIDATE=***fraction* **> < SEED=***number* **>)**
randomly assigns specified proportions of the observations in the input data table to the roles. You specify the proportions for testing and validation by using the TEST= and VALIDATE= suboptions. If you specify both the TEST= and VALIDATE= suboptions, then the sum of the specified fractions must be less than 1 and the remaining fraction is the proportion of the observations assigned to the training role. The SEED= option specifies an integer that is used to start the pseudorandom number generator for random partitioning of data for training, testing, and validation. If you do not specify SEED=*number* or if *number* is less than or equal to 0, the seed is generated by reading the time of day from the computer's clock.

ROLE=*variable* **(< TEST='***value***' > < TRAIN='***value***' > < VALIDATE='***value***' >)**

ROLEVAR=*variable* **(< TEST='***value***' > < TRAIN='***value***' > < VALIDATE='***value***' >)**
names the *variable* in the input data table whose values are used to assign roles to each observation. This *variable* cannot also appear as an analysis variable in other statements or options. The TEST=, TRAIN=, and VALIDATE= suboptions specify the formatted values of this variable that are used to assign observation roles. If you do not specify the TRAIN= suboption, then all observations whose role is not determined by the TEST= or VALIDATE= suboption are assigned to the training role.

SELECTION Statement

SELECTION < **METHOD=***method*< **(***method-options***)** > >< *options* > **;**

This section applies to the following procedures: GENSELECT, LOGSELECT, PHSELECT, QTRSELECT, and REGSELECT.

Procedures in this book that support model selection use the SELECTION statement to control details about the model selection process.

You can specify the following methods in the SELECTION statement:

METHOD=*method* **<(***method-options***)>**
specifies the method to be used to select the model. You can also specify *method-options* to apply to the specified method by enclosing them in parentheses after the *method*.

The following *methods* are available and are explained in detail in the section "Model Selection Methods" on page 75. By default, METHOD=STEPWISE.

BACKWARD specifies backward elimination. This method starts with all effects in the model and deletes effects.

BESTSUBSET specifies best-subset selection. This method uses the branch-and-bound technique (Furnival and Wilson 1974) to find the best subsets of model effects without examining all possible subsets. The only SAS Viya procedure that supports this method is the REGSELECT procedure.

ELASTICNET specifies the elastic net method. This is an extension of LASSO that estimates parameters by using constrained optimization, in which both the sum of the absolute regression coefficients and the sum of the squared regression coefficients are constrained. The SAS Viya procedures that support this method are the GENSELECT, LOGSELECT, and REGSELECT procedures.

FORWARD specifies forward selection. This method starts with no effects in the model and adds effects.

FORWARDSWAP specifies forward-swap selection, which is an extension of the forward selection method. Before any addition step, the procedure makes all pairwise swaps of one effect in the model and one effect out of the current model that improve the selection criterion. When the selection criterion is an R-square, this method is the same as the MAXR method in the REG procedure in SAS/STAT software. The only SAS Viya procedure that supports this method is the REGSELECT procedure.

LAR specifies least angle regression. Like forward selection, this method starts by adding effects to an empty model. The parameter estimates at any step are "shrunk" when they are compared to the corresponding least squares estimates. If the model contains classification variables, then these classification variables are split. For more information, see the SPLIT option in the CLASS statement. The only SAS Viya procedure that supports this method is the REGSELECT procedure.

LASSO specifies the LASSO method, which adds and deletes parameters by using a version of ordinary least squares in which the sum of the absolute regression coefficients is constrained. If the model contains classification variables, then these classification variables are split. For more information, see the SPLIT option in the CLASS statement.

MCP specifies the MCP method, which minimizes ordinary least squares plus the minimax concave penalty (MCP) function. As an alternative to the convex penalized methods such as LASSO and ELASTICNET, MCP is one of the nonconvex penalized methods. The only SAS Viya procedure that supports this method is the REGSELECT procedure.

NONE specifies no model selection.

SCAD specifies the SCAD method, which minimizes ordinary least squares plus the smoothly clipped absolute deviation (SCAD) function. As an alternative to the convex penalized methods such as LASSO and ELASTICNET, SCAD is one of the nonconvex penalized methods. The only SAS Viya procedure that supports this method is the REGSELECT procedure.

STEPWISE specifies stepwise regression. This method is similar to the FORWARD method except that effects already in the model do not necessarily stay there.

Note that not all selection *methods* are applicable to every procedure. Table 2.10 lists the applicable *methods* for each of the procedures.

Table 2.10 Applicable *methods* by Procedure

method	GENSELECT	LOGSELECT	PHSELECT	QTRSELECT	REGSELECT
BACKWARD	x	x	x	x	x
BESTSUBSET					x
ELASTICNET	x	x			x
FORWARD	x	x	x	x	x
FORWARDSWAP					x
LAR					x
LASSO	x	x	x		x
MCP					x
NONE	x	x	x	x	x
SCAD					x
STEPWISE	x	x	x	x	x

Table 2.11 lists the applicable *method-options* for each of these methods.

Table 2.11 Applicable *method-options* by *method*

method-option	BACKWARD	BESTSUBSET	ELASTICNET	FORWARD	LAR	LASSO	MCP/SCAD	STEPWISE
ABSFCONV=			x					
ADAPTIVE			x			x		
AIC		x						
ALPHA=							x	
B		x						
BEST		x						
BIC		x						
BIGM=							x	
CHOOSE=	x		x	x	x	x	x	x
COEFTOL=							x	
COMPETITIVE								x
CRITERION=	x	x		x				x
ENSCALE			x					
FAST	x							
FCONV=			x					
GCONV=			x					
GMSEP		x						
INTTOL=							x	
JP		x						
LAMBDA=			x				x	
LAMBDAGRID=							x	
LSCOEFFS					x	x		
L2=			x					
L2HIGH=			x					
L2LOW=			x					
L2STEPS=			x					
MAXALPHA=							x	
MAXEFFECTS=		x	x	x	x	x		x
MAXITERALPHA=							x	
MAXITERLAMBDA=							x	
MAXLAMBDA=							x	
MAXSTEPS=	x		x	x	x	x		x
MAXTIME=							x	
MINALPHA=							x	
MINEFFECTS=	x	x						
MINLAMBDA=							x	
MIXING=			x					
MSE		x						
NLAMBDA=			x					
RELAXED						x		
PC		x						
RHO=			x					
RMSE		x						
SBC		x						
SCALE=							x	
SELECT=	x	x		x				x
SIGMA=		x						
SLENTRY=			x	x	x	x		x
SLSTAY=	x		x			x		x
SOLVER=			x				x	
SP		x						
SSE		x						
STOP=	x		x	x	x	x		x

The syntax of the *method-options* that you can specify in parentheses after the SELECTION= option *method* follows. As described in Table 2.11, not all selection *method-options* are applicable to every *method*.

ABSFCONV=*value*
specifies an absolute function difference convergence criterion. This *method-option* is available only when the solver is the alternating direction method of multipliers (ADMM) for elastic net selection.

ADAPTIVE < (GAMMA=*nonnegative number***) >**
applies adaptive weights to each of the coefficients when METHOD=LASSO or METHOD=ELASTICNET. Ordinary least squares estimates of the model parameters are used to form the adaptive weights. You use the GAMMA= suboption to specify the power transformation that is applied to the parameters in forming the adaptive weights. By default, GAMMA=1.

AIC
displays the AIC statistic for each model in the SubsetSelectionSummary table. This *method-option* is available only for the METHOD=BESTSUBSET option.

ALPHA=*value*
specifies the regularization parameter α to use when you specify METHOD=MCP or METHOD=SCAD. If you specify this *method-option*, then the value that you specify is used to define the MCP or SCAD method, and the MAXALPHA=, MINALPHA=, and MAXITERALPHA= *method-options* are ignored. If you do not specify the ALPHA= *method-option* together with METHOD=MCP or METHOD=SCAD, then PROC REGSELECT searches for the suitable value of the α parameter according to the values of the MAXALPHA=, MINALPHA=, and MAXITERALPHA= *method-options*.

B
displays estimated regression coefficients for each model in the SubsetSelectionSummary table. This *method-option* is available only for the METHOD=BESTSUBSET option.

BEST=*number*
is used with the best-subset selection method. If you specify the SELECT=CP or SELECT=ADJRSQ *method-option*, the BEST= *method-option* specifies the maximum number of subset models to display. For the SELECT=RSQUARE *method-option*, the BEST= *method-option* requests the maximum number of subset models for each size. A small value of this *method-option* greatly reduces the computation time required for large problems.

BIC
displays the Bayesian information criterion (BIC) statistic for each model in the SubsetSelectionSummary table. This *method-option* is available only for the METHOD=BESTSUBSET option.

BIGM=*value*
specifies the constant $\mathcal{M}$ when you specify SOLVER=MILP. For more information about the constant $\mathcal{M}$, see the section "Folded Concave Penalized Selection" on page 89. This *method-option* is available only for the METHOD=MCP or METHOD=SCAD option.

CHOOSE=*criterion*

chooses from the list of models (at each step of the selection process) the model that yields the best value of the specified criterion. If the optimal value of the specified criterion occurs for models at more than one step, then the model that has the smallest number of parameters is chosen. If you omit the CHOOSE= *method-option*, then the selected model is the model at the final step in the selection process. The criteria that are supported depend on the type of model that is being fit. For the supported values of *criterion*, see the chapters for the relevant procedures.

COEFTOL=*value*

specifies the tolerance for the truncation of the estimated coefficients of parameters. In other words, if $|\beta_j| <$ COEFTOL for some j, then β_j would be set to zero; that is, the jth effect is not selected in the final model. This *method-option* is available only for the METHOD=MCP or METHOD=SCAD option.

COMPETITIVE

is applicable as a *method-option* only when METHOD=STEPWISE and the SELECT= criterion is not SL. If you specify the COMPETITIVE *method-option*, then the SELECT= criterion is evaluated for all models in which an effect currently in the model is dropped or an effect not yet in the model is added. The effect whose removal from or addition to the model yields the maximum improvement to the SELECT= criterion is dropped or added.

CRITERION=*criterion*

is an alias for the SELECT= *method-option*.

ENSCALE

scales the results of the elastic net method to offset bias from the double shrinkage inherent in this method (Zou and Hastie 2005). This *method-option* applies only when you specify METHOD=ELASTICNET. The default is not to rescale the solution; this is the so-called naive elastic net.

FAST

implements the computational algorithm of Lawless and Singhal (1978) to compute a first-order approximation to the remaining slope estimates for each subsequent elimination of a variable from the model. When applied in backward selection, this *method-option* essentially leads to approximating the selection process as the selection process of a linear regression model in which the crossproducts matrix equals the Hessian matrix in the full model under consideration. This *method-option* is available only when METHOD=BACKWARD is specified. It is computationally efficient because the model is not fit after the removal of each effect.

FCONV=*value*

specifies a relative function difference convergence criterion. This *method-option* is available only when the solver is either the alternating direction method of multipliers (ADMM) or the limited-memory BFGS method of elastic net selection.

GCONV=*value*

specifies a relative gradient convergence criterion. This *method-option* is available only when the solver is the limited-memory BFGS method of elastic net selection.

GMSEP
displays the GMSEP statistic (estimated mean square error of prediction, assuming that both independent and dependent variables are multivariate normal (Stein 1960; Darlington 1968)) for each model in the SubsetSelectionSummary table. This *method-option* is available only for the METHOD=BESTSUBSET option.

INTTOL=*value*
specifies the amount by which an integer variable value can differ from an integer and still be considered integer-feasible, when you specify SOLVER=MILP. The value of the INTTOL= option can be any number between 1E–9 and 0.5, inclusive. By default, INTTOL=1E–7. This *method-option* is available only for the METHOD=MCP or METHOD=SCAD option.

JP
displays the J_p statistic (estimated mean square error of prediction, assuming that the values of the regressors are fixed and that the model is correct) for each model in the SubsetSelectionSummary table. This *method-option* is available only for the METHOD=BESTSUBSET option.

LAMBDA=*value(s)*
specifies the regularization parameter λ to use when you specify METHOD=ELASTICNET, METHOD=MCP, or METHOD=SCAD. Depending on the selection method, this *method-option* has the following behaviors:

ELASTICNET — If you specify a single value, elastic net selection fits a single model. If you supply a list of values, elastic net selection fits multiple models and selects the best model among the candidates. If you specify neither this *method-option* nor the NLAMBDA= *method-option*, elastic net selection fits a model by using a regularization parameter that is a fraction of $\lambda_{\max}$. In PROC REGSELECT, the value is applicable only when you specify the SOLVER= *method-option* for elastic net selection.

MCP, SCAD — If you specify a single value, then the value that you specify is used to define the MCP or SCAD method, and the MAXLAMBDA=, MINLAMBDA=, MAXITERLAMBDA=, and LAMBDAGRID= *method-options* are ignored. If you do not specify the LAMBDA= *method-option* together with METHOD=MCP or METHOD=SCAD, then the procedure searches for a suitable value of the λ parameter according to the values of the MAXLAMBDA=, MINLAMBDA=, MAXITERLAMBDA=, and LAMBDAGRID= *method-options*.

LAMBDAGRID=LINSPACE | LOGSPACE
specifies the grid pattern for searching the regularization parameter λ when you specify METHOD=MCP or METHOD=SCAD. You can specify the following values:

LINSPACE — requests an evenly spaced grid search of λ in the range $[\lambda_{\min}, \lambda_{\max}]$, where $\lambda_{\min}$ and $\lambda_{\max}$ are specified by the MINLAMBDA= and MAXLAMBDA= *method-options*, respectively.

LOGSPACE — requests a log-scale grid search of λ in the range $[\lambda_{\min}, \lambda_{\max}]$, where $\lambda_{\min}$ and $\lambda_{\max}$ are specified by the MINLAMBDA= and MAXLAMBDA= *method-options*, respectively.

This *method-option* is available only for the METHOD=MCP or METHOD=SCAD option. By default, LAMBDAGRID=LOGSPACE. For more information about this option, see the section "Folded Concave Penalized Selection" on page 89.

LSCOEFFS
requests a hybrid version of the LAR and LASSO methods, in which the sequence of models is determined by the LAR or LASSO algorithm but the coefficients of the parameters for the model at any step are determined by using ordinary least squares.

L2=*value*
specifies the ridge regularization parameter to use when you specify METHOD=ELASTICNET. If you specify this *method-option*, then the value that you specify is used to define the elastic net method, and the L2HIGH=, L2LOW=, and L2STEPS= *method-options* are ignored. If you do not specify the L2= *method-option* together with METHOD=ELASTICNET, then PROC REGSELECT searches for the suitable value of the L2 parameter according to the values of the L2HIGH=, L2LOW=, and L2STEPS= *method-options*.

L2HIGH=*value*
specifies the highest value to use in the search for the suitable value of the ridge regression parameter L2 when you specify METHOD=ELASTICNET. If you specify the L2= *method-option*, then the L2HIGH= *method-option* is ignored. By default, L2HIGH=1.

L2LOW=*value*
specifies the lowest value to use in the search for the suitable value of the ridge regression parameter L2 when you specify METHOD=ELASTICNET. If you specify the L2= *method-option*, then the L2LOW= *method-option* is ignored. By default, L2LOW=0.

L2STEPS=*n*
specifies the number of steps in the search for the suitable value of the ridge regression parameter L2 when you specify METHOD=ELASTICNET. If you specify the L2= *method-option*, then the L2STEPS= *method-option* is ignored. By default, L2STEPS=50.

MAXALPHA=*value*
specifies the highest value to use in the search for the suitable value of the regularization parameter α when you specify METHOD=MCP or METHOD=SCAD. If you specify the ALPHA= *method-option*, then the MAXALPHA= *method-option* is ignored. By default, MAXALPHA=4.7 for METHOD=MCP and MAXALPHA=5.7 for METHOD=SCAD.

MAXEFFECTS=*n*
specifies the maximum number of effects in any model that is considered during the selection process. This *method-option* is ignored when you specify METHOD=BACKWARD. If, at some step of the selection process, the model contains the specified maximum number of effects, then no candidates for addition are considered.

MAXITERALPHA=*n*
specifies the number of steps in the search for the suitable value of the regularization parameter α when you specify METHOD=MCP or METHOD=SCAD. The grid of parameter α is evenly spaced between the values of the MINALPHA= and MAXALPHA= *method-options*. If you specify the ALPHA= *method-option*, then the MAXITERALPHA= *method-option* is ignored. By default, MAXITERALPHA=4.

MAXITERLAMBDA=*n*

specifies the number of steps in the search for the suitable value of the regularization parameter λ when you specify METHOD=MCP or METHOD=SCAD. If you specify the LAMBDA= *method-option*, then the MAXITERLAMBDA= *method-option* is ignored. By default, MAXITERLAMBDA=10.

MAXLAMBDA=*value*

specifies the highest value to use in the search for the suitable value of the regularization parameter λ when you specify METHOD=MCP or METHOD=SCAD. If you specify the LAMBDA= *method-option*, then the MAXLAMBDA= *method-option* is ignored. You can find the default value of the MAXLAMBDA= *method-option* in the section "Folded Concave Penalized Selection" on page 89.

MAXSTEPS=*n*

specifies the maximum number of selection steps to perform. The default value of *n* is the number of effects in the MODEL statement when METHOD=FORWARD, METHOD=BACKWARD, or METHOD=LAR. The default is three times the number of effects when METHOD=STEPWISE or METHOD=LASSO.

MAXTIME=*value*

specifies the upper limit of time (in seconds) for performing the optimization process, in each α and λ step. By default, MAXTIME=600. This *method-option* is available only for the METHOD=MCP or METHOD=SCAD option.

MINALPHA=*value*

specifies the lowest value to use in the search for the suitable value of the regularization parameter α when you specify METHOD=MCP or METHOD=SCAD. If you specify the ALPHA= *method-option*, then the MINALPHA= *method-option* is ignored. By default, MINALPHA=1.7 for METHOD=MCP and MINALPHA=2.7 for METHOD=SCAD.

MINEFFECTS=*n*

specifies the minimum number of effects in any model that is considered during backward selection. This *method-option* is ignored unless METHOD=BACKWARD is specified. The backward selection process terminates if, at some step of the selection process, the model contains the specified minimum number of effects.

MINLAMBDA=*value*

specifies the lowest value to use in the search for the suitable value of the regularization parameter λ when you specify METHOD=MCP or METHOD=SCAD. If you specify the LAMBDA= *method-option*, then the MAXLAMBDA= *method-option* is ignored. You can find the default value of the MINLAMBDA= *method-option* in the section "Folded Concave Penalized Selection" on page 89.

MIXING=*value*

specifies the mixing parameter that controls the balance between the L_1 penalty and the L_2 penalty in elastic net selection. The specified value must be between 0 and 1, inclusive. If you specify 0, elastic net selection fits ridge regression models. If you specify 1, elastic net selection fits LASSO regression models. The default value is 0.5. In PROC REGSELECT, this *method-option* is available only when you specify the SOLVER= *method-option* for elastic net selection.

MSE
displays the mean square error for each model in the SubsetSelectionSummary table. This *method-option* is available only for the METHOD=BESTSUBSET option.

NLAMBDA=*n*
specifies the number of regularization parameters for fitting candidate models. The default value is 1. This option is ignored if you supply a list of regularization parameters by using the LAMBDA= *method-option*. In PROC REGSELECT, this *method-option* is available only when you specify the SOLVER= *method-option* for elastic net selection.

PC
displays the PC statistic (Amemiya's prediction criterion (Amemiya 1976; Judge et al. 1980)) for each model in the SubsetSelectionSummary table. This *method-option* is available only for the METHOD=BESTSUBSET option.

RELAXED < (KAPPA=*value(s)***) >**
specifies relaxed LASSO selection, in which each candidate is determined by the LASSO algorithm but the coefficients of the parameters for each candidate at any step are determined by using a linear combination of LASSO coefficients and least squares estimates. You can use the KAPPA= suboption to control the balance of LASSO coefficients and ordinary least squares in finding the relaxed LASSO solution. By default, the value of the KAPPA= suboption is 0 to 1 by increments of 0.1. The RELAXED *method-option* is available only for the REGSELECT procedure.

RHO=*value*
specifies the scaling factor ρ when the NLAMBDA= *method-option* value n_λ is greater than 1. If the LAMBDA= *method-option* sequence values are omitted, elastic net selection fits candidate models by using the following sequence of the regularization parameters: $\lambda_{\max}, \lambda_{\max}\rho, \ldots, \lambda_{\max}\rho^{n_\lambda - 1}$. If you omit the RHO= *method-option*, elastic net selection uses $\rho = 10^{-4/(n_\lambda - 1)}$ when the number of observations is greater than or equal to the number of parameters, or $\rho = 10^{-2/(n_\lambda - 1)}$ when the number of observations is less than the number of parameters. In PROC REGSELECT, this *method-option* is available only when you specify the SOLVER= *method-option* for elastic net selection.

RMSE
displays the root mean square error for each model in the SubsetSelectionSummary table. This *method-option* is available only for the METHOD=BESTSUBSET option.

SBC
displays the SBC statistic for each model in the SubsetSelectionSummary table. This *method-option* is available only for the METHOD=BESTSUBSET option.

SCALE=TRUE | FALSE
specifies whether the design matrix is scaled (TRUE) or not scaled (FALSE) during model selection. This *method-option* is available only for the METHOD=MCP or METHOD=SCAD option. By default, SCALE=TRUE; this is the same as other penalized methods, such as LASSO and ELASTICNET.

SELECT=SL | *criterion*
specifies the criterion that the procedure uses to determine the order in which effects enter or leave at each step of the selection method. For each step, the effect whose addition to or removal from the current model yields the maximum improvement in the specified criterion is selected. You can use the traditional significance-level approach by specifying the SL criterion; for other supported criteria, see the chapter for the relevant procedure. This *method-option* is not valid when METHOD=LAR, METHOD=LASSO, METHOD=ELASTICNET, or METHOD=GROUPLASSO.

SIGMA=*value*
specifies the true standard deviation of the error term to use in computing the CP and BIC statistics. If you omit this *method-option*, an estimate from the full model is used. This *method-option* is available only for the METHOD=BESTSUBSET option.

SLENTRY=*value*

SLE=*value*
specifies the significance level for entry when STOP=SL or SELECT=SL. By default, SLENTRY=0.05.

SLSTAY=*value*

SLS=*value*
specifies the significance level for staying in the model when STOP=SL or SELECT=SL. By default, SLSTAY=0.05.

SOLVER=*solver-value*
specifies the optimization solver when you specify METHOD=ELASTICNET, METHOD=MCP, or METHOD=SCAD. You can specify the following *solver-values*:

ADMM specifies the alternating direction methods of multipliers (ADMM) as the solver. This *solver-value* is applicable only in PROC REGSELECT when the selection method is elastic net.

BFGS specifies the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method as the solver. This *solver-value* is applicable only when the selection method is elastic net.

LBFGS specifies the limited-memory BFGS method as the solver. This *solver-value* is applicable only when the selection method is elastic net.

MILP specifies the mixed integer linear programming (MILP) solver. This *solver-value* is applicable only in PROC REGSELECT when the selection method is either MCP or SCAD.

NLP specifies the nonlinear programming (NLP) solver.

For the MCP and SCAD selection methods, the default *solver-value* is MILP. For more information about this option for both MCP and SCAD selection, see the section "Folded Concave Penalized Selection" on page 89. For more information about this option for elastic net selection, see the section "Elastic Net Selection" on page 83.

SP
displays the SP statistic (Hocking 1976) for each model in the SubsetSelectionSummary table. This *method-option* is available only for the METHOD=BESTSUBSET option.

SSE
displays the error sum of squares for each model in the SubsetSelectionSummary table. This *method-option* is available only for the METHOD=BESTSUBSET option.

STOP=SL | NONE | *criterion*
specifies a criterion that is used to stop the selection process. The criteria that are supported depend on the type of model that is being fit. For information about the supported criteria, see the chapter for the relevant procedure.

If you omit this *method-option* but specify the SELECT= *method-option*, then the criterion specified in the SELECT= *method-option* is also used as the criterion for the STOP= *method-option*.

You can specify the following values:

NONE stops the selection process if no suitable add or drop candidates can be found or if a size-based limit is reached. For example, if you specify STOP=NONE MAXEFFECTS=5, then the selection process stops at the first step that produces a model that has five effects.

SL stops the selection process at the step where the significance level of the candidate for entry is greater than the SLENTRY= *method-option* value for addition steps when METHOD=FORWARD or METHOD=STEPWISE and where the significance level of the candidate for removal is greater than the SLSTAY= *method-option* value when METHOD=BACKWARD or METHOD=STEPWISE.

criterion stops the selection process if the selection process produces a local extremum of this *criterion* or if a size-based limit is reached. For example, if you specify STOP=AIC MAXSTEPS=5, then the selection process stops before step 5 if the sequence of models has a local minimum of the AIC criterion before step 5. The determination of whether a local minimum is reached is made on the basis of a stop horizon. The default stop horizon is 3, but you can change it by using the STOPHORIZON= option. If the stop horizon is *n* and the STOP= *criterion* at any step is better than the stop criterion at the next *n* steps, then the selection process terminates.

In addition, you can also specify the following *options*:

DETAILS=NONE | SUMMARY | ALL

DETAILS=STEPS< (CANDIDATES(ALL | *n*)) >
specifies the level of detail to be produced about the selection process. The default is DETAILS=SUMMARY.

The DETAILS=ALL and DETAILS=STEPS options produce the following output:

- tables that provide information about the model that is selected at each step of the selection process.

- entry and removal statistics for inclusion or exclusion candidates at each step. By default, only the top 10 candidates at each step are shown. If you specify STEPS(CANDIDATES(*n*)), then the best *n* candidates are shown. If you specify STEPS(CANDIDATES(ALL)), then all candidates are shown.
- a selection summary table that shows by step the effect that is added to or removed from the model in addition to the values of the SELECT, STOP, and CHOOSE criteria for the resulting model.
- a stop reason table that describes why the selection process stopped.
- a selection reason table that describes why the selected model was chosen.
- a selected effects table that lists the effects that are in the selected model.

The DETAILS=SUMMARY option produces only the selection summary, stop reason, selection reason, and selected effects tables. In addition, for METHOD=ELASTICNET with neither a specified L2 value nor a solver specification, the DETAILS=SUMMARY option also displays an elastic net summary table, which shows the ridge regularization parameter L2 (in the ratio scaling) and CHOOSE criterion values in each L2 search step.

HIERARCHY=NONE | SINGLE | SINGLECLASS

specifies whether and how to apply the model hierarchy requirement. You can specify that only classification effects, or both classification and continuous effects, be subject to the hierarchy requirement. This option is ignored unless you also specify one of the following options: METHOD=FORWARD, METHOD=BACKWARD, or METHOD=STEPWISE.

Model hierarchy refers to the requirement that, for any term to be in the model, all model effects that are contained in the term must be present in the model. For example, in order for the interaction A*B to enter the model, the main effects A and B must be in the model. Likewise, neither effect A nor effect B can leave the model while the interaction A*B is in the model.

You can specify the following values:

NONE specifies that model hierarchy not be maintained. Any single effect can enter or leave the model at any step of the selection process.

SINGLE specifies that only one effect enter or leave the model at one time, subject to the model hierarchy requirement. For example, suppose that the model contains the main effects A and B and the interaction A*B. In the first step of the selection process, either A or B can enter the model. In the second step, the other main effect can enter the model. The interaction effect can enter the model only when both main effects have already entered. Also, before A or B can be removed from the model, the A*B interaction must first be removed. All effects (CLASS and interval) are subject to the hierarchy requirement.

SINGLECLASS is the same as HIERARCHY=SINGLE except that only CLASS effects are subject to the hierarchy requirement.

By default, HIERARCHY=NONE.

ORDERSELECT
specifies that effects in the selected model are displayed in the order in which they first entered the model. If you do not specify the ORDERSELECT option, then effects in the selected model are displayed in the order in which they appear in the MODEL statement.

PLOT < (*global-plot-options*) > < = *plot-request* < (*options*) > >

PLOTS < (*global-plot-options*) > < = (*plot-request* < (*options*) > < . . . *plot-request* < (*options*) > >) >
controls the selection process plots that are produced through ODS Graphics.

When you specify only one plot request, you can omit the parentheses around it. Here are some examples:

```
plots=all
plots=coefficients(unpack)
plots(unpack)=(coefficients criteria)
```

ODS Graphics must be enabled before you can request plots. For example:

```
ods graphics on;

proc regselect;
   model y = x1-x100;
   selection method=forward plots=all;
run;

ods graphics off;
```

Global Plot Options

Each *global-plot-option* applies to all plots that the SELECTION statement generates, unless you alter the option by using a specific plot option that follows a *plot-request*. You can specify the following *global-plot-options*:

ENDSTEP=*n*
terminates the step ranges shown on the horizontal axes of plots at the specified step. By default, the step range that is shown terminates at the final step of the selection process.

LOGP | LOGPVALUE
uses a logarithmic axis to display entry and removal significance levels.

MAXPARMLABEL=*n*
specifies the maximum number of characters beyond which all parameter labels on the coefficient progression panel and the coefficient progression plot are suppressed. By default, MAXPARMLABEL=20.

MAXSTEPLABEL=*n*
specifies the maximum number of characters of effect labels to display on the horizontal axes of the plots. By default, MAXSTEPLABEL=20. If the length of any effect label exceeds *n*, then step numbers are used on the horizontal axes of the plots.

STARTSTEP=*n*
specifies that the step ranges shown on the horizontal axes of plots start at the specified step. By default, the step range that is shown starts at the initial step of the selection process.

STEPAXIS=EFFECT | NORMB | NUMBER
specifies the horizontal axis to be used on the plots, where this axis represents the sequence of entering or departing effects. You can specify the following values:

EFFECT requests that each step be labeled by a prefix followed by the name of the effect that enters or leaves at that step. The prefix consists of the step number, followed by a "+" or a "-" sign, depending on whether the effect enters (+) or leaves (-) at that step.

NORMB requests that the horizontal axis value at step *i* be the L1 norm of the parameters at step *i*, normalized by the L1 norm of the parameters at the final step. STEPAXIS=NORMB is valid only with the LAR, LASSO, and elastic net selection methods.

NUMBER requests that each step be labeled by the step number.

UNPACK | UNPACKPANEL
suppresses paneling. By default, multiple plots can appear in the coefficient progression panel and the criteria panel. Specify UNPACK to see each plot individually. You can also specify UNPACK as a suboption of the CRITERIA and COEFFICIENTS options.

Specific Plot Options

You can specify the following *plot-requests* and their *options*:

ALL
produces all relevant plots.

COEFFICIENTS | COEFFICIENTPANEL < (UNPACK | UNPACKPANEL) >
creates a panel of two plots. The upper plot shows the progression of the parameter values as the selection process proceeds. The lower plot shows the progression of the CHOOSE= criterion. If no CHOOSE= criterion is in effect, then only the coefficient progression is shown. If you specify the UNPACK option, then the coefficient progression and the CHOOSE= criterion progression are shown in separate plots.

CRITERIA | CRITERIONPANEL < (UNPACK | UNPACKPANEL) >
creates a panel of model fit criteria. If you specify the UNPACK option, then each criterion progression is shown in a separate plot.

FITBYROLE
plots the progression of a fit statistic on the training, test, and validation data. The fit statistic that is displayed depends on the type of model that is being fit. The fit by role plot is not produced if you do not specify a PARTITION statement.

NONE
suppress all selection process plots, even if you specify other plot options.

SELECTION=*method* **<(***method-options***)>**
is an alias for the METHOD= option.

STOPHORIZON=*n*
specifies the number of consecutive steps at which the STOP= criterion must worsen in order for a local extremum to be detected. For example, suppose that STOP=AIC and the sequence of AIC values at steps 1 to 6 of a selection are 10, 7, 4, 6, 5, 2. If STOPHORIZON=2, then the AIC criterion is deemed to have a local minimum at step 3 because the AIC value at the next two steps are greater than the value 4 that occurs at step 3. However, if STOPHORIZON=3, then the value at step 3 is not deemed to be a local minimum because the AIC value at step 6 is lower than the AIC value at step 3. If you specify STOP=NONE then the stop horizon value is ignored. If you specify STOP=SL, then *n* is ignored and STOPHORIZON=1 is used. By default, STOPHORIZON=3 unless otherwise stated in individual procedure chapters.

STORE Statement

STORE <OUT=>*libref.data-table* **;**

SAVESTATE <OUT=>*libref.data-table* **;**

This section applies to the following procedures: BART, GAMMOD, GAMSELECT, GENSELECT, KCLUS, LMIXED, LOGSELECT, QTRSELECT, REGSELECT, and TREESPLIT.

The STORE or SAVESTATE statement saves the context and results of the analysis. The resulting analytic store has a binary file format that cannot be modified. An example of using analytic store technology is to perform a time-consuming analysis and to store its results by using the STORE statement. Later you can perform specific analysis tasks based on the saved results of the previous analysis, without having to fit the model again. The contents of the analytic store can be processed by the ASTORE procedure (see Chapter 4, "ASTORE Procedure" (*SAS Viya: Machine Learning Procedures*)) and the **`astore`** action (see Chapter 4, "Analytic Store Scoring Action Set" (*SAS Viya: Machine Learning Programming Guide*)). There might also be other actions in your action set that use the **`restore`** parameter to perform other tasks on your analytic store.

Optimization Options

This section applies to the following procedures: GENSELECT, LOGSELECT, NLMOD, and PHSELECT.

This section describes options that are typically available in the PROC statement of the procedures in this book that perform optimizations. The following notation is used to describe the options. $\boldsymbol{\beta}$ denotes the $p \times 1$ vector of parameters for the optimization and β_i is its ith element. The objective function being minimized, its $p \times 1$ gradient vector, and its $p \times p$ Hessian matrix are denoted as $f(\boldsymbol{\beta})$, $\mathrm{g}(\boldsymbol{\beta})$, and $\mathbf{H}(\boldsymbol{\beta})$, respectively. The gradient with respect to the ith parameter is denoted as $g_i(\boldsymbol{\beta})$. Superscripts in parentheses denote the iteration count; for example, $f(\boldsymbol{\beta})^{(k)}$ is the value of the objective function at iteration k.

ABSCONV=*r*

ABSTOL=*r*

specifies an absolute function convergence criterion. For minimization, termination requires $f(\boldsymbol{\beta}^{(k)}) \leq r$, where $\boldsymbol{\beta}$ is the vector of parameters in the optimization and $f(\cdot)$ is the objective function. The default value of *r* is the negative square root of the largest double-precision value, which serves only as a protection against overflows.

ABSFCONV=*r* **<** *n* **>**

ABSFTOL=*r* **<** *n* **>**

specifies an absolute function difference convergence criterion. For all techniques except NMSIMP, termination requires a small change of the function value in successive iterations:

$$|f(\boldsymbol{\beta}^{(k-1)}) - f(\boldsymbol{\beta}^{(k)})| \leq r$$

Here, $\boldsymbol{\beta}$ is the vector of parameters in the optimization and $f(\cdot)$ is the objective function. The same formula is used for the NMSIMP technique, but $\boldsymbol{\beta}^{(k)}$ is defined as the vertex that has the lowest function value and $\boldsymbol{\beta}^{(k-1)}$ is defined as the vertex that has the highest function value in the simplex. PROC NLMOD provides an optional integer value *n*, which specifies the number of successive iterations for which the criterion must be satisfied before the process can be terminated. By default, ABSFCONV=0.

ABSGCONV=*r* **<** *n* **>**

ABSGTOL=*r* **<** *n* **>**

specifies an absolute gradient convergence criterion. Termination requires the maximum absolute gradient element to be small:

$$\max_j |g_j(\boldsymbol{\beta}^{(k)})| \leq r$$

Here, $\boldsymbol{\beta}$ is the vector of parameters in the optimization and $g_j(\cdot)$ is the gradient of the objective function with respect to the jth parameter. This criterion is not used by the NMSIMP technique. PROC NLMOD provides an optional integer value *n*, which specifies the number of successive iterations for which the criterion must be satisfied before the process can be terminated. By default, ABSGCONV=1E–5.

ABSXCONV=*r* < *n* >

ABSXTOL=*r* < *n* >

specifies an absolute parameter convergence criterion: For all techniques except NMSIMP, termination requires a small Euclidean distance between successive parameter vectors,

$$\| \boldsymbol{\beta}^{(k)} - \boldsymbol{\beta}^{(k-1)} \|_2 \leq r$$

For the NMSIMP technique, termination requires either a small length $\alpha^{(k)}$ of the vertices of a restart simplex,

$$\alpha^{(k)} \leq r$$

or a small simplex size,

$$\delta^{(k)} \leq r$$

where the simplex size $\delta^{(k)}$ is defined as the L1 distance from the simplex vertex $\boldsymbol{\xi}^{(k)}$ that has the smallest function value to the other p simplex points $\boldsymbol{\beta}_l^{(k)} \neq \boldsymbol{\xi}^{(k)}$:

$$\delta^{(k)} = \sum_{\boldsymbol{\beta}_l \neq y} \| \boldsymbol{\beta}_l^{(k)} - \boldsymbol{\xi}^{(k)} \|_1$$

PROC NLMOD provides an optional integer value *n*, which specifies the number of successive iterations for which the criterion must be satisfied before the process can terminate. The default is r = 1E–8 for the NMSIMP technique and $r = 0$ otherwise.

CORRECTIONS=*m*

specifies the number of the stored quasi-Newton update histories, which is referred to as the number of corrections, for the LBFGS technique. The m value is usually set as small as 3. In general, larger values improve convergence speed and solution quality for the LBFGS technique. However, for many problems, after reaching a certain threshold, the improvement can stall and the memory usage and solver's per-iteration computation cost can increase. In practice, you can set high values for small or medium problems and small values for large problems. By default, CORRECTIONS=20.

FCONV=*r* < *n* >

FTOL=*r* < *n* >

specifies a relative function difference convergence criterion. For all techniques except NMSIMP, termination requires a small relative change of the function value in successive iterations,

$$\frac{|f(\boldsymbol{\beta}^{(k)}) - f(\boldsymbol{\beta}^{(k-1)})|}{|f(\boldsymbol{\beta}^{(k-1)})|} \leq r$$

Here, $\boldsymbol{\beta}$ denotes the vector of parameters that participate in the optimization, and $f(\cdot)$ is the objective function. The same formula is used for the NMSIMP technique, but $\boldsymbol{\beta}^{(k)}$ is defined as the vertex that has the lowest function value and $\boldsymbol{\beta}^{(k-1)}$ is defined as the vertex that has the highest function value in the simplex.

PROC NLMOD provides an optional integer value *n*, which specifies the number of successive iterations for which the criterion must be satisfied before the process can terminate. The default value is r=$2 \times \epsilon$ where ϵ is the machine precision, which is the smallest double-precision floating-point number such that $1 + \epsilon > 1$.

FCONV2=*r* **<** *n* **>**

FTOL2=*r* **<** *n* **>**

specifies a second function convergence criterion. For all techniques except NMSIMP, termination requires a small predicted reduction of the objective function:

$$df^{(k)} \approx f(\boldsymbol{\beta}^{(k)}) - f(\boldsymbol{\beta}^{(k)} + \mathrm{s}^{(k)})$$

The predicted reduction

$$\begin{aligned} df^{(k)} &= -\mathbf{g}^{(k)'}\mathrm{s}^{(k)} - \frac{1}{2}\mathrm{s}^{(k)'}\mathbf{H}^{(k)}\mathrm{s}^{(k)} \\ &= -\frac{1}{2}\mathrm{s}^{(k)'}\mathbf{g}^{(k)} \leq r \end{aligned}$$

is computed by approximating the objective function f by the first two terms of the Taylor series and substituting the Newton step,

$$\mathrm{s}^{(k)} = -[\mathbf{H}^{(k)}]^{-1}\mathbf{g}^{(k)}$$

For the NMSIMP technique, termination requires a small standard deviation of the function values of the $p+1$ simplex vertices $\boldsymbol{\beta}_l^{(k)}$, $l = 0, \ldots, p$,

$$\sqrt{\frac{1}{n+1}\sum_l \left[f(\boldsymbol{\beta}_l^{(k)}) - \overline{f}(\boldsymbol{\beta}^{(k)})\right]^2} \leq r$$

where $\overline{f}(\boldsymbol{\beta}^{(k)}) = \frac{1}{p+1}\sum_l f(\boldsymbol{\beta}_l^{(k)})$. If there are p_{act} boundary constraints active at $\boldsymbol{\beta}^{(k)}$, the mean and standard deviation are computed only for the $n + 1 - p_{act}$ unconstrained vertices.

PROC NLMOD provides an optional integer value *n*, which specifies the number of successive iterations for which the criterion must be satisfied before the process can terminate. The default value is *r* = 1E–6 for the NMSIMP technique and *r*= 0 otherwise.

GCONV=*r* **<** *n* **>**

GTOL=*r* **<** *n* **>**

specifies a relative gradient convergence criterion. For all techniques except CONGRA and NMSIMP, termination requires that the normalized predicted function reduction be small:

$$\frac{\mathrm{g}(\boldsymbol{\beta}^{(k)})'[\mathbf{H}^{(k)}]^{-1}\mathrm{g}(\boldsymbol{\beta}^{(k)})}{|f(\boldsymbol{\beta}^{(k)})|} \leq r$$

Here, $\boldsymbol{\beta}$ denotes the vector of parameters that participate in the optimization, $f(\cdot)$ is the objective function, and $\mathrm{g}(\cdot)$ is the gradient. For the CONGRA technique (where a reliable Hessian estimate $\mathbf{H}$ is not available), the following criterion is used:

$$\frac{\| \mathrm{g}(\boldsymbol{\beta}^{(k)}) \|_2^2 \quad \| \mathrm{s}(\boldsymbol{\beta}^{(k)}) \|_2}{\| \mathrm{g}(\boldsymbol{\beta}^{(k)}) - \mathrm{g}(\boldsymbol{\beta}^{(k-1)}) \|_2 \, |f(\boldsymbol{\beta}^{(k)})|} \leq r$$

This criterion is not used by the NMSIMP technique. PROC NLMOD provides an optional integer value *n*, which specifies the number of successive iterations for which the criterion must be satisfied before the process can terminate. By default, GCONV=1E–8.

GCONV2=*r* **<** *n* **>**

GTOL2=*r* **<** *n* **>**

specifies another relative gradient convergence criterion. For the TRUREG, LEVMAR, NRRIDG, and NEWRAP techniques, the following criterion of Browne (1982) is used:

$$\max_j \frac{|\mathbf{g}_j(\boldsymbol{\beta}^{(k)})|}{\sqrt{f(\boldsymbol{\beta}^{(k)})\mathbf{H}^{(k)}_{j,j}}} \leq r$$

This criterion is not used by the other techniques.

PROC NLMOD provides an optional integer value *n*, which specifies the number of successive iterations for which the criterion must be satisfied before the process can terminate. By default, GCONV2=0.

MAXFUNC=*n*

MAXFU=*n*

specifies the maximum number *n* of function calls in the optimization process. The default values are as follows, depending on the optimization technique:

- TRUREG, NRRIDG, and NEWRAP: 125
- QUANEW and DBLDOG: 500
- CONGRA: 1,000
- LBFGS: 2,000
- NMSIMP: 3,000

The optimization can terminate only after completing a full iteration. Therefore, the number of function calls that are actually performed can exceed the number that is specified by this option. You can specify the optimization technique in the TECHNIQUE= option.

MAXITER=*n*

MAXIT=*n*

specifies the maximum number *n* of iterations in the optimization process. The default values are as follows, depending on the optimization technique:

- TRUREG, NRRIDG, and NEWRAP: 50
- QUANEW and DBLDOG: 200
- CONGRA: 400
- LBFGS and NMSIMP: 1,000

These default values also apply when *n* is specified as a missing value. You can specify the optimization technique in the TECHNIQUE= option.

MAXTIME=*r*

specifies an upper limit of *r* seconds of CPU time for the optimization process. The time specified by *r* is checked only once at the end of each iteration. Therefore, the actual running time can be longer than *r*. The default value is the largest floating-point double representation of your computer.

MINITER=*n*

MINIT=*n*

specifies the minimum number of iterations. If you request more iterations than are actually needed for convergence to a stationary point, the optimization algorithms can behave strangely. For example, the effect of rounding errors can prevent the algorithm from continuing for the required number of iterations. By default, MINITER=0.

NORMALIZE=YES | NO

specifies whether the objective function should be normalized during the optimization by the reciprocal of the used frequency count. This option affects the values reported in the "Iteration History" table. The results reported in the "Fit Statistics" are always displayed for the nonnormalized log-likelihood function. The default is to normalize the objective function.

TECHNIQUE=*technique*

TECH=*technique*

specifies the optimization technique for obtaining maximum likelihood estimates. You can specify one of the following *techniques*:

CONGRA	performs a conjugate-gradient optimization.
DBLDOG	performs a version of double-dogleg optimization.
LBFGS	performs a limited-memory BFGS optimization.
LEVMAR	performs a Levenberg-Marquardt nonlinear least-squares minimization. This technique is available only with PROC NLMOD.
NEWRAP	performs a Newton-Raphson optimization with line search.
NMSIMP	performs a Nelder-Mead simplex optimization.
NONE	performs no optimization.
NRRIDG	performs a Newton-Raphson optimization with ridging.
QUANEW	performs a dual quasi-Newton optimization.
TRUREG	performs a trust-region optimization

By default, TECHNIQUE=NRRIDG.

For more information, see the section "Choosing an Optimization Algorithm" on page 103.

XCONV=*r*

XTOL=*r*

specifies the relative parameter convergence criterion. Convergence requires a small relative parameter change in subsequent iterations,

$$\max_j |\delta_j^{(i)}| < r$$

where

$$\delta_j^{(i)} = \begin{cases} \beta_j^{(i)} - \beta_j^{(i-1)} & |\beta_j^{(i-1)}| < 0.01 \\ \frac{\beta_j^{(i)} - \beta_j^{(i-1)}}{\beta_j^{(i-1)}} & \text{otherwise} \end{cases}$$

and $\beta_j^{(i)}$ is the estimate of the jth parameter at iteration i. For the NMSIMP technique, the same formula is used, but $\beta_j^{(k)}$ is defined as the vertex that has the lowest function value and $\beta_j^{(k-1)}$ is defined as the vertex that has the highest function value in the simplex. The default value is r = 1E–8 for the NMSIMP technique and $r = 0$ otherwise.

Details for SAS Visual Statistics Procedures

Levelization of Classification Variables

This section applies to the following procedures: GAMMOD, GAMSELECT, GENSELECT, LMIXED, LOGSELECT, MODELMATRIX, NLMOD, PHSELECT, PLSMOD, QTRSELECT, REGSELECT, SANDWICH, and TREESPLIT.

A classification variable enters the statistical analysis or model not through its values but through its levels. The process of associating values of a variable with levels is called *levelization*.

During the process of levelization, observations that share the same value are assigned to the same level. The manner in which values are grouped can be affected by the inclusion of formats. The sort order of the levels can be determined by specifying the ORDER= option in the procedure statement. In procedures in this book, you can also control the sorting order separately for each variable in the CLASS statement.

Consider the data on nine observations in Table 2.12. The variable A is integer-valued, and the variable X is a continuous variable that has a missing value for the fourth observation. The fourth and fifth columns of Table 2.12 apply two different formats to the variable X.

Table 2.12 Example Data for Levelization

Obs	A	x	FORMAT x 3.0	FORMAT x 3.1
1	2	1.09	1	1.1
2	2	1.13	1	1.1
3	2	1.27	1	1.3
4	3	.	.	.
5	3	2.26	2	2.3
6	3	2.48	2	2.5
7	4	3.34	3	3.3
8	4	3.34	3	3.3
9	4	3.14	3	3.1

By default, levelization of the variables groups the observations by the formatted value of the variable, except for numerical variables for which no explicit format is provided. Numerical variables for which no explicit format is provided are sorted by their internal value. The levelization of the four columns in Table 2.12 leads to the level assignment in Table 2.13.

Table 2.13 Values and Levels

	A		X		FORMAT x 3.0		FORMAT x 3.1	
Obs	**Value**	**Level**	**Value**	**Level**	**Value**	**Level**	**Value**	**Level**
1	2	1	1.09	1	1	1	1.1	1
2	2	1	1.13	2	1	1	1.1	1
3	2	1	1.27	3	1	1	1.3	2
4	3	2	.	.	.	.	.	.
5	3	2	2.26	4	2	2	2.3	3
6	3	2	2.48	5	2	2	2.5	4
7	4	3	3.34	7	3	3	3.3	6
8	4	3	3.34	7	3	3	3.3	6
9	4	3	3.14	6	3	3	3.1	5

The sort order for the levels of CLASS variables can be specified in the ORDER= option in the CLASS statement.

When ORDER=FORMATTED (which is the default) is in effect for numeric variables for which you have supplied no explicit format, the levels are ordered by their internal values. To order numeric classification levels that have no explicit format by their BEST12. formatted values, you can specify the BEST12. format explicitly for the CLASS variables.

Table 2.14 shows how values of the ORDER= option are interpreted.

Table 2.14 Interpretation of Values of ORDER= Option

Value of ORDER=	Levels Sorted By
FORMATTED	External formatted value, except for numeric variables that have no explicit format, which are sorted by their unformatted (internal) value. The sort order is machine-dependent.
FREQ	Descending frequency count (levels that have the most observations come first in the order)
INTERNAL	Unformatted value. The sort order is machine-dependent.

For more information about sort order, see the chapter about the SORT procedure in the *Base SAS Procedures Guide* and the discussion of BY-group processing in the "Grouping Data" section of *SAS Programmers Guide: Essentials*.

When the MISSING option is specified in the CLASS statement, the missing values ('.' for a numeric variable and blanks for a character variable) are included in the levelization and are assigned a level. Table 2.15

displays the results of levelizing the values in Table 2.12 when the MISSING option is in effect.

Table 2.15 Values and Levels When the MISSING Option Is Specified

	A		X		FORMAT x 3.0		FORMAT x 3.1	
Obs	Value	Level	Value	Level	Value	Level	Value	Level
1	2	1	1.09	2	1	2	1.1	2
2	2	1	1.13	3	1	2	1.1	2
3	2	1	1.27	4	1	2	1.3	3
4	3	2	.	1	.	1	.	1
5	3	2	2.26	5	2	3	2.3	4
6	3	2	2.48	6	2	3	2.5	5
7	4	3	3.34	8	3	4	3.3	7
8	4	3	3.34	8	3	4	3.3	7
9	4	3	3.14	7	3	4	3.1	6

When the MISSING option is not specified, it is important to understand the implications of missing values for your statistical analysis. When a procedure in this book levelizes the CLASS variables, an observation for which any CLASS variable has a missing value is excluded from the analysis.

Procedures in this book print a "Number of Observations" table that shows the number of observations that are read from the data set and the number of observations that are used in the analysis. Pay careful attention to this table—especially when your data table contains missing values—to ensure that no observations are unintentionally excluded from the analysis.

Specification and Parameterization of Model Effects

This section applies to the following procedures: GAMMOD, GAMSELECT, GENSELECT, LMIXED, LOGSELECT, MODELMATRIX, NLMOD, PHSELECT, PLSMOD, QTRSELECT, REGSELECT, and SANDWICH.

Procedures in this book that have a MODEL statement support the formation of effects. An *effect* is an element in a linear model structure that is formed from one or more variables. At some point the statistical representations of these models involve linear structures such as

$$\mathbf{X}\boldsymbol{\beta}$$

or

$$\mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\boldsymbol{\gamma}$$

The model matrices $\mathbf{X}$ and $\mathbf{Z}$ are formed according to effect-construction rules.

Procedures that also have a CLASS statement support the rich set of effects that is discussed in this section.

Procedures that also have an EFFECT statement enable you to construct special *constructed* effects that are discussed in "EFFECT Statement" on page 18.

In order to correctly interpret the results from a statistical analysis, you need to understand how construction (*parameterization*) rules apply to regression-type models, whether these are linear models as in the REGSELECT procedure or generalized linear models as in the LOGSELECT and GENSELECT procedures.

Effects are specified by a special notation that uses variable names and operators. There are two types of variables: classification (or CLASS) variables and continuous variables. *Classification variables* can be either numeric or character and are specified in a CLASS statement. For more information, see the section "Levelization of Classification Variables" on page 62. An independent variable that is not declared in the CLASS statement is assumed to be *continuous*. For example, the heights and weights of subjects are continuous variables.

Two primary operators (crossing and nesting) are used for combining the variables, and several additional operators are used to simplify effect specification. Operators are discussed in the section "Effect Operators" on page 65.

Procedures in this book that have a CLASS statement support a general linear model (GLM) parameterization and might also support nonsingular parameterizations for the classification variables. The GLM parameterization, commonly called *dummy* parameterization, is the default for all procedures in this book. For more information, see the sections "GLM Parameterization of Classification Variables and Effects" on page 67 and "Nonsingular Parameterization" on page 71.

Effect Operators

Table 2.16 summarizes the operators that are available for selecting and constructing effects. These operators are discussed in the following sections.

Table 2.16 Available Effect Operators

Operator	Example	Description
Interaction	A*B	Crosses the levels of the effects
Nesting	A(B)	Nests A levels within B levels
Bar operator	A | B | C	Specifies all interactions
At sign operator	A | B | C@2	Reduces interactions in bar effects
Dash operator	A1-A10	Specifies sequentially numbered variables
Colon operator	A:	Specifies variables with common prefix
Double dash operator	A- -C	Specifies sequential variables in data set order

Bar and At Sign Operators

You can shorten the specification of a large factorial model by using the bar operator. For example, two ways of writing the model for a full three-way factorial model follow:

```
model Y = A B C   A*B A*C B*C   A*B*C;

model Y = A|B|C;
```

When the bar (|) is used, the right and left sides become effects, and the cross of them becomes an effect. Multiple bars are permitted. The expressions are expanded from left to right, using rules 2–4 in Searle (1971, p. 390).

- Multiple bars are evaluated from left to right. For example, A | B | C is evaluated as follows:

$$\begin{aligned} \text{A | B | C} &\rightarrow \{\text{ A | B }\}\text{ | C} \\ &\rightarrow \{\text{ A B A*B }\}\text{ | C} \\ &\rightarrow \text{A B A*B C A*C B*C A*B*C} \end{aligned}$$

- Crossed and nested groups of variables are combined. For example, A(B) | C(D) generates A*C(B D), among other terms.
- Duplicate variables are removed. For example, A(C) | B(C) generates A*B(C C), among other terms, and the extra C is removed.
- Effects are discarded if a variable occurs on both the crossed and nested parts of an effect. For example, A(B) | B(D E) generates A*B(B D E), but this effect is eliminated immediately.

You can also specify the maximum number of variables involved in any effect that results from bar evaluation by specifying that maximum number, preceded by an at sign (@), at the end of the bar effect. For example, the following specification selects only those effects that contain two or fewer variables:

```
model Y = A|B|C@2;
```

The preceding example is equivalent to specifying the following MODEL statement:

```
model Y = A B C   A*B A*C B*C;
```

More examples of using the bar and at operators follow:

A | C(B)	is equivalent to	A C(B) A*C(B)
A(B) | C(B)	is equivalent to	A(B) C(B) A*C(B)
A(B) | B(D E)	is equivalent to	A(B) B(D E)
A | B(A) | C	is equivalent to	A B(A) C A*C B*C(A)
A | B(A) | C@2	is equivalent to	A B(A) C A*C
A | B | C | D@2	is equivalent to	A B A*B C A*C B*C D A*D B*D C*D
A*B(C*D)	is equivalent to	A*B(C D)

Colon, Dash, and Double Dash Operators

You can simplify the specification of a large model when some of your variables have a common prefix by using the colon (:) operator and the dash (-) operator. The dash operator enables you to list variables that are numbered sequentially, and the colon operator selects all variables with a given prefix. For example, if your data set contains the variables X1 through X9, the following MODEL statements are equivalent:

```
model Y = X1 X2 X3 X4 X5 X6 X7 X8 X9;

model Y = X1-X9;

model Y = X:;
```

If your data set contains only the three covariates X1, X2, and X9, then the colon operator selects all three variables:

```
model Y = X:;
```

However, the following specification returns an error because X3 through X8 are not in the data set:

```
model Y = X1-X9;
```

The double dash (--) operator enables you to select variables that are stored sequentially in the SAS data set, whether or not they have a common prefix. You can use the CONTENTS procedure (see *Base SAS Procedures Guide*) to determine your variable ordering. For example, if you replace the dash in the preceding MODEL statement with a double dash, as follows, then all three variables are selected:

```
model Y = X1--X9;
```

If your data set contains the variables A, B, and C, then you can use the double dash operator to select these variables by specifying the following:

```
model Y = A--C;
```

GLM Parameterization of Classification Variables and Effects

Table 2.17 shows the types of effects that are available in procedures in this book; they are discussed in more detail in the following subsections. Let A, B, and C represent classification variables, and let X and Z represent continuous variables.

Table 2.17 Available Types of Effects

Effect	Example	Description
Intercept	Default	Intercept (unless the NOINT option is specified)
Regression	X Z	Continuous variables
Polynomial	X*Z	Interaction of continuous variables
Main	A B	CLASS variables
Interaction	A*B	Crossing of CLASS variables
Nested	A(B)	Main effect A nested within CLASS effect B
Continuous-by-class	X*A	Crossing of continuous and CLASS variables
Continuous-nesting-class	X(A)	Continuous variable X1 nested within CLASS variable A
General	X*Z*A(B)	Combinations of different types of effects

Table 2.18 shows some examples of MODEL statements that use various types of effects.

Table 2.18 MODEL Statement Effect Examples

Specification	Type of Model
`model Y=X;`	Simple regression
`model Y=X Z;`	Multiple regression
`model Y=X X*X;`	Polynomial regression
`model Y=A;`	One-way analysis of variance (ANOVA)
`model Y=A B C;`	Main-effects ANOVA
`model Y=A B A*B;`	Factorial ANOVA with interaction
`model y=A B(A) C(B A);`	Nested ANOVA

Table 2.18 *continued*

Specification	Type of Model
`model Y=A X;`	Analysis of covariance (ANCOVA)
`model Y=A X(A);`	Separate-slopes regression
`model Y=A X X*A;`	Homogeneity-of-slopes regression

Intercept

By default, linear models that are created by procedures in this book automatically include a column of 1s in **X**. This column corresponds to an intercept parameter. In many procedures, you can use the NOINT option in the MODEL statement to suppress this intercept. For example, the NOINT option is useful when the MODEL statement contains a classification effect and you want the parameter estimates to be in terms of the mean response for each level of that effect.

Regression Effects

Numeric variables or polynomial terms that involve them can be included in the model as regression effects (covariates). The actual values of such terms are included as columns of the relevant model matrices. You can use the bar operator along with a regression effect to generate polynomial effects. For example, X | X | X expands to X X*X X*X*X, which is a cubic model.

Main Effects

If a classification variable has m levels, the GLM parameterization generates m columns for its main effect in the model matrix. Each column is an indicator variable for a particular level. The order of the columns is the sort order of the values of their levels and can be controlled by the ORDER= option in the CLASS statement.

Table 2.19 is an example where β_0 denotes the intercept and A and B are classification variables that have two and three levels, respectively.

Table 2.19 Example of Main Effects

Data		I	A		B		
A	B	β_0	A1	A2	B1	B2	B3
1	1	1	1	0	1	0	0
1	2	1	1	0	0	1	0
1	3	1	1	0	0	0	1
2	1	1	0	1	1	0	0
2	2	1	0	1	0	1	0
2	3	1	0	1	0	0	1

There are usually more columns for these effects than there are degrees of freedom to estimate them. In other words, the GLM parameterization of main effects is *singular*.

Interaction Effects

Often a model includes interaction (crossed) effects to account for how the effect of a variable changes along with the values of other variables. With an interaction, the terms are first reordered to correspond to the order of the variables in the CLASS statement. Thus, B*A becomes A*B if A precedes B in the CLASS statement. Then, the GLM parameterization generates columns for all combinations of levels that occur in the data. The order of the columns is such that the rightmost variables in the interaction change faster than the leftmost variables (Table 2.20).

Table 2.20 Example of Interaction Effects

Data		I	A		B			A*B					
A	B	β_0	A1	A2	B1	B2	B3	A1B1	A1B2	A1B3	A2B1	A2B2	A2B3
1	1	1	1	0	1	0	0	1	0	0	0	0	0
1	2	1	1	0	0	1	0	0	1	0	0	0	0
1	3	1	1	0	0	0	1	0	0	1	0	0	0
2	1	1	0	1	1	0	0	0	0	0	1	0	0
2	2	1	0	1	0	1	0	0	0	0	0	1	0
2	3	1	0	1	0	0	1	0	0	0	0	0	1

In the preceding matrix, main-effects columns are not linearly independent of crossed-effects columns. In fact, the column space for the crossed effects contains the space of the main effect.

When your model contains many interaction effects, you might be able to code them more parsimoniously by using the bar operator (|). The bar operator generates all possible interaction effects. For example, A | B | C expands to A B A*B C A*C B*C A*B*C. To eliminate higher-order interaction effects, use the at sign (@) in conjunction with the bar operator. For example, A | B | C | D@2 expands to A B A*B C A*C B*C D A*D B*D C*D.

Nested Effects

Nested effects are generated in the same manner as crossed effects. Hence, the design columns that are generated by the following two statements are the same (but the ordering of the columns is different):

```
model Y=A B(A);

model Y=A A*B;
```

The nesting operator in procedures in this book is more of a notational convenience than an operation that is distinct from crossing. Nested effects are typically characterized by the property that the nested variables do not appear as main effects. The order of the variables within nesting parentheses is made to correspond to the order of these variables in the CLASS statement. The order of the columns is such that variables outside the parentheses index faster than those inside the parentheses, and the rightmost nested variables index faster than the leftmost variables (Table 2.21).

Table 2.21 Example of Nested Effects

Data		I	A		B(A)					
A	B	β_0	A1	A2	B1A1	B2A1	B3A1	B1A2	B2A2	B3A2
1	1	1	1	0	1	0	0	0	0	0

Table 2.21 *continued*

Data		I	A		B(A)				
1	2	1	1	0	0	1	0	0	0
1	3	1	1	0	0	0	1	0	0
2	1	1	0	1	0	0	0	1	0
2	2	1	0	1	0	0	0	0	1
2	3	1	0	1	0	0	0	0	1

Continuous-Nesting-Class Effects

When a continuous variable nests or crosses with a classification variable, the design columns are constructed by multiplying the continuous values into the design columns for the classification effect (Table 2.22).

Table 2.22 Example of Continuous-Nesting-Class Effects

Data		I	A		X(A)	
X	A	β_0	A1	A2	X(A1)	X(A2)
21	1	1	1	0	21	0
24	1	1	1	0	24	0
22	1	1	1	0	22	0
28	2	1	0	1	0	28
19	2	1	0	1	0	19
23	2	1	0	1	0	23

This model estimates a separate intercept and a separate slope for X within each level of A.

Continuous-by-Class Effects

Continuous-by-class effects generate the same design columns as continuous-nesting-class effects. Table 2.23 shows the construction of the X*A effect. The two columns for this effect are the same as the columns for the X(A) effect in Table 2.22.

Table 2.23 Example of Continuous-by-Class Effects

Data		I	X	A		X*A	
X	A	β_0	X	A1	A2	X*A1	X*A2
21	1	1	21	1	0	21	0
24	1	1	24	1	0	24	0
22	1	1	22	1	0	22	0
28	2	1	28	0	1	0	28
19	2	1	19	0	1	0	19
23	2	1	23	0	1	0	23

You can use continuous-by-class effects together with pure continuous effects to test for homogeneity of slopes.

General Effects

An example that combines all the effects is X1*X2*A*B*C(D E). The continuous list comes first, followed by the crossed list, followed by the nested list in parentheses.

Effects might be renamed by the procedure to correspond to ordering rules. For example, B*A(E D) might be renamed A*B(D E) to satisfy the following:

- Classification variables that occur outside parentheses (crossed effects) are sorted in the order in which they appear in the CLASS statement.
- Variables within parentheses (nested effects) are sorted in the order in which they appear in the CLASS statement.

The sequencing of the parameters that are generated by an effect is determined by the variables whose levels are indexed faster:

- Variables in the crossed list index faster than variables in the nested list.
- Within a crossed or nested list, variables to the right index faster than variables to the left.

For example, suppose that a model includes four effects—A, B, C, and D—each having two levels, 1 and 2, and that the CLASS statement is as follows:

```
class A B C D;
```

Then the order of the parameters for the effect B*A(C D), which is renamed A*B(C D), is as follows:

$$
\begin{array}{llll}
A_1B_1C_1D_1 \rightarrow & A_1B_2C_1D_1 \rightarrow & A_2B_1C_1D_1 \rightarrow & A_2B_2C_1D_1 \rightarrow \\
A_1B_1C_1D_2 \rightarrow & A_1B_2C_1D_2 \rightarrow & A_2B_1C_1D_2 \rightarrow & A_2B_2C_1D_2 \rightarrow \\
A_1B_1C_2D_1 \rightarrow & A_1B_2C_2D_1 \rightarrow & A_2B_1C_2D_1 \rightarrow & A_2B_2C_2D_1 \rightarrow \\
A_1B_1C_2D_2 \rightarrow & A_1B_2C_2D_2 \rightarrow & A_2B_1C_2D_2 \rightarrow & A_2B_2C_2D_2
\end{array}
$$

Note that first the crossed effects B and A are sorted in the order in which they appear in the CLASS statement so that A precedes B in the parameter list. Then, for each combination of the nested effects in turn, combinations of A and B appear. The B effect changes fastest because it is rightmost in the cross list. Then A changes next fastest, and D changes next fastest. The C effect changes most slowly because it is leftmost in the nested list.

Nonsingular Parameterization

When a parameterization of main effects provides the same number of columns for the effects as there are degrees of freedom to estimate them, the parameterization is called *nonsingular*. A variety of nonsingular parameterizations for classification effects are available for many procedures in this book. In most of these procedures you use the PARAM= option in the CLASS statement to specify the parameterization.

Consider a model with one CLASS variable A that has four levels, 1, 2, 5, and 7. Details of the possible choices for the PARAM= option follow.

REFERENCE Three columns are created to indicate group membership of the nonreference levels. For the reference level, all three dummy variables have a value of 0. For example, if the reference level is 7 (REF=7), the design matrix columns for A are as follows.

Reference Coding			
	Design Matrix		
A	A1	A2	A5
1	1	0	0
2	0	1	0
5	0	0	1
7	0	0	0

Parameter estimates of CLASS main effects that use the reference coding scheme estimate the difference in the effect of each nonreference level compared to the effect of the reference level.

EFFECT Three columns are created to indicate group membership of the nonreference levels. For the reference level, all three dummy variables have a value of –1. For example, if the reference level is 7 (REF=7), the design matrix columns for A are as follows.

Effect Coding			
	Design Matrix		
A	A1	A2	A5
1	1	0	0
2	0	1	0
5	0	0	1
7	–1	–1	–1

Parameter estimates of CLASS main effects that use the effect coding scheme estimate the difference in the effect of each nonreference level compared to the average effect over all four levels.

ORDINAL | THERMOMETER Three columns are created to indicate group membership of the higher levels of the effect. For the first level of the effect (which for A is 1), all three dummy variables have a value of 0. The design matrix columns for A are as follows.

Ordinal Coding			
	Design Matrix		
A	A2	A5	A7
1	0	0	0
2	1	0	0
5	1	1	0
7	1	1	1

The first level of the effect is a control or baseline level. Parameter estimates of CLASS main effects, using the ORDINAL coding scheme, estimate the differences

between effects of successive levels. When the parameters have the same sign, the effect is monotonic across the levels.

POLYNOMIAL | POLY Three columns are created. The first represents the linear term (x), the second represents the quadratic term (x^2), and the third represents the cubic term (x^3), where x is the level value. If the CLASS levels are not numeric, they are translated into 1, 2, 3, ... according to their sort order. The design matrix columns for A are as follows.

Polynomial Coding			
	Design Matrix		
A	APOLY1	APOLY2	APOLY3
1	1	1	1
2	2	4	8
5	5	25	125
7	7	49	343

ORTHEFFECT The columns are obtained by applying the Gram-Schmidt orthogonalization to the columns for PARAM=EFFECT. The design matrix columns for A are as follows.

Orthogonal Effect Coding			
	Design Matrix		
A	AOEFF1	AOEFF2	AOEFF3
1	1.41421	–0.81650	–0.57735
2	0	1.63299	–0.57735
5	0	0	1.73205
7	–1.41421	–0.81649	–0.57735

ORTHORDINAL | ORTHOTHERM The columns are obtained by applying the Gram-Schmidt orthogonalization to the columns for PARAM=ORDINAL. The design matrix columns for A are as follows.

Orthogonal Ordinal Coding			
	Design Matrix		
A	AOORD1	AOORD2	AOORD3
1	–1.73205	0	0
2	0.57735	–1.63299	0
5	0.57735	0.81650	–1.41421
7	0.57735	0.81650	1.41421

ORTHPOLY The columns are obtained by applying the Gram-Schmidt orthogonalization to the columns for PARAM=POLY. The design matrix columns for A are as follows.

Orthogonal Polynomial Coding			
	Design Matrix		
A	AOPOLY1	AOPOLY2	AOPOLY5
1	–1.15311	0.90712	–0.92058
2	–0.73380	–0.54041	1.47292
5	0.52414	–1.37034	–0.92058
7	1.36277	1.00363	0.36823

ORTHREF

The columns are obtained by applying the Gram-Schmidt orthogonalization to the columns for PARAM=REFERENCE. The design matrix columns for A are as follows.

Orthogonal Reference Coding			
	Design Matrix		
A	AOREF1	AOREF2	AOREF3
1	1.73205	0	0
2	–0.57735	1.63299	0
5	–0.57735	–0.81650	1.41421
7	–0.57735	–0.81650	–1.41421

Class Variable Parameterization with Unbalanced Designs

Procedures in this book initially parameterize the CLASS variables by looking at the levels of the variables across the complete data set. If you have an *unbalanced* replication of levels across variables or BY groups, then the design matrix and the parameter interpretation might be different from what you expect. For example, suppose you have a model that has one CLASS variable A with three levels (1, 2, and 3), and another CLASS variable B with two levels (1 and 2). If the third level of A occurs only with the first level of B, if you use the EFFECT parameterization, and if your model contains the effect A(B) and an intercept, then the design for A within the second level of B is not a differential effect. In particular, the design looks like the following:

		Design Matrix			
		A(B=1)		A(B=2)	
B	A	A1	A2	A1	A2
1	1	1	0	0	0
1	2	0	1	0	0
1	3	–1	–1	0	0
2	1	0	0	1	0
2	2	0	0	0	1

Procedures in this book detect linear dependency among the last two design variables and set the parameter for A2(B=2) to 0, resulting in an interpretation of these parameters as if they were reference- or dummy-coded. The REFERENCE or GLM parameterization might be more appropriate for such problems.

Model Selection Methods

The model selection methods are specified in the METHOD= option in the SELECTION statement. The following methods are available, although specific procedures might support only a subset of these methods. Furthermore, the examples in this section refer to fit criteria that might not be supported by a specific procedure.

Full Model Fitted

When METHOD=NONE, the complete model that is specified in the MODEL statement is used to fit the model, and no effect selection is done.

Forward Selection

This section applies to the following procedures: GENSELECT, LOGSELECT, PHSELECT, QTRSELECT, and REGSELECT.

METHOD=FORWARD specifies the forward selection technique, which begins with just the intercept and then sequentially adds the effect that most improves the fit. The process terminates when no significant improvement can be obtained by adding any effect.

In the traditional implementation of forward selection, the statistic that is used to determine whether to add an effect is the significance level of a hypothesis test that reflects an effect's contribution to the model if it is included. At each step, the effect that is most significant is added. The process stops when the significance level for adding any effect is greater than some specified entry significance level.

An alternative approach to address the critical problem of when to stop the selection process is to assess the quality of the models that are produced by the forward selection method and choose the model from this sequence that "best" balances goodness of fit against model complexity. You can use several criteria for this purpose.

It is important to keep in mind that forward selection bases the decision about what effect to add at any step by considering models that differ by one effect from the current model. This search paradigm cannot guarantee reaching a "best" subset model. Furthermore, the add decision is greedy in the sense that the effect that is deemed most significant is the effect that is added. However, if your goal is to find a model that is best in terms of some selection criterion other than the significance level of the entering effect, then even this one step choice might not be optimal. For example, the effect that you would add to get a model that has the smallest value of the Mallows' $C(p)$ statistic at the next step is not necessarily the same effect that is most significant based on a hypothesis test. You can specify the criterion to optimize at each step by using the SELECT= option. For example, the following statement requests that at each step the effect that is added be the one that produces a model that has the smallest value of the Mallows' $C(p)$ statistic:

```
selection method=forward(select=CP);
```

When all effects are variables (that is, effects have one degree of freedom and no hierarchy), using ADJRSQ, AIC, AICC, BIC, CP, RSQUARE, or SBC as the selection criterion for forward selection produces the same sequence of additions. However, if the degrees of freedom contributed by different effects are not constant

or if an out-of-sample prediction-based criterion is used, then different sequences of additions might be obtained.

If you specify only the SELECT= criterion, then this criterion is also used to decide when to stop the selection process. In the previous example, not only do effects enter based on the Mallows' $C(p)$ statistic, but the selection terminates when the $C(p)$ statistic has a local minimum.

You use the CHOOSE= option to specify the criterion for selecting one model from the sequence of models produced. If you do not specify a CHOOSE= criterion, then the model at the final step is the selected model.

For example, if you specify the following statement, then forward selection terminates at the step where no effect can be added at the 0.2 significance level:

```
selection method=forward(select=SL choose=AIC SLE=0.2);
```

However, the selected model is the first one that has the minimum value of Akaike's information criterion. In some cases, this minimum value might occur at a step much earlier than the final step. In other cases, the AIC might start increasing only if more steps are performed—that is, a larger value is used for the significance level for entry. If you want to minimize AIC, then too many steps are performed in the former case and too few in the latter case. To address this issue, you can use the STOP= option to specify a stopping criterion. When you specify a stopping criterion, forward selection continues until a local extremum of the stopping criterion in the sequence of models generated is reached. To be deemed a local extremum, a criterion value at a particular step must be better than its value at the next n steps, where n is known as the "stop horizon." By default, the stop horizon is three steps, but you can change this by specifying the STOPHORIZON= option.

For example, if you specify the following statement, then forward selection terminates at the step where the effect to be added at the next step would produce a model that has an AIC statistic larger than the AIC statistic of the current model:

```
selection method=forward(select=SL stop=AIC) stophorizon=1;
```

In most cases, provided that the entry significance level is large enough that the local extremum of the named criterion occurs before the final step, specifying either of the following statements selects the same model, but more steps are done in the first case:

```
selection method=forward(select=SL choose=CRITERION);
selection method=forward(select=SL stop=CRITERION);
```

In some cases, there might be a better local extremum that cannot be reached if you specify the STOP= option but can be found if you use the CHOOSE= option. Also, you can use the CHOOSE= option in preference to the STOP= option if you want to examine how the named criterion behaves as you move beyond the step where the first local minimum of this criterion occurs.

You can specify both the CHOOSE= and STOP= options. You can also use these options together with options that specify size-based limits on the selected model. You might want to consider models that are generated by forward selection and have at most some fixed number of effects, but select from within this set based on a criterion that you specify. For example, specifying the following statements requests that forward selection continue until there are 20 effects in the final model and chooses among the sequence of models the one that has the largest value of the adjusted R-square statistic:

```
selection method=forward(stop=none maxeffects=20 choose=ADJRSQ);
```

You can also combine these options to select a model where one of two conditions is met. For example, the following statement chooses whatever occurs first between a local minimum of the sum of squares on validation data and a local minimum of the corrected Akaike's information criterion (AICC):

```
selection method=forward(stop=AICC choose=VALIDATE);
```

You can find discussion and references to studies about criteria for variable selection in Burnham and Anderson (2002), along with some cautions and recommendations.

Examples of Forward Selection Specifications

The following statement adds effects that at each step produce the lowest value of the SBC statistic and stops at the step where adding any effect would increase the SBC statistic:

```
selection method=forward stophorizon=1;
```

The following statement adds effects based on significance level and stops when all candidate effects for entry at a step have a significance level greater than the default entry significance level of 0.05:

```
selection=forward(select=SL);
```

The following statement adds effects based on significance level and stops at a step where adding any effect increases the error sum of squares computed on the validation data:

```
selection=forward(select=SL stop=validation) stophorizon=1;
```

The following statement adds effects that at each step produce the lowest value of the AIC statistic and stops at the first step whose AIC value is smaller than the AIC value at the next three steps:

```
selection=forward(select=AIC);
```

The following statement adds effects that at each step produce the largest value of the adjusted R-square statistic and stops at the step where the significance level that corresponds to the addition of this effect is greater than 0.2:

```
selection=forward(select=ADJRSQ stop=SL SLE=0.2);
```

Backward Elimination

This section applies to the following procedures: GENSELECT, LOGSELECT, PHSELECT, QTRSELECT, and REGSELECT.

METHOD=BACKWARD specifies the backward elimination technique. This technique starts from the full model, which includes all independent effects. Then effects are deleted one by one until a stopping condition is satisfied. At each step, the effect that shows the smallest contribution to the model is deleted.

In the traditional implementation of backward selection, the statistic that is used to determine whether to drop an effect is significance level. At any step, the least significant predictor is dropped and the process continues until all effects that remain in the model are significant at a specified stay significance level (SLS).

Just as with forward selection, you can use the SELECT= option to change the criterion that is used to assess effect contributions. You can also specify a stopping criterion in the STOP= option and use a CHOOSE=

option to provide a criterion for selecting among the sequence of models produced. For more information, see the discussion in the section "Forward Selection" on page 75.

Examples of Backward Selection Specifications

The following statement removes effects that at each step produce the largest value of the Schwarz Bayesian information criterion (SBC) statistic and stops at the step where removing any effect increases the SBC statistic:

```
selection method=backward stophorizon=1;
```

The following statement bases removal of effects on the corrected Akaike's information criterion (AICC) and stops at the first step whose AICC value is smaller than the AICC value at the next three steps:

```
selection method=backward(select=AICC);
```

The following statement bases removal of effects on significance level and stops when all candidate effects for removal at a step are significant at the default stay significance level of 0.05:

```
selection method=backward(select=SL);
```

The following statement bases removal of effects on significance level and stops when all effects in the model are significant at the 0.1 level. Finally, from the sequence of models generated, the chosen model is the one that produces the smallest average square error when scored on the validation data:

```
selection method=backward(select=SL choose=VALIDATE SLS=0.1);
```

The following statement applies in logistic regression models the fast backward technique of Lawless and Singhal (1978), a first-order approximation that has greater numerical efficiency than full backward selection:

```
selection method=backward(fast);
```

The fast technique fits an initial full logistic model and a reduced model after the candidate effects have been dropped. On the other hand, full backward selection fits a logistic regression model each time an effect is removed from the model.

Stepwise Selection

This section applies to the following procedures: GENSELECT, LOGSELECT, PHSELECT, QTRSELECT, and REGSELECT.

METHOD=STEPWISE specifies the stepwise method, which modifies the forward selection technique by allowing effects already in the model to be removed.

In the traditional implementation of stepwise selection method, the same entry and removal significance levels for the forward selection and backward elimination methods are used to assess contributions of effects as they are added to or removed from a model. If, at a step of the stepwise method, any effect in the model is not significant at the SLSTAY= level, then the least significant of these effects is removed from the model and the algorithm proceeds to the next step. This ensures that no effect can be added to a model while some effect currently in the model is not deemed significant. Another effect can be added to the model only after all necessary deletions have been accomplished. In this case the effect whose addition is the most significant

is added to the model and the algorithm proceeds to the next step. The stepwise process ends when none of the effects outside the model is significant at the SLENTRY= level and every effect in the model is significant at the SLSTAY= level. In some cases, neither of these two conditions for stopping is met and the sequence of models cycles. In this case, the stepwise method terminates at the end of the cycle.

Just as you can in forward selection and backward elimination, you can use the SELECT= option to change the criterion that is used to assess effect contributions. You can also use the STOP= option to specify a stopping criterion and use a CHOOSE= option to provide a criterion for selecting among the sequence of models produced. For more information, see the section "Forward Selection" on page 75.

For selection criteria other than the significance level, further modification to the stepwise method is supported. In the standard stepwise method, no effect can enter the model if removing any effect currently in the model would yield an improved value of the selection criterion. In the modification, you can use the COMPETITIVE option to specify that addition and deletion of effects should be treated competitively. The selection criterion is evaluated for all models that are produced by deleting an effect from the current model or by adding an effect to this model. The action that most improves the selection criterion is the action taken.

Examples of Stepwise Selection Specifications

The following statement requests stepwise selection based on the SBC criterion:

```
selection method=stepwise;
```

First, if removing any effect yields a model that has a lower SBC statistic than the current model, then the effect that produces the smallest SBC statistic is removed. If removing any effect increases the SBC statistic, then provided that adding some effect lowers the SBC statistic, the effect that produces the model that has the lowest SBC is added.

The following statement requests the traditional stepwise method:

```
selection=stepwise(select=SL)
```

First, if the removal of any effect in the model is not significant at the default stay level of 0.05, then the least significant effect is removed and the algorithm proceeds to the next step. Otherwise, the effect whose addition is the most significant is added, provided that it is significant at the default entry level of 0.05.

The following statement requests the traditional stepwise method, where effects enter and leave based on significance levels, but with the following extra check: if any effect to be added or removed yields a model whose SBC statistic is greater than the SBC statistic of the current model, then the stepwise method terminates at the current model.

```
selection method=stepwise(select=SL stop=SBC) stophorizon=1;
```

In this case, the entry and stay significance levels still play a role because they determine whether an effect is deleted from or added to the model. This extra check might result in the selection terminating before a local minimum of the SBC criterion is found.

The following statement selects effects to enter or drop as in the previous example except that the significance level for entry is now 0.1 and the significance level to stay is 0.08. From the sequence of models produced, the selected model is chosen to yield the minimum AIC statistic:

```
selection method=stepwise(select=SL SLE=0.1 SLS=0.08 choose=AIC);
```

The following statement requests stepwise selection that is based on the AICC criterion and treats additions and deletions competitively:

```
selection method=stepwise(select=AICC competitive);
```

Each step evaluates the AICC statistics that correspond to the removal of any effect in the current model or the addition of any effect to the current model and chooses the addition or removal that produced the minimum value, provided that this minimum is lower than the AICC statistic of the current model.

The following statement requests stepwise selection that is based on the SBC criterion, treats additions and deletions competitively, and stops based on the average square error over the validation data:

```
selection=stepwise(select=SBC competitive stop=VALIDATE);
```

At any step, SBC statistics that correspond to the removal of any effect from the current model or the addition of any effect to the current model are evaluated. The addition or removal that produces the minimum SBC value is made. The average square error on the validation data for the model with this addition or removal is evaluated. The selection stops when the average square error so produced increases for three consecutive steps.

Forward-Swap Selection

This section applies to the REGSELECT procedure.

METHOD=FORWARDSWAP specifies the forward-swap selection method, which is an extension of the forward selection method. The forward-swap selection method incorporates steps that improve a model by replacing an effect in the model with an effect that is not in the model. When the model selection criterion is R square, this method is the same as the maximum R-square improvement (MAXR) method that is implemented in the REG procedure in SAS/STAT software. You cannot use the effect significance level as the selection criterion for the forward-swap method.

The forward-swap selection method begins by finding the one-effect model that produces the best value of the selection criterion. Then another effect (the one that yields the greatest improvement in the selection criterion) is added. After the two-effect model is obtained, each of the effects in the model is compared to each effect that is not in the model. For each comparison, the forward-swap method determines whether removing one effect and replacing it with the other effect improves the selection criterion. After comparing all possible swaps, the forward-swap method makes the swap that produces the greatest improvement in the selection criterion. Comparisons begin again, and the process continues until the forward-swap method finds that no other swap could improve the selection criterion. Thus, the two-variable model that is produced is considered the “best” two-variable model that the technique can find. Another variable is then added to the model, and the comparing-and-swapping process is repeated to find the “best” three-variable model, and so on.

The difference between the stepwise selection method and the forward-swap selection method is that all swaps are evaluated before any addition is made in the forward-swap method. In the stepwise selection method, the “worst” effect might be removed without considering what adding the “best” remaining effects might accomplish. Because the forward-swap method needs to examine all possible pairwise effect swaps at

each step of the selection process, the forward-swap method is much more computationally expensive than the stepwise selection method; it might not be appropriate for models that contain a large number of effects.

Least Angle Regression

This section applies to the REGSELECT procedure.

METHOD=LAR specifies least angle regression (LAR), which was introduced by Efron et al. (2004). Not only does this algorithm provide a selection method in its own right, but with one additional modification, it can be used to efficiently produce LASSO solutions. Just like the forward selection method, the LAR algorithm produces a sequence of regression models in which one parameter is added at each step, terminating at the full least squares solution when all parameters have entered the model.

The algorithm starts by centering the covariates and response and scaling the covariates so that they all have the same corrected sum of squares. Initially all coefficients are 0, as is the predicted response. The predictor that is most correlated with the current residual is determined, and a step is taken in the direction of this predictor. The length of this step determines the coefficient of this predictor and is chosen so that some other predictor and the current predicted response have the same correlation with the current residual. At this point, the predicted response moves in the direction that is equiangular between these two predictors. Moving in this direction ensures that these two predictors continue to have a common correlation with the current residual. The predicted response moves in this direction until a third predictor has the same correlation with the current residual as the two predictors already in the model. A new direction is determined that is equiangular among these three predictors, and the predicted response moves in this direction until a fourth predictor, which has the same correlation with the current residual, joins the set. This process continues until all predictors are in the model.

As in other selection methods, the issue of when to stop the selection process is crucial. You can use the CHOOSE= option to specify a criterion for choosing among the models at each step. You can also use the STOP= option to specify a stopping criterion. These formulas use the approximation that at step k of the LAR algorithm, the model has k degrees of freedom. For a detailed discussion of this so-called simple approximation, see Efron et al. (2004).

A modification of LAR selection that is suggested in Efron et al. (2004) uses the LAR algorithm to select the set of covariates in the model at any step, but it uses ordinary least squares regression with just these covariates to obtain the regression coefficients. You can request this hybrid method by specifying the LSCOEFFS suboption of METHOD=LAR.

LASSO Selection

This section applies to the following procedures: GENSELECT, LOGSELECT, PHSELECT, and REGSELECT.

METHOD=LASSO specifies the least absolute shrinkage and selection operator (LASSO) method. LASSO arises from a constrained form of ordinary least squares regression where the sum of the absolute values of the regression coefficients is constrained to be smaller than a specified parameter. More precisely let $X = (x_1, x_2, \ldots, x_m)$ denote the matrix of covariates and let y denote the response, where the x_i have been

centered and scaled to have unit standard deviation and mean 0 and y has mean 0. Then for a particular parameter t, the LASSO regression coefficients $\beta = (\beta_1, \beta_2, \ldots, \beta_m)$ are the solution to the following constrained optimization problem:

$$\text{minimize} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 \qquad \text{subject to} \quad \sum_{j=1}^{m} |\beta_j| \leq t$$

Provided that the LASSO parameter t is small enough, some of the regression coefficients are exactly 0. Hence, you can view the LASSO as selecting a subset of the regression coefficients for each LASSO parameter. By increasing the LASSO parameter in discrete steps, you obtain a sequence of regression coefficients in which the nonzero coefficients at each step correspond to selected parameters.

Early implementations (Tibshirani 1996) of LASSO selection used quadratic programming techniques to solve the constrained least squares problem for each LASSO parameter of interest. Later Osborne, Presnell, and Turlach (2000) developed a "homotopy method" that generates the LASSO solutions for all values of t. Efron et al. (2004) derived a variant of their algorithm for least angle regression (called LARS algorithm) that can be used to obtain a sequence of LASSO solutions from which all other LASSO solutions can be obtained by linear interpolation. This algorithm for METHOD=LASSO is used in PROC REGSELECT. It can be viewed as a stepwise procedure with a single addition to or deletion from the set of nonzero regression coefficients at any step.

As in the other selection methods, you can use the CHOOSE= option to specify a criterion to choose among the models at each step of the LASSO algorithm. You can also use the STOP= option to specify a stopping criterion. For more information, see the discussion in the section "Forward Selection" on page 75. The model degrees of freedom used at any step of the LASSO are simply the number of nonzero regression coefficients in the model at that step. Efron et al. (2004) cite empirical evidence for doing this but do not give any mathematical justification for this choice.

A modification of LASSO selection suggested in Efron et al. (2004) uses the LASSO algorithm to select the set of covariates in the model at any step, but it uses ordinary least squares regression and just these covariates to obtain the regression coefficients. You can request this hybrid method by specifying the LSCOEFFS suboption of SELECTION=LASSO.

Adaptive LASSO Selection

This section applies to the REGSELECT procedure.

Adaptive LASSO selection is a modification of LASSO selection. In adaptive LASSO selection, weights are applied to each of the parameters in forming the LASSO constraint (Zou 2006). More precisely, suppose that the response y has mean 0 and the regressors x are scaled to have mean 0 and common standard deviation. Furthermore, suppose that you can find a suitable estimator $\hat{\beta}$ of the parameters in the true model and you define a weight vector by $w = 1/|\hat{\beta}|^{\gamma}$, where $\gamma \geq 0$. Then the adaptive LASSO regression coefficients $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_m)$ are the solution to the following constrained optimization problem:

$$\text{minimize} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 \qquad \text{subject to} \quad \sum_{j=1}^{m} |w_j \beta_j| \leq t$$

The solution to the unconstrained least squares problem is used as the estimator $\hat{\beta}$. This is appropriate unless collinearity is a concern. If the regressors are collinear or nearly collinear, then Zou (2006) suggests using a ridge regression estimate to form the adaptive weights.

Adaptive LASSO enjoys the oracle properties; namely, it performs as well as if the true underlying model were given in advance.

Relaxed LASSO Selection

This section applies to the REGSELECT procedure.

Relaxed LASSO selection (Hastie, Tibshirani, and Tibshirani 2020) is a variation of the original LASSO selection method. Relaxed LASSO selection uses the LASSO algorithm to choose the set of covariates in the model at any step. However, the relaxed LASSO regression coefficients $\boldsymbol{\beta}_{\text{relaxed}}$ are a linear combination of the original LASSO coefficients $\boldsymbol{\beta}_{\text{lasso}}$ and least squares estimates $\boldsymbol{\beta}_{\text{lscoeffs}}$:

$$\boldsymbol{\beta}_{\text{relaxed}} = \kappa \boldsymbol{\beta}_{\text{lasso}} + (1 - \kappa)\boldsymbol{\beta}_{\text{lscoeffs}}$$

where κ is a mixing parameter that controls the balance between the LASSO solution and the ordinary least squares solution. If $\kappa = 1$, relaxed LASSO selection reduces to ordinary LASSO selection; if $\kappa = 0$, relaxed LASSO selection is the same as LASSO selection with the LSCOEFFS suboption.

If you have a good estimate of κ, you can specify a single value in the KAPPA= suboption. If you supply a list of κ values by using the KAPPA= suboption, PROC REGSELECT sorts the values in descending order, computes the model fit criterion by using $\boldsymbol{\beta}_{\text{relaxed}}$, and then selects candidate models. If you do not specify a value for κ, then by default PROC REGSELECT searches for a value from the series $\{0, 0.1, 0.2, \ldots, 0.9, 1\}$ that is optimal according to the current value of the CHOOSE= criterion (by default, CHOOSE=SBC).

Elastic Net Selection

This section applies to the following procedures: GENSELECT, LOGSELECT, and REGSELECT.

The METHOD=ELASTICNET option specifies the elastic net method proposed by Zou and Hastie (2005), which bridges the LASSO method and ridge regression. The elastic net method strikes a balance between having a parsimonious model and borrowing strength from correlated regressors, by solving the least squares regression problem with constraints on both the sum of the absolute coefficients and the sum of the squared coefficients.

More specifically, the elastic net coefficients $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_m)$ are the solution to the constrained optimization problem

$$\min \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 \qquad \text{subject to} \quad \sum_{j=1}^{m} |\beta_j| \le t_1, \ \sum_{j=1}^{m} \beta_j^2 \le t_2$$

This can be written as the equivalent Lagrangian form

$$\min \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda_1 \sum_{j=1}^{m} |\beta_j| + \lambda_2 \sum_{j=1}^{m} \beta_j^2$$

Elastic net can be treated as a convex combination of LASSO and ridge penalty; pure LASSO and pure ridge are two limiting cases. If t_1 is set to a very large value or, equivalently, if λ_1 is set to 0, then the elastic net method reduces to ridge regression. If t_2 is set to a very large value or, equivalently, if λ_2 is set to 0, then the elastic net method reduces to LASSO. If t_1 and t_2 are both large or, equivalently, if λ_1 and λ_2 are both set to 0, then the elastic net method reduces to ordinary least squares regression.

The elastic net method can overcome the limitations of LASSO in the following three scenarios:

- If you have more parameters than observations ($m > n$), the LASSO method selects at most n variables before it saturates, because of the nature of the convex optimization problem. This can be a defect for a variable selection method. By contrast, the elastic net method can select more than n variables in this case because of the ridge regression regularization.
- If there is a group of variables that have high pairwise correlations, then whereas LASSO tends to select only one variable from that group, the elastic net method can select more than one variable.
- If you have more observations than parameters ($n > m$), and there are high correlations between predictors, then it has been empirically observed that the prediction performance of LASSO is dominated by ridge regression. In this case, the elastic net method can achieve better prediction performance by using ridge regression regularization.

The Lagrangian form of the elastic net optimization problem can be reformulated as

$$\min \|\tilde{\mathbf{y}} - \tilde{\mathbf{X}}\boldsymbol{\beta}\|^2 + \lambda_1 \sum_{j=1}^{m} |\beta_j|$$

where the augmented design matrix $\tilde{\mathbf{X}}$ and response $\tilde{\mathbf{y}}$ are defined by

$$\tilde{\mathbf{X}}_{(n+m)\times m} = \begin{bmatrix} \mathbf{X} \\ \sqrt{\lambda_2}\mathbf{I} \end{bmatrix}, \quad \tilde{\mathbf{y}}_{(n+m)\times 1} = \begin{bmatrix} \mathbf{y} \\ \mathbf{0} \end{bmatrix}$$

This implies that for a given λ_2, the coefficients of the elastic net fit follow the same piecewise linear path as LASSO and can be solved using least angle regression (LARS) algorithm. Moreover, Zou and Hastie (2005) suggest rescaling the coefficients by $1 + \lambda_2$ to deal with the double amount of shrinkage in the elastic net fit; such rescaling is applied when you specify the ENSCALE option in the SELECTION statement.

If you have a good estimate of λ_2, you can specify the value in the L2= option. If you do not specify a value for λ_2, then by default PROC REGSELECT searches for a value between 0 and 1 that is optimal according to the current value of the CHOOSE= criterion (by default, CHOOSE=SBC).

Computing the entire solution path can be prohibitive for large problems when m is large. Instead of using the LARS algorithm, you can use other optimization techniques to solve LASSO and elastic net problems

for a reduced set of regularization parameters. You can reformulate the general elastic net objective into the following optimization problem:

$$\min_{\boldsymbol{\beta}} \frac{1}{2n}\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\|^2 + \alpha\lambda\sum_{j=1}^{m}|\beta_j| + \frac{(1-\alpha)\lambda}{2}\sum_{j=1}^{m}\beta_j^2$$

where λ is the regularization parameter and α is the mixing parameter that controls the balance between the LASSO penalty and the ridge penalty. If $\alpha = 1$, the problem reduces to the LASSO regression; if $\alpha = 0$, the problem reduces to the ridge regression. Because the LASSO penalty is nonsmooth, the optimization is not readily solved by traditional techniques that use gradient information. To solve the problem, the REGSELECT procedure takes the following three general approaches:

- Uses the Orthant-Wise limited-memory quasi-Newton (OWL-QN) method (Andrew and Gao 2007), which is based on LBFGS and can efficiently optimize L_1 regularized objective functions. This solver efficiently handles wide data, where $m \gg n$. You can use it by specifying SOLVER=LBFGS.
- Uses the alternating direction method of multipliers (ADMM), which decomposes the objective into smooth and nonsmooth parts and solves efficiently with augmented Lagrangians (Boyd et al. 2011). This solver efficiently handles tall data, where $n \gg m$. You can use it by specifying SOLVER=ADMM. This method is only available for the REGSELECT procedure.
- Turns the nonsmooth LASSO penalty into a smooth penalty by using the reformulation of $\beta = \beta_+ - \beta_-$, and thus $|\beta| = \beta_+ + \beta_-$, where $\beta_+ = \max(\beta, 0), \beta_- = \max(-\beta, 0)$. This reformulation converts the optimization to a constrained nonlinear problem. You can use two solvers by specifying SOLVER=BFGS and SOLVER=NLP. For low-dimensional problems, both solvers can provide accurate solutions.

If you use one of these solvers to optimize the elastic net objective without supplying a list of regularization parameters or specifying the number of regularization parameters, PROC REGSELECT uses a single heuristic value for regularization:

$$\lambda = \begin{cases} 0.001\lambda_{\max} & \text{if } n > m \\ 0.1\lambda_{\max} & \text{otherwise} \end{cases}$$

where $\alpha\lambda_{\max}$ makes the lower bound of the L_1 regularization such that $\boldsymbol{\beta} = \mathbf{0}$. Note that for ridge regression when $\alpha = 0$, $\lambda_{\max}$ does not exist because the ridge penalty is not sparsity-inducing. In this case, elastic net selection computes $\lambda_{\max}$ by assuming that $\alpha = 1$ if you do not supply a list of regularization parameters.

If you supply a list of regularization parameters by using the LAMBDA= suboption, PROC REGSELECT sorts the values in descending order and then selects candidate models. If you supply only the number of regularization parameters n_λ by using the NLAMBDA= suboption, PROC REGSELECT constructs a series of λ values by using the following mechanism:

$$\lambda_k = \lambda_{\max} \cdot \rho^{k-1}, k = 1, \ldots, n_\lambda$$

You can specify ρ by using the RHO= option. Otherwise PROC REGSELECT uses

$$\rho = \begin{cases} 10^{-4/(n_\lambda-1)} & \text{if } n > m \\ 10^{-2/(n_\lambda-1)} & \text{otherwise} \end{cases}$$

The idea of penalization by using both LASSO and ridge penalty also extends to generalized linear models, including logistic regression models. For those models, the general elastic net objective function is formulated as

$$\min_{\boldsymbol{\beta}} -\frac{1}{n} L(\boldsymbol{\mu}(\boldsymbol{\beta}); \mathbf{y}) + \alpha\lambda \sum_{j=1}^{m} |\beta_j| + \frac{(1-\alpha)\lambda}{2} \sum_{j=1}^{m} \beta_j^2$$

where $L(\boldsymbol{\mu}(\boldsymbol{\beta}); \mathbf{y})$ is the log-likelihood function. If you do not supply a list of regularization parameters or do not specify the number of regularization parameters, PROC GENSELECT and PROC LOGSELECT use a default NLAMBDA= suboption value of 20 in order to choose a good candidate in a reasonable amount of time.

Adaptive Elastic Net Selection

This section applies to the REGSELECT procedure.

Adaptive elastic net selection (Zou and Zhang 2009) is an improved version of the elastic net and adaptive LASSO selection methods. Adaptive elastic net penalizes the squared error loss by using a combination of the ℓ_2 penalty and the adaptive ℓ_1 penalty.

More specifically, the adaptive elastic net coefficients $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_m)$ are the solution to the optimization problem

$$\min \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda_1 \sum_{j=1}^{m} |w_j \beta_j| + \lambda_2 \sum_{j=1}^{m} \beta_j^2$$

The adaptively weighted ℓ_1 penalty achieves the oracle property, and the elastic net penalty handles the collinearity. The adaptive weights can be obtained by ridge regression estimation.

Like the naive elastic net, the adaptive elastic net can be transformed into an adaptive LASSO-type problem in some augmented space

$$\min \|\tilde{\mathbf{y}} - \tilde{\mathbf{X}}\boldsymbol{\beta}\|^2 + \lambda_1 \sum_{j=1}^{m} |w_j \beta_j|$$

where the augmented design matrix $\tilde{\mathbf{X}}$ and response $\tilde{\mathbf{y}}$ are defined by

$$\tilde{\mathbf{X}}_{(n+m)\times m} = \begin{bmatrix} \mathbf{X} \\ \sqrt{\lambda_2}\mathbf{I} \end{bmatrix}, \quad \tilde{\mathbf{y}}_{(n+m)\times 1} = \begin{bmatrix} \mathbf{y} \\ \mathbf{0} \end{bmatrix}$$

Group LASSO Selection

This section applies to the following procedures: GENSELECT, LOGSELECT, and PHSELECT.

The group LASSO method, proposed by Yuan and Lin (2006), is a variant of LASSO that is specifically designed for models defined in terms of effects that have multiple degrees of freedom, such as the main effects of CLASS variables and interactions between CLASS variables. If all effects in the model are continuous, then the group LASSO method is the same as the LASSO method.

Recall that LASSO selection depends on solving a constrained optimization problem of the form

$$\min\{-L(\boldsymbol{\mu};\mathbf{y})\} \qquad \text{subject to} \quad \sum_{j=1}^{m} |\beta_j| \le t$$

where L is the log-likelihood function. In this formulation, individual parameters can be included or excluded from the model independently, subject only to the overall constraint. In contrast, the group LASSO method uses a constraint that forces all parameters that correspond to the same effect to be included or excluded simultaneously. For a model that has k effects, let β_{G_j} be the group of linear coefficients that correspond to effect j in the model. Then group LASSO depends on solving a constrained optimization problem of the form

$$\min\{-L(\boldsymbol{\mu};\mathbf{y})\} \qquad \text{subject to} \quad \sum_{j=1}^{k} \sqrt{|G_j|}\|\beta_{G_j}\| \le t$$

where $|G_j|$ is the number of parameters that correspond to effect j, and $\|\beta_{G_j}\|$ denotes the Euclidean norm of the parameters β_{G_j},

$$\|\beta_{G_j}\| = \sqrt{\sum_{i=1}^{G_j} \beta_i^2}$$

That is, instead of constraining the sum of the absolute value of individual parameters, group LASSO constrains the Euclidean norm of groups of parameters, where groups are defined by effects.

You can write the group LASSO method in the equivalent Lagrangian form, which is an example of a penalized log-likelihood function:

$$\min\{-L(\boldsymbol{\mu};\mathbf{y})\} + \lambda \sum_{j=1}^{k} \sqrt{|G_j|}\|\beta_{G_j}\|$$

The weight $\sqrt{|G_j|}$ was suggested by Yuan and Lin (2006) in order to take the size of the group into consideration in group LASSO.

Unlike LASSO for linear models, group LASSO does not allow a piecewise linear constant solution path as generated by a LAR algorithm. Instead, the method proposed by Nesterov (2013) is adopted to solve the Lagrangian form of the group LASSO problem that corresponds to a prespecified regularization parameter λ. Nesterov's method is known to have an optimal convergence rate for first-order black-box optimization. Because the optimal λ is usually unknown, a series of regularization parameters $\rho, \rho^2, \rho^3, \ldots$ is used, where

ρ is a positive value less than 1. You can specify ρ by using the LASSORHO= option in the procedure; the default value is $\rho = 0.8$. In the ith step of group LASSO selection, the value that is used for λ is ρ^i.

A unique feature of the group LASSO method is that it does not necessarily add or remove precisely one effect at each step of the process. This is different from the forward, stepwise, and backward selection methods.

As with the other selection methods, you can specify a criterion to choose among the models at each step of the group LASSO algorithm by using the CHOOSE= option in the SELECTION statement. You can also specify a stopping criterion by using the STOP= option in the SELECTION statement. If you do not specify either the CHOOSE= or STOP= option, the model at the last LASSO step is chosen as the selected model and parameter estimates are reported for this model. These parameter estimates are used to compute predicted values for the output data tables.

For more information, see the discussion in the section "SELECTION Statement" on page 42.

The model degrees of freedom at any step of the LASSO are simply the number of nonzero regression coefficients in the model at that step. Efron et al. (2004) cite empirical evidence for doing this but do not give any mathematical justification for this choice.

Some distributions involve a dispersion parameter (the parameter ϕ in the expressions for the log likelihood). These parameters are not estimated by the LASSO optimization algorithm, and are set to either the default value or a value that you specify. You can use the PHI= option in the procedure to set the dispersion to a fixed value.

Best-Subset Selection

This section applies to the REGSELECT procedure.

The METHOD=BESTSUBSET option specifies the best-subset selection method, which uses the branch-and-bound technique (Furnival and Wilson 1974) to efficiently search for subsets of model effects that best predict the response variable. For a multiple regression model that contains m predictors, there are $2^m - 1$ candidate subset models. When m becomes large, the number of candidate submodels grows exponentially, and searching within the model space becomes impossible. By presenting all possible candidates in a tree diagram where the full model is at the root node and simple one-parameter models are at the leaf nodes, the problem becomes how to efficiently search the tree. When model B consists of a subset of model effects that are in model A a criterion $g(\cdot)$ that satisfies the inequality

$$g(A) \leq g(B)$$

so that better models have smaller value of $g(\cdot)$, enables you to avoid fitting models such as B farther down the branch when the criterion $g(A)$ is already larger than the current best criterion at one node. Thus the technique can greatly reduce the computation cost of searching for the best candidates.

Three model fit criteria are available for the search: the R-square (SELECT=RSQUARE), the adjusted R-square (SELECT=ADJRSQ), and Mallow's C_p statistic (SELECT=CP). The computation time that the analysis requires is highly dependent on the data and on the values of the BEST=, MINEFFECTS=, and MAXEFFECTS= *method-options*. In particular, for these three criteria for the SELECT= *method-option* and a large value of the BEST= *method-option*, adding one more effect to the list from which regressors are selected might significantly increase the computation time.

RSQUARE

The SELECT=RSQUARE *method-option* finds subsets of effects that best predict a dependent variable by linear regression in the given data. You can specify the largest and smallest number of effects to appear in a subset and the number of subsets of each size to be selected. This *method-option* can efficiently perform all possible subset regressions and display the models in decreasing order of R-square magnitude within each subset size. Other statistics are available for comparing subsets of different sizes.

The subset models that this SELECT=RSQUARE *method-option* selects are optimal in terms of R-square for the given data, but they are not necessarily optimal for the population from which the data sample is drawn or for any other data for which you might want to make predictions. If a subset model is selected on the basis of a large R-square value or any other criterion commonly used for model selection, then all regression statistics that are computed for that model under the assumption that the model is given a priori are biased, including all other computed statistics.

Although the SELECT=RSQUARE *method-option* is a useful tool for building exploratory models, no single statistical method can be relied on to identify the "true" model. Effective model building requires substantive theory to suggest relevant predictors and plausible functional forms for the model.

The SELECT=RSQUARE *method-option* differs from the other selection *method-options* in that it always identifies the model that has the largest R-square value for each number of effects considered. The other selection methods are not guaranteed to find the model with the largest R-square.

ADJRSQ

The SELECT=ADJRSQ *method-option* is similar to the SELECT=RSQUARE *method-option*, except that the adjusted R-square statistic is used as the criterion for selecting models, and the option finds the models with the highest adjusted R-square value within the range of sizes.

CP

The SELECT=CP *method-option* is similar to the SELECT=ADJRSQ *method-option*, except that Mallows' C_p statistic is used as the criterion for the best-subset selection. Models are listed in ascending order of C_p.

Folded Concave Penalized Selection

This section applies to the REGSELECT procedure.

Recall that adaptive LASSO selection consists of a two-stage approach that involves an initial estimator to reduce bias for large regression coefficients. An alternative single-stage approach is to use a penalty that tapers off as $\boldsymbol{\beta}$ becomes larger in absolute value.

More specifically, the folded concave penalized (FCP) selection coefficients $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_m)^T$ are the solution to the regularized least squares

$$\min \frac{1}{2n}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \sum_{j=1}^{m} P_{\lambda,\alpha}(|\beta_j|)$$

where $P_{\lambda,\alpha}(\cdot)$ is a folded concave penalty.

Figure 2.6 Penalty Functions: LASSO, SCAD, and MCP

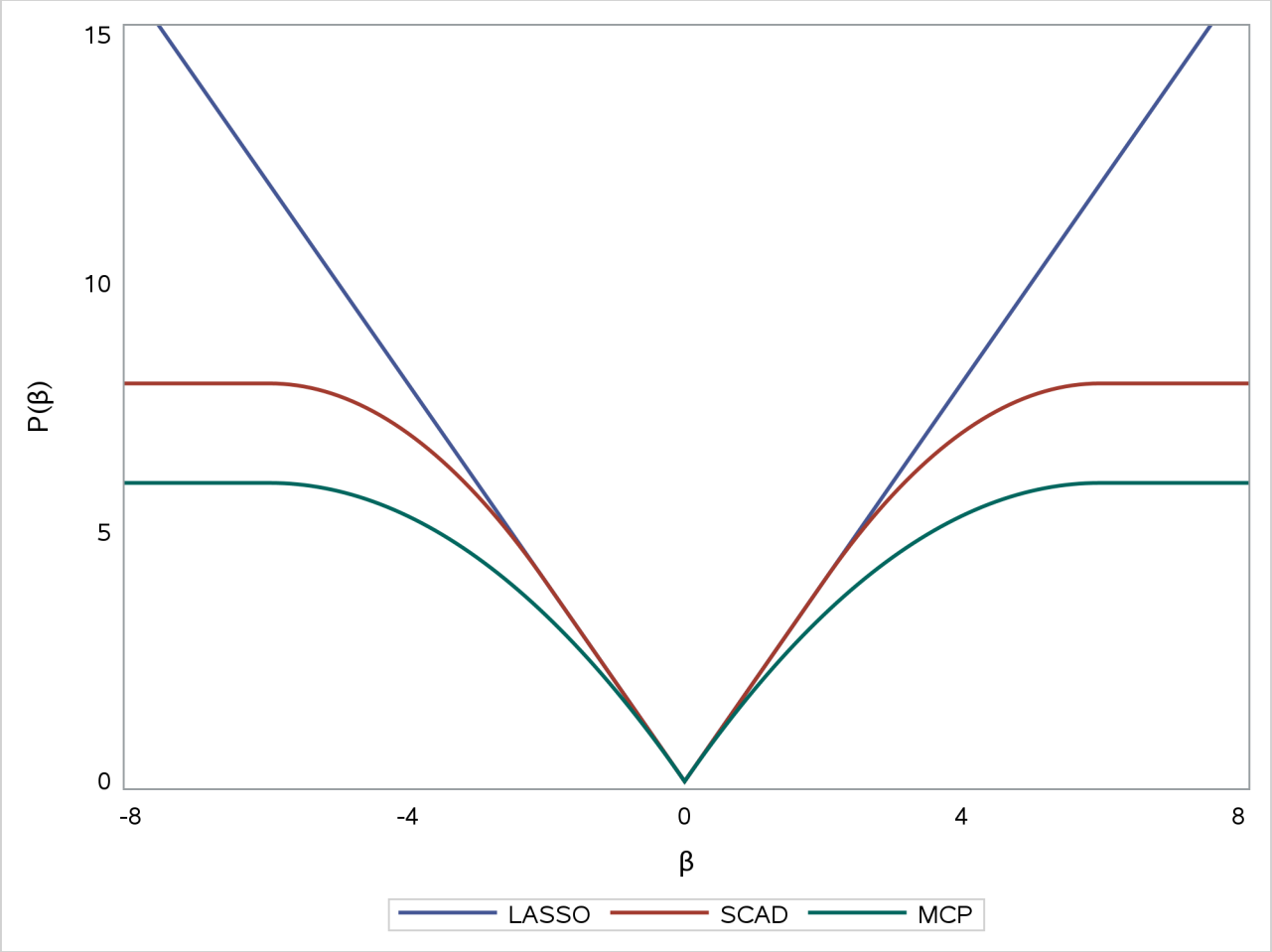


Figure 2.6 shows the penalty functions in the LASSO, SCAD, and MCP methods. When the absolute value of β becomes large, the corresponding folded concave penalty value increases much more slowly than the LASSO penalty value.

If you have a good estimate of α or λ, you can specify the value in either the ALPHA= or LAMBDA= option. If you omit the values for α and λ, then by default the procedure searches for a pair of λ, α that is optimal according to the current value of the CHOOSE= criterion (by default, CHOOSE=SBC).

A variety of nonconvex penalties have been proposed. Two of the earliest and most influential nonconvex penalties are the smoothly clipped absolute deviation (SCAD) and the minimax concave penalty (MCP).

SCAD

In SCAD selection (Fan and Li 2001), the $P_{\lambda,\alpha}(\cdot)$ takes the following form:

$$P^{\text{scad}}_{\lambda,\alpha}(\theta) = \begin{cases} \lambda\theta & \text{if} \quad 0 \le \theta \le \lambda \\ \frac{-1}{2(\alpha-1)}\left(\theta^2 - 2\alpha\lambda\theta + \lambda^2\right) & \text{if} \quad \lambda < \theta < \alpha\lambda \\ \frac{\alpha+1}{2}\lambda^2 & \text{if} \quad \theta > \alpha\lambda \end{cases}$$

The SCAD estimator $\hat{\boldsymbol{\beta}}^{\text{scad}}$ solves the following minimization problem:

$$\hat{\boldsymbol{\beta}}^{\text{scad}} = \arg\min_{\boldsymbol{\beta}} \left\{ \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + n\sum_{j=1}^{m} P_{\lambda,\alpha}^{\text{scad}}(|\beta_j|) \right\}$$

The quadratic program (QP) reformulation of the preceding SCAD problem is given by

$$\begin{aligned} \min_{\boldsymbol{\beta},\mathbf{g},\mathbf{h}\in R^m} \quad & \frac{1}{2}\left[\boldsymbol{\beta}^T\mathbf{X}^T\mathbf{X}\boldsymbol{\beta} + n(\alpha-1)\mathbf{g}^T\mathbf{g} + 2n\mathbf{g}^T\mathbf{h}\right] - \mathbf{y}^T\mathbf{X}\boldsymbol{\beta} - n\alpha\lambda\mathbf{1}^T\mathbf{g} \\ \text{subject to} \quad & -\mathbf{h} \le \boldsymbol{\beta} \le \mathbf{h} \\ & \mathbf{0} \le \mathbf{g} \le \lambda\mathbf{1} \end{aligned} \tag{1}$$

where $\mathbf{0} = (0,\ldots,0)^T$ and $\mathbf{1} = (1,\ldots,1)^T$.

Liu, Yao, and Li (2016) show that the preceding QP (1) is equivalent to the following mixed integer linear program (MILP):

$$\begin{aligned} \min_{\boldsymbol{\beta},\mathbf{g},\mathbf{h},\{\boldsymbol{\gamma}_k\}_{k=1}^4,\{\mathbf{z}_k\}_{k=1}^4\in R^m} \quad & -\frac{1}{2}\mathbf{y}^T\mathbf{X}\boldsymbol{\beta} - \frac{1}{2}n\alpha\lambda\mathbf{1}^T\mathbf{g} - \frac{1}{2}\lambda\mathbf{1}^T\boldsymbol{\gamma}_4 \\ \text{subject to} \quad & \mathbf{X}^T\mathbf{X}\boldsymbol{\beta} - \mathbf{X}^T\mathbf{y} + \boldsymbol{\gamma}_1 - \boldsymbol{\gamma}_2 = \mathbf{0} \\ & n\mathbf{g} - \boldsymbol{\gamma}_1 - \boldsymbol{\gamma}_2 = \mathbf{0} \\ & n(\alpha-1)\mathbf{g} + n\mathbf{h} - \boldsymbol{\gamma}_3 + \boldsymbol{\gamma}_4 - n\alpha\lambda\mathbf{1} = \mathbf{0} \\ & \mathbf{0} \le \boldsymbol{\gamma}_1 \le \mathcal{M}\mathbf{z}_1, \quad \mathbf{0} \le \mathbf{h} - \boldsymbol{\beta} \le \mathcal{M}(\mathbf{1} - \mathbf{z}_1) \\ & \mathbf{0} \le \boldsymbol{\gamma}_2 \le \mathcal{M}\mathbf{z}_2, \quad \mathbf{0} \le \mathbf{h} + \boldsymbol{\beta} \le \mathcal{M}(\mathbf{1} - \mathbf{z}_2) \\ & \mathbf{0} \le \boldsymbol{\gamma}_3 \le \mathcal{M}\mathbf{z}_3, \quad \mathbf{0} \le \mathbf{g} \le \mathcal{M}(\mathbf{1} - \mathbf{z}_3) \\ & \mathbf{0} \le \boldsymbol{\gamma}_4 \le \mathcal{M}\mathbf{z}_4, \quad \mathbf{0} \le \lambda\mathbf{1} - \mathbf{g} \le \mathcal{M}(\mathbf{1} - \mathbf{z}_4) \\ & \mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \mathbf{z}_4 \in \{0,1\}^m \end{aligned} \tag{2}$$

where $\mathcal{M} > 0$ is a properly large constant.

MCP

In MCP selection (Zhang 2010), the $P_{\lambda,\alpha}(\cdot)$ takes the following form:

$$P_{\lambda,\alpha}^{\text{mcp}}(\theta) = \begin{cases} \lambda\theta - \frac{1}{2\alpha}\theta^2 & \text{if} \quad 0 \le \theta \le \alpha\lambda \\ \frac{\alpha}{2}\lambda^2 & \text{if} \quad \theta > \alpha\lambda \end{cases}$$

The MCP estimator $\hat{\boldsymbol{\beta}}^{\text{mcp}}$ solves the following minimization problem:

$$\hat{\boldsymbol{\beta}}^{\text{mcp}} = \arg\min_{\boldsymbol{\beta}} \left\{ \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + n\sum_{j=1}^{m} P_{\lambda,\alpha}^{\text{mcp}}(|\beta_j|) \right\}$$

The QP reformulation of the preceding MCP problem is given by

$$\begin{aligned} \min_{\boldsymbol{\beta},\mathbf{g},\mathbf{h}\in R^m} \quad & \frac{1}{2}\left[\boldsymbol{\beta}^T\mathbf{X}^T\mathbf{X}\boldsymbol{\beta} + \frac{n}{\alpha}\mathbf{g}^T\mathbf{g} - \frac{2n}{\alpha}\mathbf{g}^T\mathbf{h}\right] - \mathbf{y}^T\mathbf{X}\boldsymbol{\beta} + n\lambda\mathbf{1}^T\mathbf{h} \\ \text{subject to} \quad & -\mathbf{h} \le \boldsymbol{\beta} \le \mathbf{h} \\ & \mathbf{0} \le \mathbf{g} \le \alpha\lambda\mathbf{1} \end{aligned} \tag{3}$$

Liu, Yao, and Li (2016) show that the preceding QP (3) is equivalent to the following MILP:

$$\begin{aligned}
\min_{\boldsymbol{\beta},\mathbf{g},\mathbf{h},\{\boldsymbol{\gamma}_k\}_{k=1}^4,\{\mathbf{z}_k\}_{k=1}^4 \in R^m} \quad & -\frac{1}{2}\mathbf{y}^T\mathbf{X}\boldsymbol{\beta} + \frac{1}{2}n\lambda\mathbf{1}^T\mathbf{h} - \frac{1}{2}\alpha\lambda\mathbf{1}^T\boldsymbol{\gamma}_4 \\
\text{subject to} \quad & \mathbf{X}^T\mathbf{X}\boldsymbol{\beta} - \mathbf{X}^T\mathbf{y} + \boldsymbol{\gamma}_1 - \boldsymbol{\gamma}_2 = \mathbf{0} \\
& \frac{n}{\alpha}\mathbf{g} + \boldsymbol{\gamma}_1 + \boldsymbol{\gamma}_2 - n\lambda\mathbf{1} = \mathbf{0} \\
& \frac{n}{\alpha}\mathbf{g} - \frac{n}{\alpha}\mathbf{h} - \boldsymbol{\gamma}_3 + \boldsymbol{\gamma}_4 = \mathbf{0} \\
& \mathbf{0} \le \boldsymbol{\gamma}_1 \le \mathcal{M}\mathbf{z}_1, \quad \mathbf{0} \le \mathbf{h} - \boldsymbol{\beta} \le \mathcal{M}(\mathbf{1} - \mathbf{z}_1) \\
& \mathbf{0} \le \boldsymbol{\gamma}_2 \le \mathcal{M}\mathbf{z}_2, \quad \mathbf{0} \le \mathbf{h} + \boldsymbol{\beta} \le \mathcal{M}(\mathbf{1} - \mathbf{z}_2) \\
& \mathbf{0} \le \boldsymbol{\gamma}_3 \le \mathcal{M}\mathbf{z}_3, \quad \mathbf{0} \le \mathbf{g} \le \mathcal{M}(\mathbf{1} - \mathbf{z}_3) \\
& \mathbf{0} \le \boldsymbol{\gamma}_4 \le \mathcal{M}\mathbf{z}_4, \quad \mathbf{0} \le \alpha\lambda\mathbf{1} - \mathbf{g} \le \mathcal{M}(\mathbf{1} - \mathbf{z}_4) \\
& \mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \mathbf{z}_4 \in \{0,1\}^m
\end{aligned} \tag{4}$$

where $\mathcal{M} > 0$ is a properly large constant.

SOLVER

When you specify SOLVER=MILP, the coefficients $\boldsymbol{\beta}$ are obtained by solving the minimization problem (2) or (3). Furthermore, you can also specify the BIGM= option for the $\mathcal{M}$ value and the INTTOL= option for the tolerance of integer variables $\{\mathbf{z}_k\}_{k=1}^4$. By default, $\mathcal{M} = 2|\mathbf{y}^T\mathbf{X}|_\infty$ and INTTOL=1E–7.

When you specify SOLVER=NLP, the coefficients $\boldsymbol{\beta}$ are obtained by solving the minimization problem (1) or (3).

LAMBDAGRID

When you search for the optimal model, the default values of $\lambda_{\max}$ and $\lambda_{\min}$ are, respectively,

$$\lambda_{\max} = \sigma_y\sqrt{\log(m)/m}, \quad \lambda_{\min} = \frac{\sigma_y}{20\sqrt{\log(m)}}$$

where σ_y is the standard deviation of $\mathbf{y}$, m is the length of $\boldsymbol{\beta}$, and log is the natural logarithm.

Suppose that the number of searching λ steps is n_λ. You can define the value by specifying the MAXITER-LAMBDA= option. When you specify LAMBDAGRID=LINSPACE, the values of λ series are generated as follows:

$$\lambda_i^{\text{linspace}} = \lambda_{\min} + (i-1) \times \delta_\lambda^{\text{linspace}}, \quad i = 1, 2, \ldots, n_\lambda$$

where $\delta_\lambda^{\text{linspace}} = (\lambda_{\max} - \lambda_{\min})/(n_\lambda - 1)$.

When you specify LAMBDAGRID=LOGSPACE, the values of λ series are generated as follows:

$$\log_{10}(\lambda_i^{\text{logspace}}) = \log_{10}(\lambda_{\min}) + (i-1) \times \delta_\lambda^{\text{logspace}}, \quad i = 1, 2, \ldots, n_\lambda$$

where $\delta_\lambda^{\text{logspace}} = (\log_{10}(\lambda_{\max}) - \log_{10}(\lambda_{\min}))/(n_\lambda - 1)$.

Model Selection Plots

This section describes the ODS graphical displays that you can request by using the PLOTS= option in the SELECTION statement.

This section applies to the following procedures: GENSELECT, LOGSELECT, PHSELECT, QTRSELECT, and REGSELECT.

The example plots shown in this section are produced by using the REGSELECT procedure.

The following DATA step produces simulated example data that contain a variable that you can use to assign observations to the training, validation, and testing roles. In this case, each role has 5,000 observations. These statements assume that your libref is named mylib, as in the section "Using CAS Sessions and CAS Engine Librefs" on page 7, but you can substitute any appropriately defined libref.

```
data mylib.exampleData;
   drop i j;
   array x{20} x1-x20;
   array c{5}  c1-c5;

   call streaminit(1);

 do i=1 to 15000;
    do j=1 to dim(x);
       x{j} = rand('NORMAL');
    end;

    do j=1 to dim(c);
       c{j} = 1+ int(rand('UNIFORM')*3);
    end;

    y = 1 + x1 + 2*x5 - 1.3*x10 + x20 + 5*(c1=1)+3*(c1=3) - 4*c4 + 25*rand('NORMAL');

         if mod(i,3)=1 then Role = 'TRAIN';
    else if mod(i,3)=2 then Role = 'VAL';
    else                    Role = 'TEST';

    output;
 end;
run;
```

The following statements perform forward selection to obtain a parsimonious linear regression model for the response:

```
ods graphics on;

proc regselect data=mylib.exampleData;
   partition rolevar=Role(train='TRAIN' validate='VAL' test='TEST');
   class c: ;
   model y = x: c:  ;
```

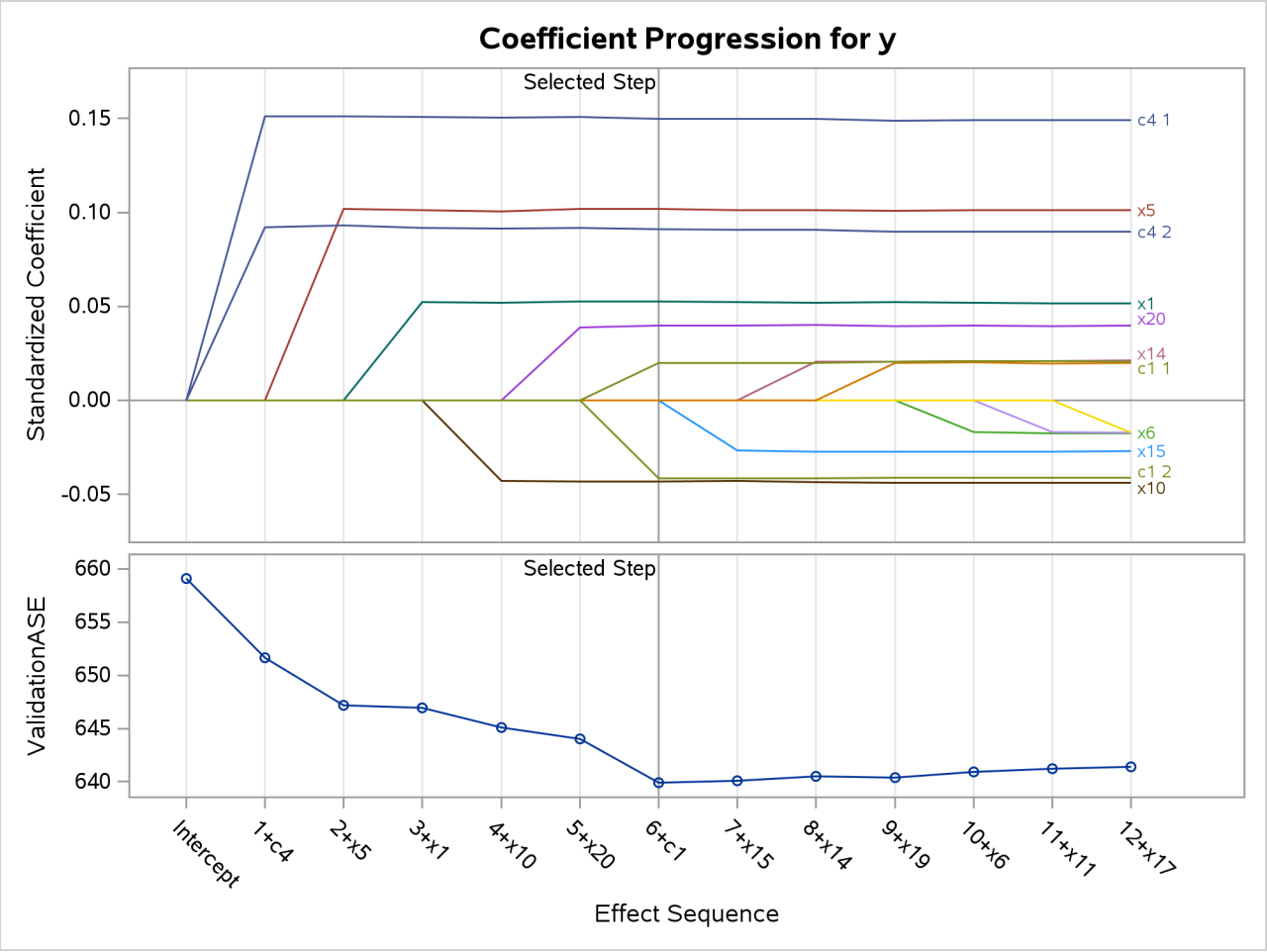
```
   selection method=forward(stop=AICC CHOOSE=validate) plots=all;
run;
```

Because you specified the PLOTS=ALL option in the SELECTION statement and have enabled ODS Graphics, the plots that are described in the following sections are produced.

Coefficient Panel

Figure 2.7 shows the coefficient panel, which contains two plots that show how the standardized coefficients and the criterion used to choose the final model evolve as the selection progresses.

Figure 2.7 Coefficient Panel



The upper plot in the panel displays the standardized coefficients as a function of the step number. To help trace the changes in a parameter, the standardized coefficients for each parameter are connected by lines. Coefficients that correspond to effects not in the selected model at a step are zero and hence not observable. Parameters that are nonzero at the final step of the selection are labeled if their magnitudes are greater than 1% of the range of the magnitudes of all the nonzero parameters at this step. To avoid collision, labels that correspond to parameters that have similar values at the final step might be suppressed.

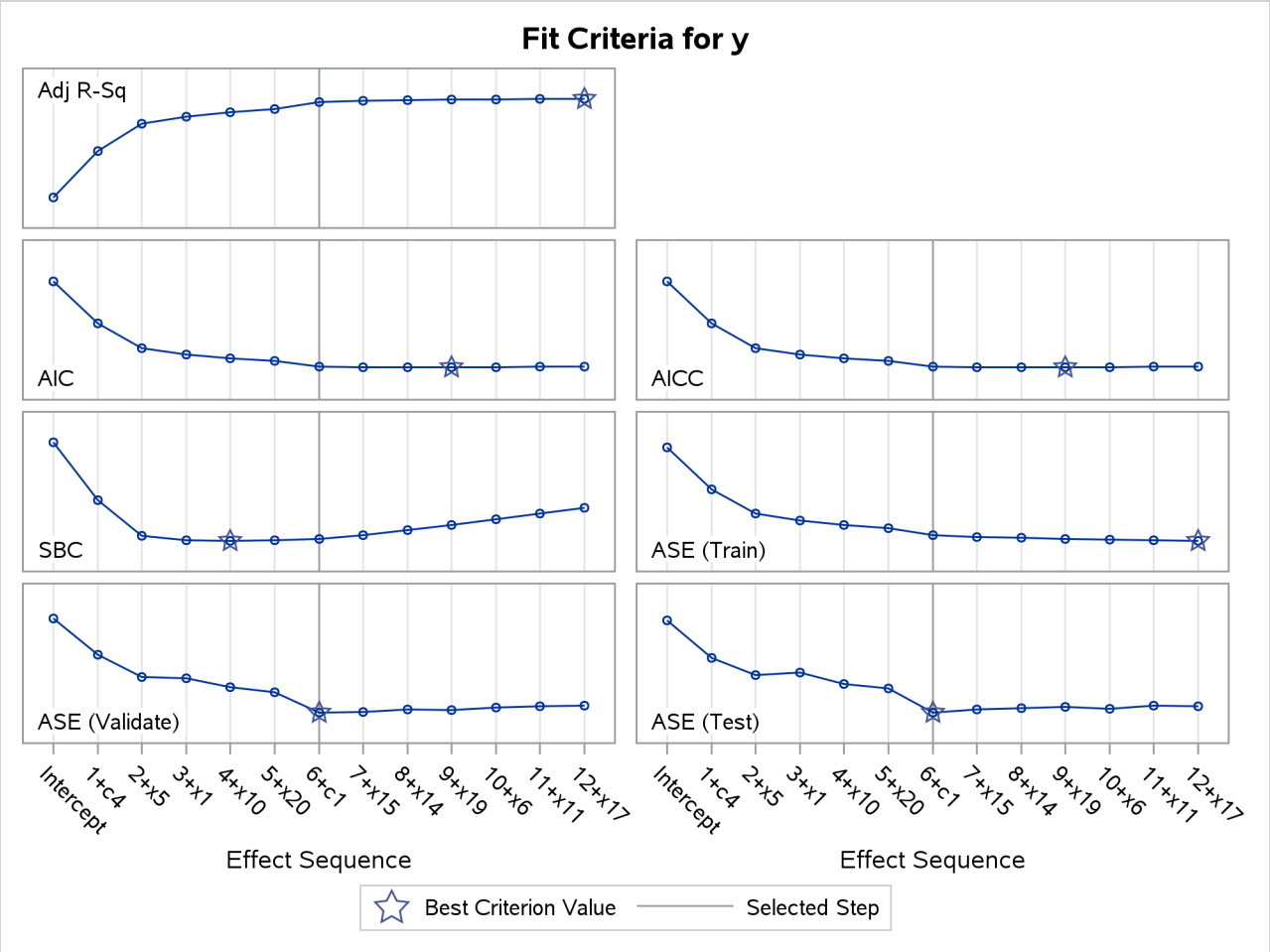
The lower plot in the panel shows how the criterion used to choose among the examined models progresses.

The selected step occurs at the optimal value of this criterion. In this example, this criterion is the average square error (ASE) on the validation data, and it achieves its minimal value at step 6 of the forward selection.

Criterion Panel

Figure 2.8 shows the criterion panel, which contains plots that show the progression of fit criteria.

Figure 2.8 Criterion Panel

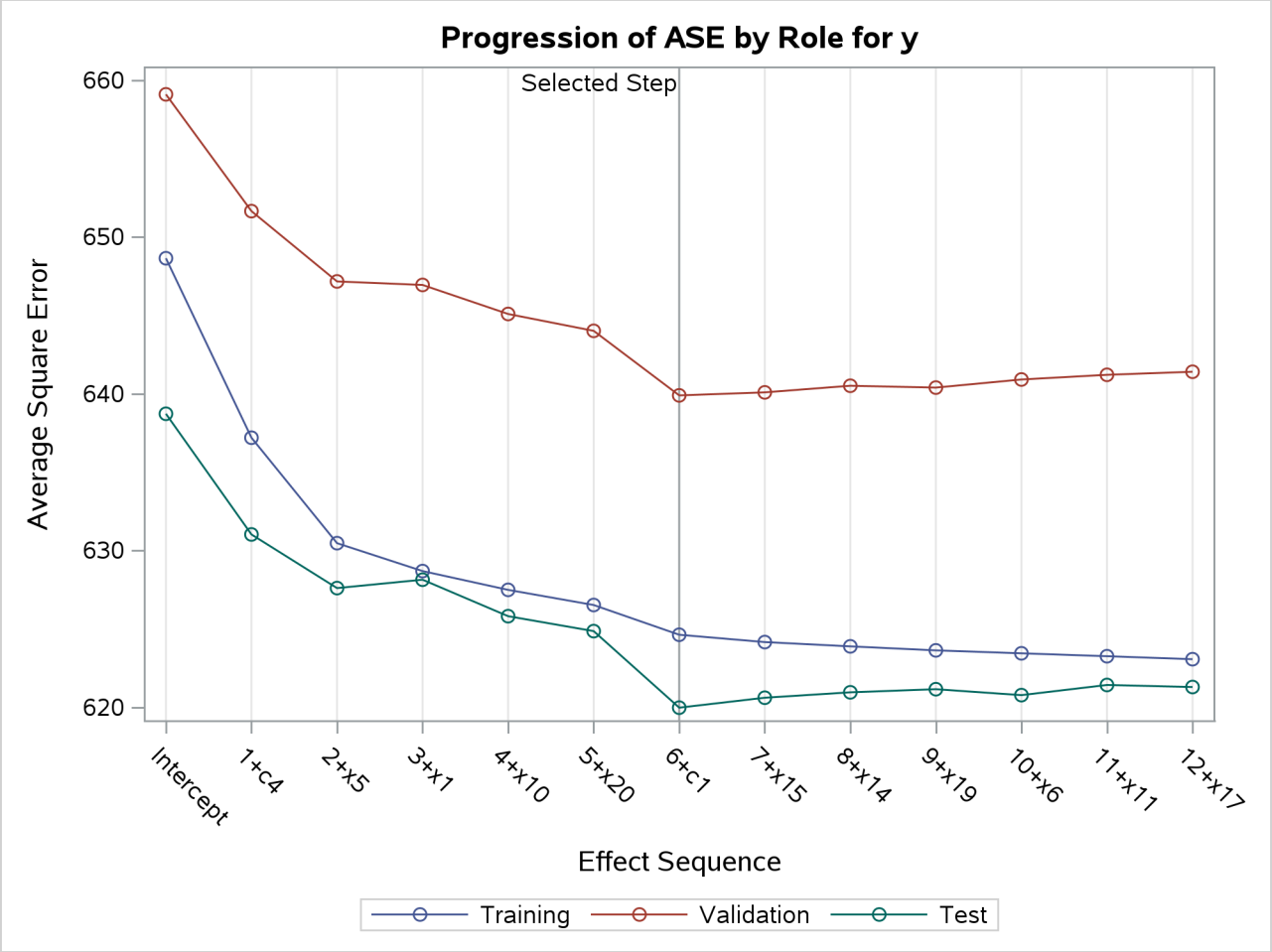


The selected step is displayed as a vertical reference line on the plot of each criterion. The optimal value of each criterion is indicated with the "star" marker. Note that it is possible that a better value of a criterion might have been reached if more steps of the selection process had been done.

Fit by Role Plot

Figure 2.9 displays the fit by role plot, which shows the progression of a fit statistic evaluated separately on the training, validation, and test data. If you do not partition your data into roles by using a PARTITION statement, then the fit by role plot is not produced.

Figure 2.9 Fit by Role Plot



You use this plot to detect overfitting of the training data. For many types of regression (including the linear regression that is used in the examples in this section), the fit statistic shown in this plot is the average square error (ASE) evaluated separately for each data role. The ASE decreases monotonically on the training data as parameters are added to a model. However, the ASE on test and validation data usually starts increasing when overfitting occurs.

Using Options to Customize Plots

In some cases, particularly when the final step contains a large number of parameters, you might be interested in using the coefficient panel only to discern if and when the parameters in the model are essentially unchanged beyond a certain step. In such cases, you might want to suppress the labeling of the parameters and use a numeric axis on the horizontal axis of the plot. You can do this by using the STEPAXIS= and MAXPARMLABEL= global plot options. The following statements provide an example:

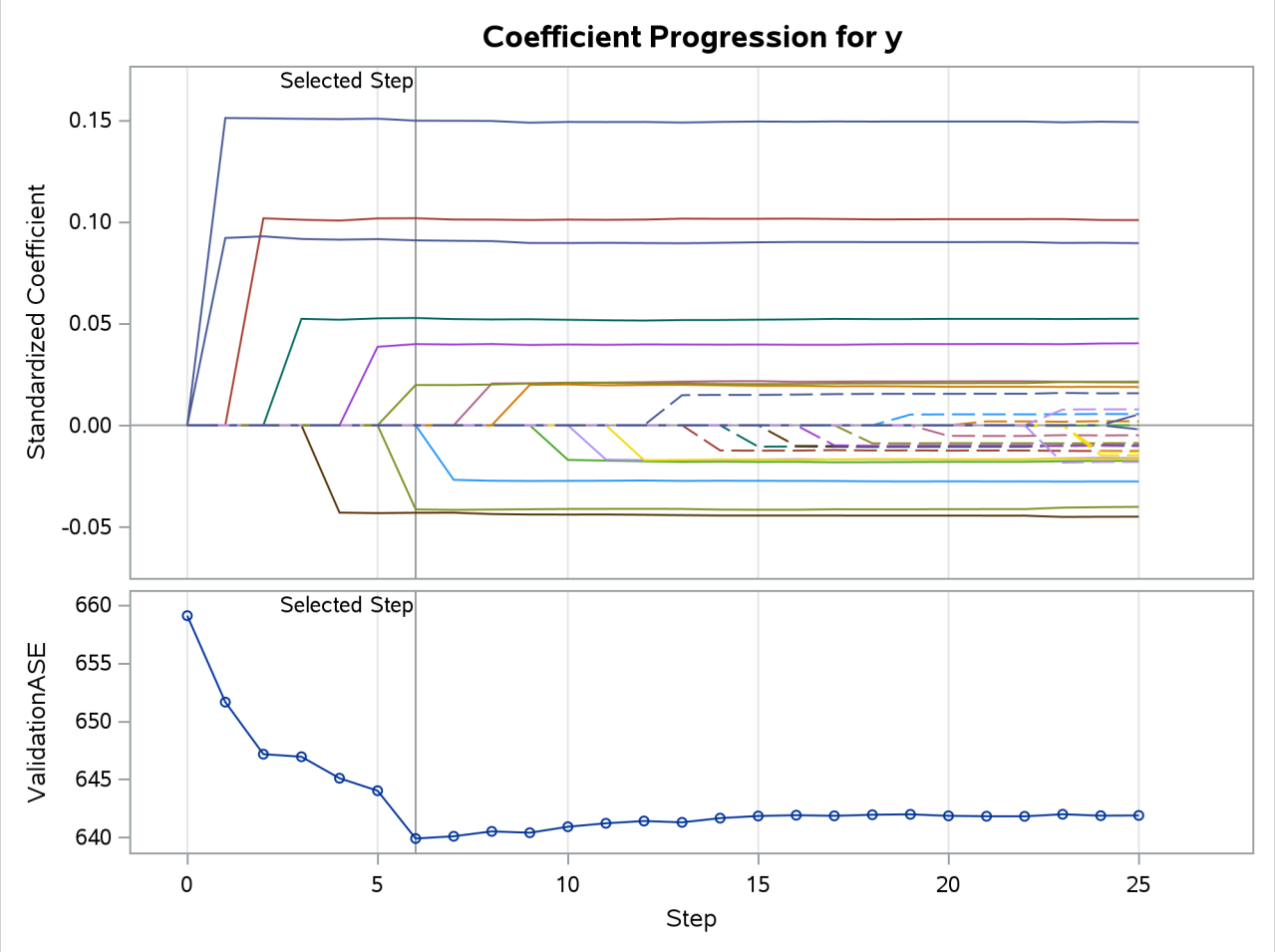
```
proc regselect data=mylib.exampleData;
   partition rolevar=Role(train='TRAIN' validate='VAL' test='TEST');
   class c: ;
   model y = x: c: ;
```

```
    selection method=forward(stop=none choose=validate)
              plots(maxparmlabel=1 stepaxis=number)=(coefficients criteria(unpack));
run;
```

The STEPAXIS=NUMBER option requests that a numeric horizontal axis showing step number be used on all plots. The MAXPAMLABEL=1 option suppresses the labels for the parameters in the coefficient progression panel.

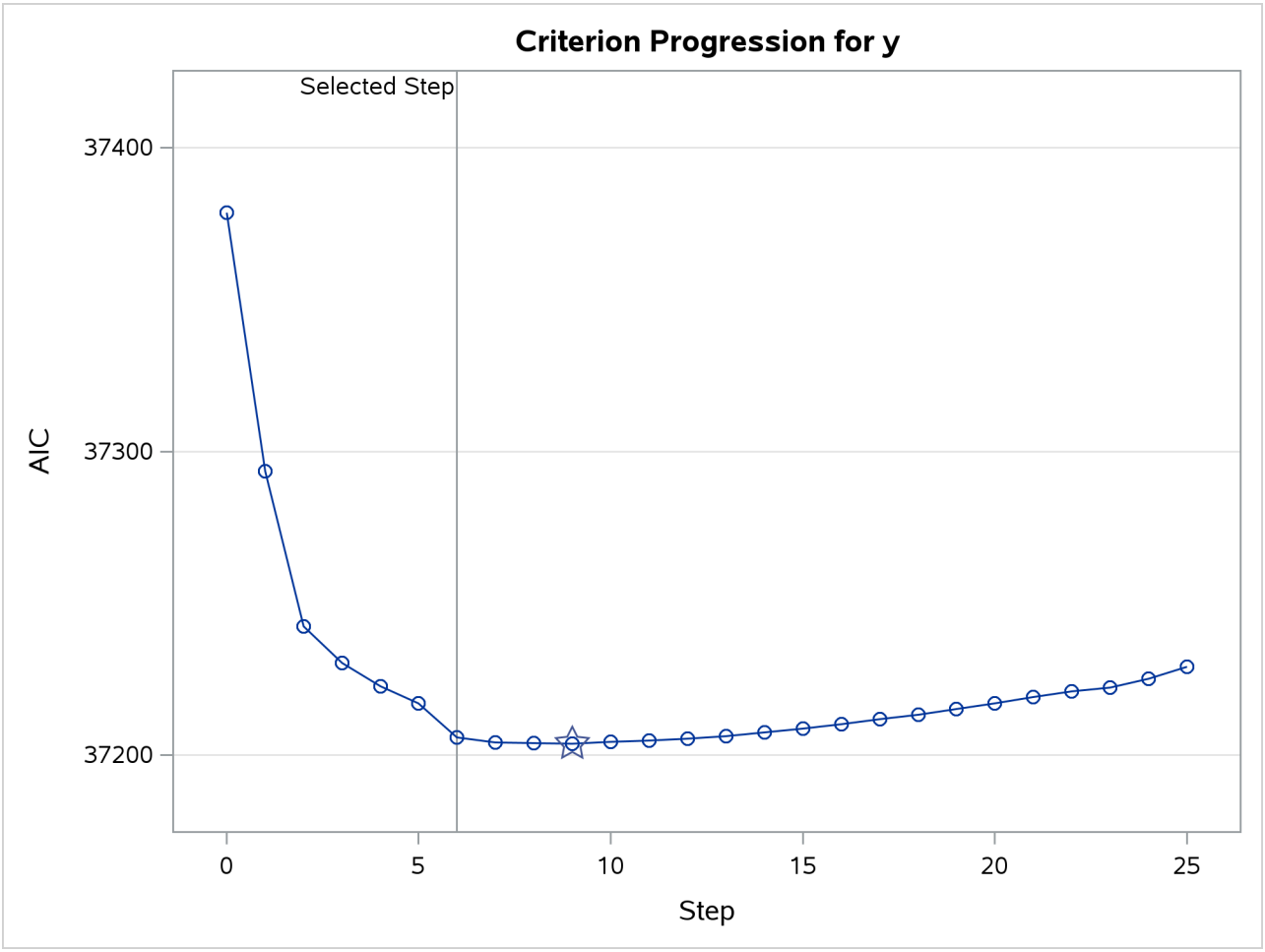
The "Coefficient Progression Panel" is shown in Figure 2.10. You can see that the standardized coefficients of the selected parameters do not vary greatly after the selected step (step 6).

Figure 2.10 Coefficient Panel



The UNPACK option for the criterion panel specifies that the individual plots in this panel be shown as separate plots. Figure 2.11 shows the plot for the AIC criterion.

Figure 2.11 AIC Criterion



Examining Specific Step Ranges

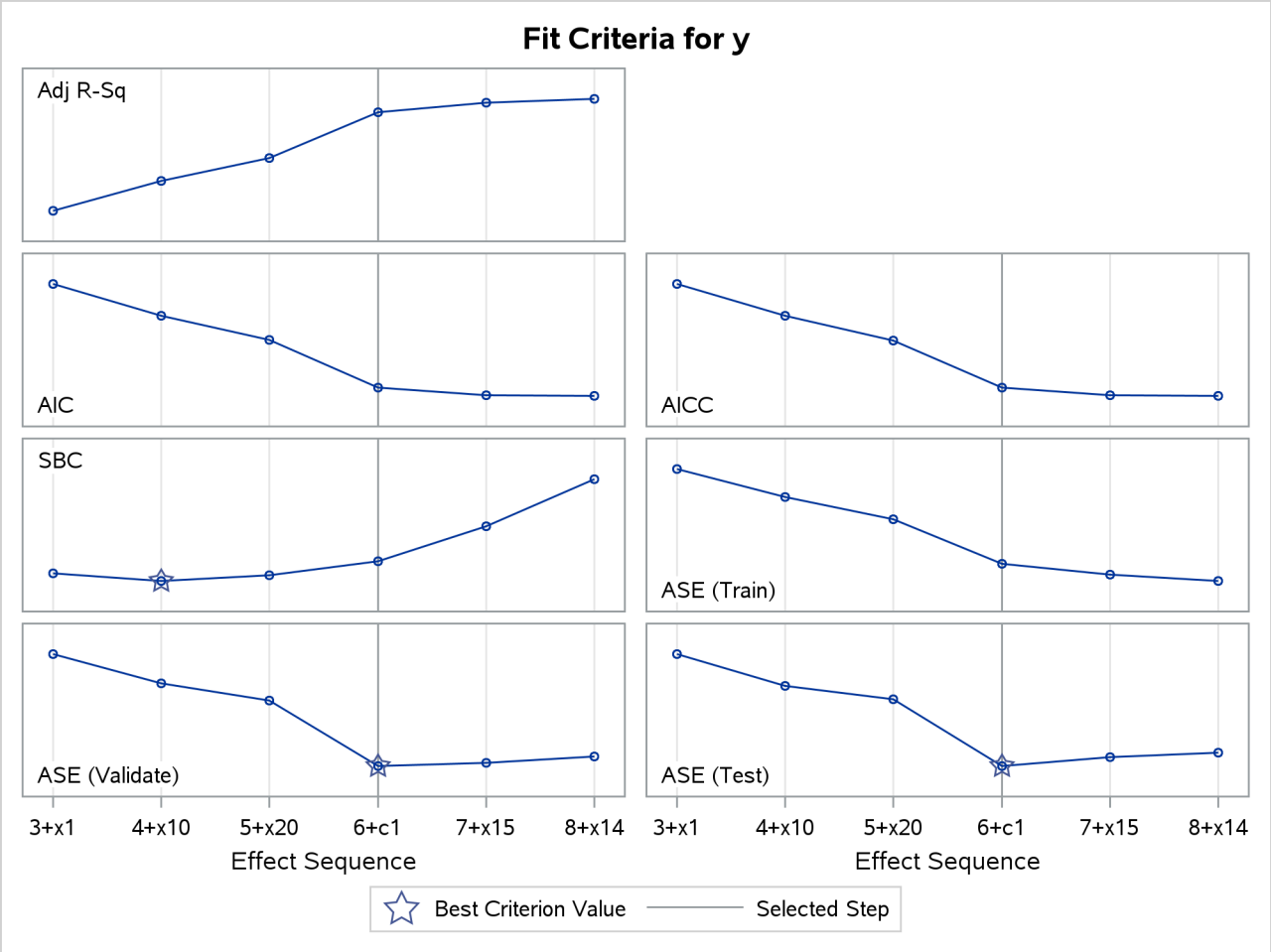
The coefficient panel, criterion panel, and average square error plot display information for all the steps examined in the selection process. In some cases, you might want to focus attention on just a particular step range. For example, it is hard to discern the variation in the criteria displayed in Figure 2.8 near the selected step because the variation in these criteria in the steps close to the selected step is small relative to the variation across all steps. You can request a range of steps to display by using the STARTSTEP= and ENDSTEP= suboptions of the PLOTS= option. The following statements provide an example:

```
proc regselect data=mylib.exampleData;
   partition rolevar=Role(train='TRAIN' validate='VAL' test='TEST');
   class c: ;
   model y = x: c: ;
   selection method=forward(stop=none choose=validate)
             plots(startstep=3 endstep=8)=criteria;
run;

ods graphics off;
```

Figure 2.12 shows the progression of the fit criteria between steps 3 and 8. Note that if the optimal value of a criterion does not occur in this specified step range, then no optimal marker appears for that criterion.

Figure 2.12 Criterion Panel for Specified Step Range



Informative Missingness

This section applies to the following procedures: GENSELECT, LOGSELECT, PHSELECT, QTRSELECT, and REGSELECT.

The concept of informative missingness is one way to account for missing values in statistical analyses and, in particular, statistical modeling. Missing values can be a problem because they reduce the amount of available data. When you work with classification variables (factors, which are levelized variables), you can treat a missing value as an actual level of the variable and allow it to participate in the analysis.

However, when continuous variables have missing values, the observation is removed from the analysis. In data that have many missing values, removing observations can reduce the amount of available data greatly,

and the sets of observations used in one model versus another model can vary based on which variables are included in the model.

Of course, there are many reasons for missing values, and substituting values for missing values has to be done with caution. For example, the famous Framingham Heart study data set contains 5,209 observations on subjects in a longitudinal study that helped understand the relationship between smoking, cholesterol, and coronary heart disease. One of the variables in the data set is AgeCHDdiag. This variable represents the age at which a patient was diagnosed with coronary heart disease (CHD). If you include this variable in a statistical model, only 1,449 observations are available, because the value cannot be observed unless a patient has experienced CHD. Including this variable acts as a filter that reduces the analysis set to the subjects who have CHD. You cannot impute the value for subjects where the variable has a missing value, because you cannot impute an age at which someone who has not had CHD would have contracted coronary heart disease.

With informative missingness, you are not so much substituting imputed values for the missing values as you are modeling the missingness. Consider a simple linear regression model:

$$y = \beta_0 + \beta_1 x + \epsilon$$

Suppose that some of the values for the regressor variable x are missing. The fitted model uses only observations for which y and x have been observed.

In order to predict the outcome y for an observation that has a missing x, either you assume that y is missing or you substitute a value (such as the average value, $\bar{x}$) for the missing x. Because the estimate for the intercept is in the simple linear regression model, the predicted value would be the average response of the nonmissing values, $\bar{y}$.

With informative missingness, you extend the model by adding extra effects for each effect that contains at least one continuous variable. In the simple linear regression model, you add one column to the model and slightly change the content of the x variable:

$$y = \beta_0 + \beta_1 x^* + \beta_2 x\_miss + \epsilon_1$$

The variable x^* contains the original values of x if they are not missing, and the average of x otherwise:

$$x^* = \begin{cases} x & \text{if } x \text{ is not missing} \\ \bar{x} & \text{otherwise} \end{cases}$$

The variable *x_miss* is a dummy variable whose value is 1 when x is missing, and 0 otherwise:

$$x\_miss = \begin{cases} 1 & \text{if } x \text{ is missing} \\ 0 & \text{otherwise} \end{cases}$$

The fitted model is not the same model that results from substituting $\bar{x}$ for the missing values during training, because the model that simply substitutes $\bar{x}$ for the missing values is

$$y = \beta_0 + \beta_1 x^* + \epsilon_2$$

The informative missing model has an extra parameter, and unless all values of *x_miss* are 0 (in which case there are no missing values), the informative missing model has a higher R-square value, because it picks up more variation.

The parameter estimate for β_2 measures the amount by which the predicted value differs from a predicted value at $\bar{x}$.

Using Validation and Test Data

This section applies to the following procedures: GAMSELECT, GENSELECT, LOGSELECT, PHSELECT, QTRSELECT, REGSELECT, and TREESPLIT.

When you have sufficient data, you can divide your data into three parts called the training, validation, and test data. During the selection process, models are fit on the training data, and the prediction errors for the models so obtained are found by using the validation data. This prediction error on the validation data can be used to decide when to terminate the selection process and to decide which model to select. Finally, after a model has been selected, the test set can be used to assess how the selected model generalizes on data that played no role in selecting the model.

In some cases, you might want to use only training and test data. For example, you might decide to use an information criterion to decide which effects to include and when to terminate the selection process. In this case, no validation data are required, but test data can still be useful in assessing the predictive performance of the selected model. In other cases, you might decide to use validation data during the selection process but forgo assessing the selected model on test data. Hastie, Tibshirani, and Friedman (2001) note that it is difficult to provide a general rule for how many observations you should assign to each role. They note that a typical split might be 50% for training and 25% each for validation and testing.

You use a PARTITION statement to logically subdivide the input data table into separate roles. You can specify the fractions of the data that you want to reserve as test data and validation data. For example, the following statements randomly divide the inData data table, reserving 50% for training and 25% each for validation and testing:

```
proc logselect data=mylib.inData;
   partition fraction(test=0.25 validate=0.25);
   ...
run;
```

You can specify the SEED= option in the PARTITION statement to create the same partition data tables for a particular number of compute nodes. However, changing the number of compute nodes changes the initial distribution of data, resulting in different partition data tables.

In some cases, you might need to exercise more control over the partitioning of the input data table. You can do this by naming both a variable in the input data table and a formatted value of that variable for each role. For example, the following statements assign roles to the observations in the inData data table that are based on the value of the variable Group in that data table. Observations whose value of Group is `'Group 1'` are assigned for testing, and those whose value is `'Group 2'` are assigned to training. All other observations are ignored.

```
proc logselect data=mylib.inData;
   partition roleVar=Group(test='Group 1' train='Group 2')
   ...
run;
```

When you have reserved observations for training, validation, and testing, a model that is fit on the training data is scored on the validation and test data, and statistics are computed separately for each of these subsets.

Using the Validation Statistic as the CHOOSE= Criterion

When you specify the CHOOSE=VALIDATE suboption of the METHOD= option in the SELECTION statement, the ASE is computed on the validation data for the models at each step of the selection process. The smallest model at any step that yields the smallest validation ASE is selected.

Using the Validation Statistic as the STOP= Criterion

When you specify the STOP=VALIDATE suboption of the METHOD= option in the SELECTION statement, the ASE is computed on the validation data for the models at each step of the selection process. At step k of the selection process, the best candidate effect to enter or leave the current model is determined and the validation ASE for this new model is computed. If this validation ASE is greater than the validation ASE for the model at step k, then the selection process terminates at step k.

Multithreading

Threading refers to the organization of computational work into multiple tasks (processing units that can be scheduled by the operating system). A task is associated with a thread. Multithreading refers to the concurrent execution of threads. When multithreading is possible, substantial performance gains can be realized compared to sequential (single-threaded) execution. The number of threads spawned by a procedure in this book is determined by your installation.

The tasks that are multithreaded by procedures in this book are primarily defined by dividing the data that are processed on a single machine among the threads—that is, the procedures implement multithreading through a data-parallel model. For example, if the input data table has 1,000 observations and the procedure is running on four threads, then 250 observations are associated with each thread. All operations that require access to the data are then multithreaded. These operations include the following (not all operations are required for all procedures):

- variable levelization
- effect levelization
- formation of the initial crossproducts matrix
- formation of approximate Hessian matrices for candidate evaluation during model selection
- objective function calculation
- gradient calculation
- Hessian calculation
- scoring of observations

In addition, operations on matrices such as sweeps can be multithreaded provided that the matrices are of sufficient size to realize performance benefits from managing multiple threads for the particular matrix operation.

Choosing an Optimization Algorithm

This section applies to the following procedures: GAMMOD, GENSELECT, LMIXED, LOGSELECT, NLMOD, and PHSELECT.

First- or Second-Order Algorithms

The factors that go into choosing a particular optimization technique for a particular problem are complex. Trial and error can be involved.

For many optimization problems, computing the gradient takes more computer time than computing the function value. Computing the Hessian sometimes takes *much* more computer time and memory than computing the gradient, especially when there are many decision variables. Unfortunately, optimization techniques that do not use some kind of Hessian approximation usually require many more iterations than techniques that do use a Hessian matrix, and as a result the total run time of these techniques is often longer. Techniques that do not use the Hessian also tend to be less reliable. For example, they can terminate more easily at stationary points than at global optima.

Table 2.24 shows which derivatives are required for each optimization technique.

Table 2.24 Derivatives Required

Algorithm	First-Order	Second-Order
TRUREG	x	x
NEWRAP	x	x
NRRIDG	x	x
QUANEW	x	-
DBLDOG	x	-
CONGRA	x	-
LBFGS	x	-
LEVMAR	x	-
NMSIMP	-	-

The second-derivative methods TRUREG, NEWRAP, and NRRIDG are best for small problems for which the Hessian matrix is not expensive to compute. Sometimes the NRRIDG algorithm can be faster than the TRUREG algorithm, but TRUREG can be more stable. The NRRIDG algorithm requires only one matrix with $p(p+1)/2$ double words; TRUREG and NEWRAP require two such matrices. Here, p denotes the number of parameters in the optimization.

The first-derivative methods QUANEW and DBLDOG are best for medium-sized problems for which the objective function and the gradient can be evaluated much faster than the Hessian. In general, the QUANEW and DBLDOG algorithms require more iterations than TRUREG, NRRIDG, and NEWRAP, but each iteration can be much faster. The QUANEW and DBLDOG algorithms require only the gradient to update an approximate Hessian, and they require slightly less memory than TRUREG or NEWRAP.

The first-derivative methods CONGRA and LBFGS are best for large problems for which the objective

function and the gradient can be computed much faster than the Hessian and for which too much memory is required to store the (approximate) Hessian. In general, the CONGRA and LBFGS algorithms require more iterations than QUANEW or DBLDOG, but each iteration can be much faster. Because CONGRA and LBFGS require only a factor of p double-word memory, many large applications can be solved only by CONGRA and LBFGS.

The no-derivative method NMSIMP is best for small problems for which derivatives are not continuous or are very difficult to compute.

The LEVMAR method is appropriate only for least squares optimization problems.

Each optimization method uses one or more convergence criteria that determine when it has converged. An algorithm is considered to have converged when any one of the convergence criteria is satisfied. For example, under the default settings, the QUANEW algorithm converges if ABSGCONV $<$ 1E–5, FCONV $< 2 \times \epsilon$, or GCONV $<$ 1E–8.

By default, procedures in this book apply the NRRIDG algorithm because it can take advantage of multi-threading in Hessian computations and inversions. If the number of parameters becomes large, specifying the QUANEW technique, which is a first-order method with good overall properties, is recommended.

Algorithm Descriptions

The following subsections provide details about each optimization technique and follow the same order as Table 2.24.

Trust Region Optimization (TRUREG)

The trust region method uses the gradient $\mathbf{g}(\boldsymbol{\beta}^{(k)})$ and the Hessian matrix $\mathbf{H}(\boldsymbol{\beta}^{(k)})$; thus, it requires that the objective function $f(\boldsymbol{\beta})$ have continuous first- and second-order derivatives inside the feasible region.

The trust region method iteratively optimizes a quadratic approximation to the nonlinear objective function within a hyperelliptic trust region with radius Δ that constrains the step size that corresponds to the quality of the quadratic approximation. The trust region method is implemented based on Dennis, Gay, and Welsch (1981), Gay (1983), and Moré and Sorensen (1983).

The trust region method performs well for small- to medium-sized problems, and it does not need many function, gradient, and Hessian calls. However, if the computation of the Hessian matrix is computationally expensive, one of the dual quasi-Newton or conjugate gradient algorithms might be more efficient.

Newton-Raphson Optimization with Line Search (NEWRAP)

The NEWRAP technique uses the gradient $\mathbf{g}(\boldsymbol{\beta}^{(k)})$ and the Hessian matrix $\mathbf{H}(\boldsymbol{\beta}^{(k)})$; thus, it requires that the objective function have continuous first- and second-order derivatives inside the feasible region. If second-order derivatives are computed efficiently and precisely, the NEWRAP method can perform well for medium-sized to large problems, and it does not need many function, gradient, and Hessian calls.

This algorithm uses a pure Newton step when the Hessian is positive-definite and when the Newton step reduces the value of the objective function successfully. Otherwise, a combination of ridging and line search is performed to compute successful steps. If the Hessian is not positive-definite, a multiple of the identity matrix is added to the Hessian matrix to make it positive-definite (Eskow and Schnabel 1991).

In each iteration, a line search is performed along the search direction to find an approximate optimum of the objective function. The line-search method uses quadratic interpolation and cubic extrapolation.

Newton-Raphson Ridge Optimization (NRRIDG)

The NRRIDG technique uses the gradient $\mathbf{g}(\boldsymbol{\beta}^{(k)})$ and the Hessian matrix $\mathbf{H}(\boldsymbol{\beta}^{(k)})$; thus, it requires that the objective function have continuous first- and second-order derivatives inside the feasible region.

This algorithm uses a pure Newton step when the Hessian is positive-definite and when the Newton step reduces the value of the objective function successfully. If at least one of these two conditions is not satisfied, a multiple of the identity matrix is added to the Hessian matrix.

Because the NRRIDG technique uses an orthogonal decomposition of the approximate Hessian, each iteration of NRRIDG can be slower than that of the NEWRAP technique, which works with a Cholesky decomposition. However, NRRIDG usually requires fewer iterations than NEWRAP.

The NRRIDG method performs well for small- to medium-sized problems, and it does not require many function, gradient, and Hessian calls. However, if the computation of the Hessian matrix is computationally expensive, one of the dual quasi-Newton or conjugate gradient algorithms might be more efficient.

Quasi-Newton Optimization (QUANEW or DUQUANEW)

The dual quasi-Newton method uses the gradient $\mathbf{g}(\boldsymbol{\beta}^{(k)})$, and it does not need to compute second-order derivatives because they are approximated. It works well for medium-sized to moderately large optimization problems, where the objective function and the gradient can be computed much faster than the Hessian. However, in general the QUANEW technique requires more iterations than the TRUREG, NEWRAP, and NRRIDG techniques, which compute second-order derivatives. The QUANEW technique provides an appropriate balance between the speed and stability required for most nonlinear mixed model applications.

The QUANEW technique implemented by procedures in this book is the dual quasi-Newton algorithm, which updates the Cholesky factor of an approximate Hessian.

In each iteration, a line search is performed along the search direction to find an approximate optimum. The line-search method uses quadratic interpolation and cubic extrapolation to obtain a step size α that satisfies the Goldstein conditions (Fletcher 1987). One of the Goldstein conditions can be violated if the feasible region defines an upper limit of the step size. Violating the left-side Goldstein condition can affect the positive-definiteness of the quasi-Newton update. In that case, either the update is skipped or the iterations are restarted with an identity matrix, resulting in the steepest descent or ascent search direction.

Double-Dogleg Optimization (DBLDOG)

The double-dogleg optimization method combines the ideas of the quasi-Newton and trust region methods. In each iteration, the double-dogleg algorithm computes the step $\mathbf{s}^{(k)}$ as the linear combination of the steepest descent or ascent search direction $\mathbf{s}_1^{(k)}$ and a quasi-Newton search direction $\mathbf{s}_2^{(k)}$:

$$\mathbf{s}^{(k)} = \alpha_1 \mathbf{s}_1^{(k)} + \alpha_2 \mathbf{s}_2^{(k)}$$

The step is requested to remain within a prespecified trust region radius (Fletcher 1987, p. 107). Thus, the DBLDOG subroutine uses the dual quasi-Newton update but does not perform a line search.

The double-dogleg optimization technique works well for medium-sized to moderately large optimization problems, where the objective function and the gradient can be computed much faster than the Hessian. The implementation is based on Dennis and Mei (1979) and Gay (1983), but it is extended for dealing with boundary and linear constraints. The DBLDOG technique generally requires more iterations than the TRUREG, NEWRAP, and NRRIDG techniques, which require second-order derivatives; however, each of the DBLDOG iterations is computationally cheap. Furthermore, the DBLDOG technique requires only gradient calls for the update of the Cholesky factor of an approximate Hessian.

Conjugate Gradient Optimization (CONGRA)

Second-order derivatives are not required by the CONGRA algorithm and are not even approximated. The CONGRA algorithm can be expensive in function and gradient calls, but it requires only $O(p)$ memory for unconstrained optimization. In general, many iterations are required to obtain a precise solution, but each of the CONGRA iterations is computationally cheap.

The CONGRA subroutine should be used for optimization problems with large p. For the unconstrained or boundary-constrained case, CONGRA requires only $O(p)$ bytes of working memory, whereas all other optimization methods require order $O(p^2)$ bytes of working memory. During p successive iterations, uninterrupted by restarts or changes in the working set, the conjugate gradient algorithm computes a cycle of p conjugate search directions. In each iteration, a line search is performed along the search direction to find an approximate optimum of the objective function. The line-search method uses quadratic interpolation and cubic extrapolation to obtain a step size α that satisfies the Goldstein conditions. One of the Goldstein conditions can be violated if the feasible region defines an upper limit for the step size.

Levenberg-Marquardt Optimization (LEVMAR)

The LEVMAR algorithm performs a highly stable optimization; however, for large problems, it consumes more memory and takes longer than the other techniques. The Levenberg-Marquardt optimization technique is a slightly improved variant of the Moré (1978) implementation.

Nelder-Mead Simplex Optimization (NMSIMP)

The Nelder-Mead simplex method does not use any derivatives and does not assume that the objective function has continuous derivatives. The objective function itself needs to be continuous. This technique is quite expensive in the number of function calls, and it might be unable to generate precise results for $p \gg 40$.

The original Nelder-Mead simplex algorithm is implemented and extended to boundary constraints. This algorithm does not compute the objective for infeasible points, but it changes the shape of the simplex adapting to the nonlinearities of the objective function. This change contributes to an increased speed of convergence and uses a special termination criterion.

Limited-Memory BFGS Optimization (LBFGS)

The limited-memory BFGS (LBFGS) method is a variation of the Broyden-Fletcher-Goldfarb-Shanno quasi-Newton (BFGS) method. In the BFGS method, a full $n \times n$ dense approximation of the inverse Hessian matrix, H, is maintained and updated at each iteration. The line-search direction d is obtained from $d = -Hg$, where g is the current gradient. However, in the LBFGS method, only the update information from the most recent m iteration histories is kept; this requires storing only $2 \times m \times n$ doubles in memory. The line-search direction d is constructed from the saved update information and current gradient. The LBFGS method is preferred for solving large-scale problems, because it uses a limited amount of computer memory and less computing time to find a line-search direction. Currently, the LBFGS method supports only two convergence criteria: FCONV and GCONV.

OutputCasTables Table

The OutputCasTables table is a special table that has information about each CAS table that is created during a CAS action execution. The information for each CAS table consists of the CAS table name, the caslib in which the table resides, and the number of columns and rows in the CAS table. Because this table is not a typical ODS table that contains analytical results, you cannot include it in the *table-spec-list* in the DISPLAYOUT statement.

The APPLYROWORDER Option

This section applies to the following procedures: BART, GENSELECT, KCLUS, LOGSELECT, MVOUTLIER, PARTITION, SUPERLEARNER, and TREESPLIT.

Procedures in this book require the input data to reside on a CAS server. Some of these procedures are sensitive to the order of the data on the CAS server, and they might provide different results when the order of the data changes or the distribution of the data across workers changes, even when options (including the seed) are the same. The listed procedures include the APPLYROWORDER option, which enables you to use a prespecified ordering of the data and a prespecified distribution of observations to threads when you perform an analysis of the data.

The APPLYROWORDER option requires you to use the **partition** action in the **table** action set. The **partition** action distributes the data to threads and workers in your distributed system according to a partition variable that you specify. All rows of data that have the same partition key value reside on the same worker and the same thread in your distributed system. In addition to the partition key, an order-by key enables you to specify the order of the rows of data on each thread of the worker. This key must be unique to each row in order to provide reproducible row ordering.

For example, you can use the CAS procedure as follows to run the **partition** action:

```
proc cas;
   action table.partition /
      table={name="inData",
        groupby={"partitionKey"},
        orderby={"orderbyKey"}
      },
      casout={name="inData2", replace=True};
run;
```

The **partition** action redistributes the data that are specified by the inData table according to the variables that are specified in the **groupby** and **orderby** parameters. The redistributed data are saved in the data table named inData2. The **groupby** parameter specifies the partition key, and the **orderby** parameter specifies the order-by key.

Only procedures that support the APPLYROWORDER option can use the data ordering that the **partition** action provides. If you want the procedure to use this data ordering, specify the APPLYROWORDER option in the PROC statement. For example, the following PROC FOREST statement indicates that you want to use the ordered inData2 data table, making use of the row order that you previously applied:

```
proc forest data=mycas.inData2 applyroworder;
   ...
run;
```

If you have not previously run the **partition** action along with the **groupby** and **orderby** parameters before running a procedure with the APPLYROWORDER option, then the procedure issues a warning and the option has no other effect.

Finally, note that even if you have the same data with the same partition keys and order-by keys, the distribution will be different between systems that have different numbers of threads or different numbers of workers.

References

Amemiya, T. (1976). *Selection of Regressors*. Technical Report 225, Stanford University, Stanford, CA.

Andrew, G., and Gao, J. (2007). "Scalable Training of L_1-Regularized Log-Linear Models." In *Proceedings of the 24th International Conference on Machine Learning*, 33–40. Corvallis, OR: ACM.

Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. (2011). "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers." *Foundations and Trends in Machine Learning* 2:1–122.

Browne, M. W. (1982). "Covariance Structures." In *Topics in Applied Multivariate Analysis*, edited by D. M. Hawkins, 72–141. Cambridge: Cambridge University Press.

Burnham, K. P., and Anderson, D. R. (2002). *Model Selection and Multimodel Inference*. 2nd ed. New York: Springer-Verlag.

Darlington, R. B. (1968). "Multiple Regression in Psychological Research and Practice." *Psychological Bulletin* 69:161–182.

Dennis, J. E., Gay, D. M., and Welsch, R. E. (1981). "An Adaptive Nonlinear Least-Squares Algorithm." *ACM Transactions on Mathematical Software* 7:348–368.

Dennis, J. E., and Mei, H. H. W. (1979). "Two New Unconstrained Optimization Algorithms Which Use Function and Gradient Values." *Journal of Optimization Theory and Applications* 28:453–482.

Edwards, D., and Berry, J. J. (1987). "The Efficiency of Simulation-Based Multiple Comparisons." *Biometrics* 43:913–928.

Efron, B., Hastie, T. J., Johnstone, I. M., and Tibshirani, R. (2004). "Least Angle Regression." *Annals of Statistics* 32:407–499. With discussion.

Eskow, E., and Schnabel, R. B. (1991). "Algorithm 695: Software for a New Modified Cholesky Factorization." *ACM Transactions on Mathematical Software* 17:306–312.

Fan, J., and Li, R. (2001). "Variable Selection via Nonconcave Penalized Likelihood and Its Oracle Properties." *Journal of the American Statistical Association* 96:1348–1360.

Fletcher, R. (1987). *Practical Methods of Optimization*. 2nd ed. Chichester, UK: John Wiley & Sons.

Furnival, G. M., and Wilson, R. W. (1974). "Regression by Leaps and Bounds." *Technometrics* 16:499–511.

Gay, D. M. (1983). "Subroutines for Unconstrained Minimization." *ACM Transactions on Mathematical Software* 9:503–524.

Hastie, T. J., Tibshirani, R., and Tibshirani, R. J. (2020). "Best Subset, Forward Stepwise or Lasso? Analysis and Recommendations Based on Extensive Comparisons." *Statistical Science* 35:579–592.

Hastie, T. J., Tibshirani, R. J., and Friedman, J. H. (2001). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York: Springer-Verlag.

Hocking, R. R. (1976). "The Analysis and Selection of Variables in a Linear Regression." *Biometrics* 32:1–50.

Hsu, J. C. (1992). "The Factor Analytic Approach to Simultaneous Inference in the General Linear Model." *Journal of Computational and Graphical Statistics* 1:151–168.

Hsu, J. C., and Nelson, B. L. (1998). "Multiple Comparisons in the General Linear Model." *Journal of Computational and Graphical Statistics* 7:23–41.

Judge, G. G., Griffiths, W. E., Hill, R. C., and Lee, T.-C. (1980). *The Theory and Practice of Econometrics*. New York: John Wiley & Sons.

Kramer, C. Y. (1956). "Extension of Multiple Range Tests to Group Means with Unequal Numbers of Replications." *Biometrics* 12:307–310.

Lawless, J. F., and Singhal, K. (1978). "Efficient Screening of Nonnormal Regression Models." *Biometrics* 34:318–327.

Liu, H., Yao, T., and Li, R. (2016). "Global Solutions to Folded Concave Penalized Nonconvex Learning." *Annals of Statistics* 44:629–659.

Moré, J. J. (1978). "The Levenberg-Marquardt Algorithm: Implementation and Theory." In *Lecture Notes in Mathematics*, vol. 30, edited by G. A. Watson, 105–116. Berlin: Springer-Verlag.

Moré, J. J., and Sorensen, D. C. (1983). "Computing a Trust-Region Step." *SIAM Journal on Scientific and Statistical Computing* 4:553–572.

Nelson, P. R. (1982). "Exact Critical Points for the Analysis of Means." *Communications in Statistics—Theory and Methods* 11:699–709.

Nelson, P. R. (1991). "Numerical Evaluation of Multivariate Normal Integrals with Correlations $\rho_{lj} = -\alpha_l \alpha_j$." In *Frontiers of Statistical Scientific Theory and Industrial Applications: Proceedings of the ICOSCO I Conference*, edited by A. Öztürk and E. C. van der Meulen, 97–114. Columbus, OH: American Sciences Press.

Nelson, P. R. (1993). "Additional Uses for the Analysis of Means and Extended Tables of Critical Values." *Technometrics* 35:61–71.

Nesterov, Y. (2013). "Gradient Methods for Minimizing Composite Objective Function." *Mathematical Programming* 140:125–161.

Osborne, M. R., Presnell, B., and Turlach, B. A. (2000). "A New Approach to Variable Selection in Least Squares Problems." *IMA Journal of Numerical Analysis* 20:389–404.

Ott, E. R. (1967). "Analysis of Means: A Graphical Procedure." *Industrial Quality Control* 24:101–109. Reprinted in *Journal of Quality Technology* 15 (1983): 10–18.

Schabenberger, O., Gregoire, T. G., and Kong, F. (2000). "Collections of Simple Effects and Their Relationship to Main Effects and Interactions in Factorials." *American Statistician* 54:210–214.

Searle, S. R. (1971). *Linear Models*. New York: John Wiley & Sons.

Stein, C. (1960). "Multiple Regression." In *Contributions to Probability and Statistics: Essays in Honor of Harold Hotelling*, edited by I. Olkin, S. G. Ghurye, W. Hoeffding, W. G. Madow, and H. B. Mann, 264–305. Stanford, CA: Stanford University Press.

Tibshirani, R. (1996). "Regression Shrinkage and Selection via the Lasso." *Journal of the Royal Statistical Society, Series B* 58:267–288.

Westfall, P. H., Tobias, R. D., Rom, D., Wolfinger, R. D., and Hochberg, Y. (1999). *Multiple Comparisons and Multiple Tests Using the SAS System*. Cary, NC: SAS Institute Inc.

Westfall, P. H., and Young, S. S. (1993). *Resampling-Based Multiple Testing: Examples and Methods for* p*-Value Adjustment*. New York: John Wiley & Sons.

Winer, B. J. (1971). *Statistical Principles in Experimental Design*. 2nd ed. New York: McGraw-Hill.

Yuan, M., and Lin, L. (2006). "Model Selection and Estimation in Regression with Grouped Variables." *Journal of the Royal Statistical Society, Series B* 68:49–67.

Zhang, C.-H. (2010). "Nearly Unbiased Variable Selection under Minimax Concave Penalty." *Annals of Statistics* 38:894–942.

Zou, H. (2006). "The Adaptive Lasso and Its Oracle Properties." *Journal of the American Statistical Association* 101:1418–1429.

Zou, H., and Hastie, T. (2005). "Regularization and Variable Selection via the Elastic Net." *Journal of the Royal Statistical Society, Series B* 67:301–320.

Zou, H., and Zhang, H. H. (2009). "On the Adaptive Elastic-Net with a Diverging Number of Parameters." *Annals of Statistics* 37:1733–1751.

Chapter 3
ASSESS Procedure

Contents

Overview: ASSESS Procedure **112**
PROC ASSESS Features 112
Using CAS Sessions and CAS Engine Librefs 112
Getting Started: ASSESS Procedure **113**
Syntax: ASSESS Procedure **115**
PROC ASSESS Statement 116
BY Statement 118
FITSTAT Statement 118
FREQ Statement 119
INPUT Statement 119
TARGET Statement 119
WEIGHT Statement 120
Details: ASSESS Procedure **120**
Lift Information 120
ROC Information 121
Fit Statistics 122
Displayed Output 124
Fit Statistics 124
Lift Information 124
ROC Information 124
Lift Regression Information 124
ODS Table Names 124
Examples: ASSESS Procedure **125**
Example 3.1: Assess a Model That Has a Binary Target 125
References **129**

Overview: ASSESS Procedure

The ASSESS procedure assesses and compares supervised learning models in SAS Viya. For a supervised learning model that has a nominal target, the ASSESS procedure produces lift information and receiver operating characteristic (ROC) information. For a regression model, the ASSESS procedure performs a quantile binning of the predictions and then returns the summary statistics of the response variable for each bin. PROC ASSESS also calculates fit statistics such as average square error, mean square logarithmic error, mean absolute error, mean consequential error, and multiclass log loss.

The expected input to the ASSESS procedure is score data from supervised learning models.

PROC ASSESS Features

The ASSESS procedure can be used to select a champion model from several models or to evaluate a certain model during the model assessment phase. The following list summarizes some basic features of PROC ASSESS:

- executes analysis in parallel on multiple machines
- is multithreaded
- calculates lift and gain in each bucket based on percentile
- calculates ROC information based on cut size
- calculates popular fit statistics for regression and classification models
- performs analysis for each partition

Using CAS Sessions and CAS Engine Librefs

SAS Cloud Analytic Services (CAS) is the analytic server and associated cloud services in SAS Viya. This section describes how to create a CAS session and set up a CAS engine libref that you can use to connect to the CAS session. It assumes that you have a CAS server already available; contact your system administrator if you need help starting and terminating a server. This CAS server is identified by specifying the host on which it runs and the port on which it listens for communications. To simplify your interactions with this CAS server, the host information and port information for the server are stored as SAS option values that are retrieved automatically whenever this CAS server needs to be accessed. You can examine the host and port values for the server at your site by using the following statements:

```
proc options option=(CASHOST CASPORT);
run;
```

In addition to starting a CAS server, your system administrator might also have created a CAS session and a CAS engine libref for your use. You can define your own sessions and CAS engine librefs that connect to the CAS server as shown in the following statements:

```
cas mysess;
libname mylib cas sessref=mysess;
```

The CAS statement creates the CAS session named mysess, and the LIBNAME statement creates the mylib CAS engine libref that you use to connect to this session. It is not necessary to explicitly name the CASHOST and CASPORT of the CAS server in the CAS statement, because these values are retrieved from the corresponding SAS option values.

If you have created the mysess session, you can terminate it by using the TERMINATE option in the CAS statement as follows:

```
cas mysess terminate;
```

For more information about the CAS and LIBNAME statements, see the section "Introduction to Shared Concepts" on page 6 in Chapter 2, "Shared Concepts."

Getting Started: ASSESS Procedure

(View the complete code for this example at https://github.com/sascommunities/doc-supplement-casstat/tree/main/Examples/assess_gs.sas.)

NOTE: Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections "Using CAS Sessions and CAS Engine Librefs" on page 7 and "Loading a SAS Data Set onto a CAS Server" on page 7 in Chapter 2, "Shared Concepts."

This example demonstrates how to use the ASSESS procedure to compute lift regression information and fit statistics.

The following SAS DATA step creates the SAS data table mylib.score, which is similar to score data from a linear regression, and loads the table into your CAS session:

```
data mylib.score;
   input _PartInd_ good p_good;
   datalines;
0 0.8224 0.7590
0 0.6538 0.4632
0 0.7693 0.7069
0 0.7491 0.7087
0 0.7779 0.7209
1 0.7161 0.8389
1 0.6779 0.6209
1 0.6392 0.6077
1 0.8090 0.9096
1 0.6064 0.7355
;
```

These statements assume that your CAS engine libref is named mylib, but you can substitute any appropriately defined CAS engine libref.

In this data table, the variable good is the ground truth and the variable p_good is the prediction from modeling. The variable _PartInd_ assigns each observation to one of two partitions for the analysis.

The following statements call the ASSESS procedure to compute lift regression information and fit statistics.

```
proc assess data=mylib.score nbins=2;
   var p_good;
   target good;
   by _PartInd_;
run;
```

The VAR statement specifies that the variable p_good should be analyzed in the model assessment. The TARGET statement specifies that the variable good is the response variable. The BY statement specifies that PROC ASSESS should perform the analysis separately for each value of _PartInd_.

Figure 3.1 shows the minimum, maximum, and mean of the actual and predicted values for _PartInd_=0.

Figure 3.1 Lift Regression Information for the First Partition

The ASSESS Procedure

PartInd=0

Lift Regression Information

Variable	Depth	Number of Observations	Predicted Minimum	Predicted Maximum	Predicted Mean	Actual Minimum	Actual Maximum	Actual Mean
p_good	0	0	.	.	.	.	.	.
	50	3	0.708700	0.759000	0.729533	0.749100	0.822400	0.783133
	100	2	0.463200	0.706900	0.585050	0.653800	0.769300	0.711550

Figure 3.2 shows the error metrics of the interval target for _PartInd_=0.

Figure 3.2 Fit Statistics for the First Partition

PartInd=0

Fit Statistics

Number of Observations	Divisor of Average	Squared Error Average	Root Average	Absolute Error Mean	Root Mean	Squared Logarithmic Error Mean	Root Mean	R-Square
5	5	0.009825	0.099119	0.082760	0.287680	0.003829	0.061879	-2.15970

Figure 3.3 shows the minimum, maximum, and mean of the actual and predicted values for _PartInd_=1.

Figure 3.3 Lift Regression Information for the Second Partition

The ASSESS Procedure

PartInd=1

Lift Regression Information

Variable	Depth	Number of Observations	Predicted Minimum	Predicted Maximum	Predicted Mean	Actual Minimum	Actual Maximum	Actual Mean
p_good	0	0	.	.	.	.	.	.
	50	3	0.735500	0.909600	0.828000	0.606400	0.809000	0.710500
	100	2	0.607700	0.620900	0.614300	0.639200	0.677900	0.658550

Figure 3.4 shows the error metrics for the interval target for _PartInd_=1.

Figure 3.4 Fit Statistics for the Second Partition

PartInd=1

Fit Statistics

Number of Observations	Divisor of Average	Squared Error Average	Squared Error Root Average	Absolute Error Mean	Absolute Error Root Mean	Squared Logarithmic Error Mean	Squared Logarithmic Error Root Mean	R-Square
5	5	0.009222	0.096029	0.088200	0.296985	0.003050	0.055230	-0.87754

Syntax: ASSESS Procedure

The following statements are available in the ASSESS procedure:

PROC ASSESS < *options* > **;**
BY *variable* **;**
FITSTAT PVAR=*variables* / **PEVENT=***"event-list"* < **DLM=***"character"* > **;**
FREQ *variable* **;**
INPUT *variable* **;**
TARGET *variables* < */options* > **;**
WEIGHT *variable* **;**

The PROC ASSESS, INPUT, and TARGET statements are required.

The following sections describe the PROC ASSESS statement and then describe the other statements in alphabetical order.

PROC ASSESS Statement

PROC ASSESS < *options* > **;**

The PROC ASSESS statement invokes the ASSESS procedure.

You can specify the following *options*:

DATA=*libref.data-table*
names the input data table for PROC ASSESS to use. The default is the most recently created data table. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to the data, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 112.

data-table specifies the name of the input data table.

FITSTATOUT=*libref.data-table*
specifies the name of the fit statistics results table. *libref.data-table* is a two-level name, where *libref* refers to the library, and *data-table* specifies the name of the output data table. For more information about this two-level name, see the DATA= option and the section "Using CAS Sessions and CAS Engine Librefs" on page 112.

If you specify this option, all ODS tables are suppressed. This option is valid only if you also include a FITSTAT statement.

LIFTOUT=*libref.data-table*
specifies the name of the lift results table. *libref.data-table* is a two-level name, where *libref* refers to the library, and *data-table* specifies the name of the output data table. For more information about this two-level name, see the DATA= option and the section "Using CAS Sessions and CAS Engine Librefs" on page 112.

If you specify this option, all ODS tables are suppressed.

MAXITER=*integer*
specifies the maximum number of iterations for the percentile algorithm, where *integer* must be an integer greater than or equal to 1. By default, MAXITER=5 times the value of the NBINS= option.

METHOD=EXACT | ITERATIVE
specifies the analytic method to use to compute the percentile for binning in the lift analysis.

You can specify the following values:

EXACT uses smart technology to perform an exact count to compute the percentile without global sorting in the distributed environment.

ITERATIVE uses the iterative approach to approximate the percentile in the distributed environment.

If you specify METHOD=EXACT, the MAXITER= option is ignored. By default, METHOD=ITERATIVE.

NBINS=*integer*
specifies the number of bins to use in the lift calculation, where *integer* must be an integer greater than or equal to 2.

By default, NBINS=20.

NCUTS=*integer*
specifies the number of cuts to use in the ROC calculation, where *integer* must be an integer greater than or equal to 2. The inverse of *integer* represents the step size of the quantile. For example, NCUTS=10 generates 10 intervals for ROC analysis, with each step size 0.1. You cannot specify this option when the response variable that is specified in the TARGET statement is interval.

By default, NCUTS=100.

NOMISSINGTARGET
excludes the observations that have missing values of a nominal target from assessment analysis.

NTHREADS=*number*
specifies the number of threads to use in the computation, where *number* must be an integer between 1 and 64, inclusive. The default value is the number of CPUs available in the machine.

QUANTILEBIN
improves binning allocations in the lift analysis in certain situations where data are not evenly distributed, with spikes in some intervals.

ROCOUT=*libref.data-table*
specifies the name of the ROC results table. *libref.data-table* is a two-level name, where *libref* refers to the library, and *data-table* specifies the name of the output data table. For more information about this two-level name, see the DATA= option and the section "Using CAS Sessions and CAS Engine Librefs" on page 112.

If you specify this option, all ODS tables are suppressed. This option is valid only for a nominal target (LEVEL=NOMINAL option in the TARGET statement).

USERCUTOFF=*ratio*
specifies a user-defined threshold for events, where *ratio* must be between 0 and 1, exclusive.

If you specify this option, the ROC table includes only one row, which contains the user-defined cutoff value, and the NCUTS= option is ignored.

USEDATARANGE(NROCBINS=*number* **LB=***number* **UB=***number***)**
computes the ROC-related statistics on the basis of the specified range of the input data.

You can specify the following additional suboptions:

NROCBINS=*number*
specifies the number of bins to apply when you compute ROC-related statistics on the basis of the range of the input data.

By default, NROCBINS=5000.

LB=*number*
specifies the lower-bound value of the range of the input data to apply when you compute ROC-related statistics, where *number* must be between 0 and 1, inclusive.

By default, LB=0.

UB=*number*
specifies the upper-bound value of the range of the input data to apply when you compute ROC-related statistics, where *number* must be between 0 and 1, inclusive.

By default, UB=1.

If neither the LB= option nor UB= option is specified, the ROC-related statistics are computed on the basis of the range of the input data.

BY Statement

BY *variable* **;**

The BY statement specifies one partition *variable* to be used in the predictive model. If you specify the BY statement, PROC ASSESS performs the analysis separately for each value of the BY variable. If you do not specify the BY statement, PROC ASSESS treats the entire input data as one group for analysis.

FITSTAT Statement

FITSTAT PVAR=*variables* / **PEVENT=***"event-list"* < **DLM=***"character"*> **;**

The FITSTAT statement reports the following error metrics for a nominal target: average square error, divisor of average square error, root average square error, mean consequential error, and multiclass log loss. (For an interval target, PROC ASSESS creates the following metrics by default, without your specifying the FITSTAT statement: average square error, divisor of average square error, root average square error, mean absolute error, root mean absolute error, mean square logarithmic error, and root mean square logarithmic error.)

You must specify the following options:

PVAR=*variables*
specifies the posterior probability for each level in model prediction except the variable that you specify in the INPUT statement. Duplicate *variables* are not allowed. If you specify the *variable* that is specified in the INPUT statement, it is ignored.

PEVENT="*event-list***"**
specifies the events that correspond to each *variable* in the PVAR= option. The "*event-list*" cannot include the event that corresponds to the variable in the TARGET statement. You must specify the PEVENT= and PVAR= options one after the other, in this order, and the same number of times. The maximum length of the "*event-list*" is 5,000 characters.

You can also specify the following option:

DELIMITER="*character***"**

DLM="*character***"**

specifies the delimiter that is used to separate events that you specify in the PEVENT= option. You use a delimiter when event names contain embedded spaces (for example, "Fuel Oil") or special characters (for example, ";" or "*"). You must specify the quotation marks around the "*character*". You can specify only one delimiter; combinations of delimiters are not supported. The valid values of the DELIMITER= option are " " (space), ";" (semicolon), "*" (asterisk), "." (period), and "," (comma).

By default, DELIMITER=" " (space).

If you specify a delimiter other than " " (space), PROC ASSESS treats any spaces, including empty spaces, as valid characters. You are responsible for mapping events correctly in order to obtain correct results.

FREQ Statement

FREQ *variable* **;**

The *variable* in the FREQ statement identifies a numeric variable in the input data table that contains the frequency of occurrence of each observation. PROC ASSESS treats each observation as if it appears f times, where f is the value of the FREQ *variable* for the observation. If f is not an integer, it is truncated to an integer. If f is less than 1 or missing, the observation is not used in the analysis. When you omit the FREQ statement, each observation is assigned a frequency of 1.

INPUT Statement

INPUT *variable* **;**

VAR *variable* **;**

The INPUT statement specifies the name of one *variable* to be analyzed in model assessment. When the variable specified in the TARGET statement is nominal, *variable* is the posterior probability of the event to be analyzed; when the variable specified in the TARGET statement is interval, *variable* is the predicted value.

TARGET Statement

TARGET *variable* < */options* > **;**

The TARGET statement specifies the response *variable* in supervised learning. You can specify the following *options*:

EVENT=“*string*”

specifies the formatted value of the response *variable* that represents the event. For an interval target (LEVEL=INTERVAL), the EVENT= option is ignored. For a nominal target (LEVEL=NOMINAL), the EVENT= option is required.

LEVEL=INTERVAL | NOMINAL
specifies the *variable* type.

You can specify the following values:

INTERVAL specifies that the response *variable* is interval, which must be numeric.

NOMINAL specifies that the response *variable* is nominal, also known as a classification variable, which can be numeric or character.

By default, LEVEL=INTERVAL.

WEIGHT Statement

WEIGHT *variable* **;**

The *variable* in the WEIGHT statement is used as a weight to perform a weighted analysis of ROC information and fit statistics. If specified, the WEIGHT statement is ignored for lift analysis. If you analyze ROC information and fit statistics, nonpositive or missing weights are ignored in the analysis. When the WEIGHT statement is omitted, all observations that are used in the analysis are assigned a weight of 1.

Details: ASSESS Procedure

Lift Information

In the lift information table, the number of events are first allocated to each bin based on a percentile analysis of the probability of the event. In distributed mode, the percentile analysis is based on a heuristic, so it is an estimation of the precise percentile cutoff points that are used for binning. Consequently, the resulting cutoff points might be different from the cutoff points that are based on sorting the whole data set. This difference is more likely to happen when the data set is small. If specified, the weight variable is ignored for lift analysis.

Events are allocated to bins in two ways in order to perform the analysis. One way is the baseline case, which allocates events evenly to bins with the assumption that no analytic models are applied. The second way is the predicted case, which allocates events to each bin based on probabilities from modeling. In each case, the following metrics are calculated independently:

- captured response percentage, which is the number of events in each bin divided by the total number of events
- cumulative captured response percentage, which accumulates the captured response percentages over bins
- response percentage, which is the number of events divided by the number of observations in each bin
- cumulative response percentage, which accumulates the response percentages over bins

Lift is defined as the ratio of the captured response percentage from the predicted case to the captured response percentage from the baseline case. Gain is defined as the ratio of the cumulative response percentage from the predicted case to the cumulative response percentage from the baseline case, minus one.

ROC Information

In the receiver operating characteristic (ROC) information table, the confusion matrix is calculated on the basis of the event in each cutoff point. If weight or frequency is specified, it is included in the confusion matrix computation. More terminology and the derivation of a confusion matrix are available in Fawcett (2006), Powers (2011), and Derby (2013).

The following notation is necessary for discussion of ROC information:

- m is the number of cutoff points
- n is the number of observations
- N is the sum of observation frequencies with weight in the data
- w_i is the frequency of observation i and r_i is the weight of observation i, so

$$N = \sum_{i=1}^{n} w_i r_i$$

- a_k is true positive at cutoff point k, $k \in [0, m-1]$
- b_k is false positive at cutoff point k, $k \in [0, m-1]$
- c_k is false negative at cutoff point k, $k \in [0, m-1]$

The definitions of C (area under the curve), Gini, gamma, and tau are as follows:

$$\text{C} = \frac{\mu + \frac{\theta}{2}}{\rho}$$

$$\text{Gini} = \frac{\mu - \omega}{\rho}$$

$$\text{gamma} = \frac{\mu - \omega}{\mu + \omega}$$

$$\text{tau} = \frac{\mu - \omega}{\frac{N}{2}(N-1)}$$

where:

$$\theta = \sum_{k=1}^{m} ((a_{k-1} - a_k)(b_{k-1} - b_k))$$

$$\mu = \sum_{k=2}^{m} \left((a_{k-1} - a_k) \sum_{j=1}^{k-1} (b_{j-1} - b_j) \right)$$

$$\omega = \sum_{k=1}^{m-1} \left((a_{k-1} - a_k) \sum_{j=k+1}^{m} (b_{j-1} - b_j) \right)$$

$$\rho = a_0 b_0, a_m = 0 \text{ and } b_m = 0$$

The F0.5 score at cutoff point k is extended from F1 and is defined as follows:

$$(1 + \beta^2) \frac{pr}{\beta^2 p + r}$$

where $p = \frac{a_k}{a_k + b_k}$, $r = \frac{a_k}{a_k + c_k}$, and $\beta = 0.5$. For more information about F0.5, see Kaggle Inc. (2015). The Kolmogorov-Smirnov chart and other metrics are available in Derby (2013).

Fit Statistics

The following notation is necessary for discussion of fit statistics:

- n is the number of observations
- r is the number of levels for a nominal target
- N is the sum of observation frequencies in the data
- M is the sum of observation frequencies with weight in the data
- w_i is the frequency of observation i, so

$$N = \sum_{i=1}^{n} w_i$$

- v_i is the weight of observation i, so

$$M = \sum_{i=1}^{n} w_i v_i$$

- y_i is the actual target value of observation i
- $\hat{y}_i$ is the predicted target value of observation i
- t_i is the level from the target in observation i
- $\hat{t}_i$ is the predicted level from the target in observation i
- $y_{i,j}$ is 1 if observation i is assigned to target level j; otherwise it is 0
- $p_{i,j}$ is the predicted probability that observation i is assigned to target level j

For interval targets, PROC ASSESS calculates the following metrics:

- average square error

$$\text{ASE} = \frac{1}{N}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 w_i v_i$$

- root average square error

$$\text{RASE} = \sqrt{\text{ASE}}$$

- mean square logarithmic error

$$\text{MSLE} = \frac{1}{N}\sum_{i=1}^{n}\left(\log(\hat{y}_i + 1) - \log(y_i + 1)\right)^2 w_i v_i$$

- mean absolute error

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{n}|y_i - \hat{y}_i| w_i v_i$$

For nominal targets, PROC ASSESS calculates the following metrics:

- average square error

$$\text{ASE} = \frac{1}{rN}\sum_{i=1}^{n}\sum_{j=1}^{r}(y_{i,j} - p_{i,j})^2 w_i v_i$$

- root average square error

$$\text{RASE} = \sqrt{\text{ASE}}$$

- mean consequential error (misclassification)

$$\text{MCE} = \frac{1}{M}\sum_{t_i \neq \hat{t}_i} w_i v_i$$

- multiclass log loss

$$\text{logloss} = -\frac{1}{N}\sum_{i=1}^{n}\sum_{j=1}^{r} y_{i,j}\log(p_{i,j}) w_i v_i$$

Displayed Output

The following sections describe the output that PROC ASSESS produces. The output is organized into various tables, which are discussed in the order of their appearance.

Fit Statistics

The "Fit Statistics" table contains the following error metrics for a nominal target if you specify the FITSTAT statement: average square error, divisor of average square error, root average square error, mean consequential error, and multiclass log loss.

For an interval target, the following metrics are created without specifying the FITSTAT statement: average square error, divisor of average square error, root average square error, mean absolute error, root mean absolute error, mean square logarithmic error, and root mean square logarithmic error.

Lift Information

The "Lift Information" table contains information about the lift and gain of each partition for a nominal target.

ROC Information

The "ROC Information" table contains information about the receiver operating characteristic (ROC) of each partition for a nominal target.

Lift Regression Information

The "Lift Regression Information" table contains information about lift regression for each partition for an interval target.

ODS Table Names

Each table created by the ASSESS procedure has a name associated with it, and you must use this name to refer to the table when you use ODS statements. The names of each table and a short description of the contents are listed in Table 3.1.

Table 3.1 ODS Tables Produced by PROC ASSESS

Table Name	Description	Statement	Option
FitStat	Fit statistics	FITSTAT	PVAR= and PEVENT= for a nominal target; default for interval target
LIFTInfo	Lift information	PROC ASSESS	Default for nominal target
LIFTRegInfo	Lift regression information	PROC ASSESS	Default for interval target

Table 3.1 *continued*

Table Name	Description	Statement	Option
ROCInfo	ROC information	PROC ASSESS	Default for nominal target

Examples: ASSESS Procedure

The examples in this chapter are available in the GitHub repository located at https://github.com/sascommunities/doc-supplement-casstat.

NOTE: Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections "Using CAS Sessions and CAS Engine Librefs" on page 7 and "Loading a SAS Data Set onto a CAS Server" on page 7 in Chapter 2, "Shared Concepts."

Example 3.1: Assess a Model That Has a Binary Target

(View the complete code for this example (assess_ex1.sas) in the example repository.)

The following example demonstrates how to use the ASSESS procedure to compute ROC information, lift information, and fit statistics for a model that has a nominal target.

The following DATA step creates a score data table from a previously trained model. For example, the model could be a tree-based model, a neural network, or a support vector machine.

```
data mylib.score2;
   length good_bad $4;
   input _PartInd_ good_bad p_good p_bad;
   datalines;
0 good 0.6675 0.3325
0 good 0.5189 0.4811
0 good 0.6852 0.3148
0 bad  0.0615 0.9385
0 bad  0.3053 0.6947
0 bad  0.6684 0.3316
0 good 0.6422 0.3578
0 good 0.6752 0.3248
0 good 0.5396 0.4604
0 good 0.4983 0.5017
0 bad  0.1916 0.8084
0 good 0.5722 0.4278
0 good 0.7099 0.2901
0 good 0.4642 0.5358
0 good 0.4863 0.5137
1 bad  0.4942 0.5058
1 bad  0.4863 0.5137
```

```
1 bad  0.4942 0.5058
1 good 0.6118 0.3882
1 good 0.5375 0.4625
1 good 0.8132 0.1868
1 good 0.6914 0.3086
1 good 0.5700 0.4300
1 good 0.8189 0.1811
1 good 0.2614 0.7386
1 good 0.1910 0.8090
1 good 0.5129 0.4871
1 good 0.8417 0.1583
1 good 0.5500 0.4500
;
```

The following PROC ASSESS call uses five bins to do lift analysis and uses 0.2 as the incremental step size for ROC analysis:

```
proc assess data=mylib.score2 ncuts=5 nbins=5;
   var p_good;
   target good_bad / event="good" level=nominal;
   fitstat pvar=p_bad / pevent="bad" ;
   by _PartInd_;
run;
```

Output 3.1.1 shows the lift and gain for each bin in the first partition.

Output 3.1.1 Lift Regression Information for the First Partition

The ASSESS Procedure

PartInd=0

Lift Information

Variable	Event	Depth	Number of Observations	Number of Events	Captured Response Percent Individual	Captured Response Percent Cumulative	Lift Individual	Lift Cumulative	Cumulative Best
p_good	good	0	0	0	0.00	0.00	.	.	.
		20	3	3	27.27	27.27	1.363636	1.363636	1.363636
		40	3	2	18.18	45.45	0.909091	1.136364	1.363636
		60	3	3	27.27	72.73	1.363636	1.212121	1.363636
		80	3	3	27.27	100.00	1.363636	1.250000	1.250000
		100	3	0	0.00	100.00	0	1	1

Lift Information

Response Percent Individual	Response Percent Cumulative	Gain Individual	Gain Best
.	.	.	.
100.00	100.00	0.363636	0.363636
66.67	83.33	0.136364	0.363636
100.00	88.89	0.212121	0.363636
100.00	91.67	0.250000	0.250000
0.00	73.33	0	0

Output 3.1.2 shows the ROC information, including the confusion matrix and its derivations, for each bin in the first partition.

Output 3.1.2 ROC Information for the First Partition

PartInd=0

ROC Information

Variable	Event	Cutoff	TP	FP	FN	TN	FPR	FDR	TPR	FNR	TNR	ACC	KS	Youden Index	F1 Score
p_good	good	0	11	4	0	0	1	0.266667	1	0	0	0.733333	0	0	0.846154
		0.200000	11	2	0	2	0.500000	0.153846	1	0	0.500000	0.866667	0	0.500000	0.916667
		0.400000	11	1	0	3	0.250000	0.083333	1	0	0.750000	0.933333	1	0.750000	0.956522
		0.600000	5	1	6	3	0.250000	0.166667	0.454545	0.545455	0.750000	0.533333	0	0.204545	0.588235
		0.800000	0	0	11	4	0	.	0	1	1	0.266667	0	0	0
		1	0	0	11	4	0	.	0	1	1	0.266667	0	0	0

ROC Information

F0.5 Score	AUC	Gini	Gamma	Tau	Misclassification (Event)
0.774648	0.806818	0.613636	0.692308	0.257143	0.266667
0.873016	0.806818	0.613636	0.692308	0.257143	0.133333
0.932203	0.806818	0.613636	0.692308	0.257143	0.066667
0.714286	0.806818	0.613636	0.692308	0.257143	0.466667
0	0.806818	0.613636	0.692308	0.257143	0.733333
0	0.806818	0.613636	0.692308	0.257143	0.733333

Output 3.1.3 shows the error metrics of the nominal target in the first partition.

Output 3.1.3 Fit Statistics for the First Partition

PartInd=0

Fit Statistics

		Squared Error			
Number of Observations	Divisor of Average	Average	Root Average	Mean Consequential Error	Multiclass Log Loss
15	15	0.169127	0.411250	0.266667	0.514728

Output 3.1.4 shows the lift and gain for each bin in the second partition.

Output 3.1.4 Lift Regression Information for the Second Partition

The ASSESS Procedure

PartInd=1

Lift Information

Variable	Event	Depth	Number of Observations	Number of Events	Captured Response Percent Individual	Captured Response Percent Cumulative	Lift Individual	Lift Cumulative	Lift Cumulative Best
p_good	good	0	0	0	0.00	0.00	.	.	.
		20	3	3	27.27	27.27	1.363636	1.363636	1.363636
		40	3	3	27.27	54.55	1.363636	1.363636	1.363636
		60	3	3	27.27	81.82	1.363636	1.363636	1.363636
		80	3	0	0.00	81.82	0	1.022727	1.250000
		100	2	2	18.18	100.00	0.909091	1	1

Lift Information

Response Percent Individual	Response Percent Cumulative	Gain Individual	Gain Best
.	.	.	.
100.00	100.00	0.363636	0.363636
100.00	100.00	0.363636	0.363636
100.00	100.00	0.363636	0.363636
0.00	75.00	0.022727	0.250000
100.00	78.57	0	0

Output 3.1.5 shows the ROC information, including the confusion matrix and its derivations, for each bin in the second partition.

Output 3.1.5 ROC Information for the Second Partition

PartInd=1

ROC Information

Variable	Event	Cutoff	TP	FP	FN	TN	FPR	FDR	TPR	FNR	TNR	ACC	KS	Youden Index	F1 Score
p_good	good	0	11	3	0	0	1	0.214286	1	0	0	0.785714	0	0	0.880000
		0.200000	10	3	1	0	1	0.230769	0.909091	0.090909	0	0.714286	0	-0.09091	0.833333
		0.400000	9	3	2	0	1	0.250000	0.818182	0.181818	0	0.642857	0	-0.18182	0.782609
		0.600000	5	0	6	3	0	0	0.454545	0.545455	1	0.571429	1	0.454545	0.625000
		0.800000	3	0	8	3	0	0	0.272727	0.727273	1	0.428571	0	0.272727	0.428571
		1	0	0	11	3	0	.	0	1	1	0.214286	0	0	0

ROC Information

F0.5 Score	AUC	Gini	Gamma	Tau	Misclassification (Event)
0.820896	0.636364	0.272727	0.428571	0.098901	0.214286
0.793651	0.636364	0.272727	0.428571	0.098901	0.285714
0.762712	0.636364	0.272727	0.428571	0.098901	0.357143
0.806452	0.636364	0.272727	0.428571	0.098901	0.428571
0.652174	0.636364	0.272727	0.428571	0.098901	0.571429
0	0.636364	0.272727	0.428571	0.098901	0.785714

Output 3.1.6 shows the error metrics of the nominal target in the second partition.

Output 3.1.6 Fit Statistics for the Second Partition

PartInd=1

Fit Statistics					
		Squared Error			
Number of Observations	Divisor of Average	Average	Root Average	Mean Consequential Error	Multiclass Log Loss
14	14	0.221587	0.470731	0.142857	0.636734

References

Derby, N. (2013). "Managing and Monitoring Statistical Models." In *Proceedings of the SAS Global Forum 2013 Conference*. Cary, NC: SAS Institute Inc. http://support.sas.com/resources/papers/proceedings13/190-2013.pdf.

Fawcett, T. (2006). "An Introduction to ROC Analysis." *Pattern Recognition Letters* 27:861–874. Special issue on ROC analysis in pattern recognition, edited by F. Tortorella.

Kaggle Inc. (2015). "ICDM 2015: Drawbridge Cross-Device Connections." Accessed April 23, 2019. https://www.kaggle.com/c/icdm-2015-drawbridge-cross-device-connections/overview/evaluation.

Powers, D. M. W. (2011). "Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness and Correlation." *Journal of Machine Learning Technologies* 2:37–63.

Chapter 4
BART Procedure

Contents

Overview: BART Procedure **132**
PROC BART Features 133
PROC BART Compared with Other SAS Procedures 133
Using CAS Sessions and CAS Engine Librefs 134
Getting Started: BART Procedure **135**
Syntax: BART Procedure **138**
PROC BART Statement 139
BY Statement 144
CLASS Statement 144
DISPLAY Statement 144
DISPLAYOUT Statement 145
FREQ Statement 146
GAUSSPRIOR Statement 146
MARGIN Statement 147
MARGINDIFF Statement 148
MODEL Statement 148
OUTPUT Statement 149
OUTPUTMARGIN Statement 151
PARTITION Statement 151
SAMPSUMMARY Statement 152
STORE Statement 153
TREEPRIOR Statement 153
Details: BART Procedure **154**
Missing Values 154
BART Model for Normal Data 155
Tree Prior 155
Variance Prior 156
Sampling the Posterior 156
Splitting Rules 158
Continuous Predictors 158
Categorical Predictors 158
BART Model for Binary Data 159
Computational Resources 161
Displayed Output 161
ODS Table Names 163
Examples: BART Procedure **164**

Example 4.1: Storing and Scoring 164
Example 4.2: Probit Model for Binary Data 168
References **170**

Overview: BART Procedure

The BART procedure fits Bayesian additive regression trees (BART) models in SAS Viya for response variables that follow a normal or binary distribution.

A BART model uses a sum-of-trees ensemble to approximate the conditional mean of a response variable (also called a target variable), given a vector of input variables (also called predictor variables). Training a BART model requires you to specify a sum-of-trees ensemble, a likelihood function, and a prior distribution on the model parameters. The BART procedure uses a Bayesian backfitting Markov chain Monte Carlo (MCMC) algorithm to generate posterior samples of the sum-of-trees ensemble and to fit the model. Predictions from the BART model are obtained by averaging predictions from the posterior samples. You can assess uncertainty in the BART model predictions by examining the variability of the posterior samples. PROC BART supports models of normally distributed response variables and probit models of binary response variables.

The sum-of-trees ensemble that is used in a BART model consists of multiple binary decision trees. A decision tree is a type of predictive model that has been developed independently in the statistics community and the machine learning community. The predictor variables that a tree model uses can be categorical or continuous. The set of all possible combinations of the predictor variables is called the predictor space. A single decision tree partitions the predictor space into a set of nonoverlapping segments. In the terminology of the tree metaphor, segments in the partition correspond to the terminal nodes, or leaves, of the tree. The partition is described by a series of splitting rules that are used to route an observation, starting from the root node of the tree, to one of the terminal nodes. The data in a leaf are used to estimate parameters that enable you to predict the value of the response.

In the terminology of machine learning, BART models are predictive models. A predictive model defines a relationship between the input variables and a target variable. The purpose of a predictive model is to predict a target value from the inputs. You create a predictive model by using training data in which the target values are known. You can then apply the model to observations in which the target is unknown. If the predictions fit the new data well, the model is said to generalize well. Good generalization is the primary goal of predictive modeling. BART models use a regularization prior for the sum-of-trees ensemble in order to limit overfitting of the training data and obtain good generalization. The regularization prior controls the complexity of the trees in the ensemble and limits the contribution of an individual tree to the ensemble prediction.

For more information about BART models, see the section “Details: BART Procedure” on page 154.

PROC BART Features

PROC BART has the following basic features:

- trains Bayesian additive regression trees (BART) models for normally distributed response data and probit models for binary response data
- provides a STORE statement to create an analytic store for the model and saves it as a binary object in a data table
- provides a FREQ statement for grouped analysis
- provides an OUTPUT statement to produce a data table that contains predicted values and other observationwise statistics
- provides a PARTITION statement to divide data into training data and test data roles
- provides a TREEPRIOR statement to specify the BART prior parameters for the sum-of-trees ensemble
- provides a GAUSSPRIOR statement to specify the BART prior parameters for a model that is fitted to normally distributed data

Because the BART procedure runs on SAS Cloud Analytic Services (CAS), it also does the following:

- enables you to run on a cluster of machines that distribute the data and the computations
- enables you to run in single-machine mode
- exploits available cores and concurrent threads

PROC BART Compared with Other SAS Procedures

The GRADBOOST and FOREST procedures both fit predictive models by using ensembles of decision trees. Although Bayesian additive regression trees (BART) are also predictive models that consist of tree ensembles, BART models differ substantially from the models that the GRADBOOST and FOREST procedures fit in terms of what you must specify to train the model, how you train the model, and how you obtain predictions from the model.

The GRADBOOST procedure uses a gradient boosting algorithm to construct a sum-of-trees ensemble (Friedman 2001). In the gradient boosting algorithm, trees are added sequentially to the ensemble. The algorithm determines the target value to use for training a new tree by evaluating the gradient of a loss function, given the current ensemble fit. Each tree that the algorithm adds to the ensemble is subject to a "learning rate" that limits the contribution of a single tree to the overall ensemble prediction. You can impose further regularization by applying penalties to the loss function and by constraining the complexity of the individual trees. Predictions from the fitted model are obtained by summing the predictions from each tree in the ensemble and multiplying the total by the learning rate. For more information about the boosting algorithm that PROC GRADBOOST implements, see the section "Boosting" (Chapter 17, *SAS Viya: Machine Learning Procedures*).

The FOREST procedure uses a bagging algorithm to construct a tree ensemble (Breiman 1996, 2001). The bagging algorithm trains each tree in the model on a sample of the input data that it takes with replacement. In addition, when you grow a tree, the input variables that are considered for splitting a node are randomly selected from all available inputs. You can impose further forms of regularization by constraining the complexity of the individual trees. Predictions from the ensemble are obtained by averaging the predictions from each tree. For more information about the bagging algorithm that PROC FOREST implements, see the section "Bagging the Data" (Chapter 11, *SAS Viya: Machine Learning Procedures*).

Bayesian additive regression trees (BART) models consist of samples of a sum-of-trees ensemble that you produce by using a Bayesian backfitting MCMC algorithm. This algorithm uses a form of Gibbs sampling to obtain posterior samples of the sum-of-trees ensemble. In contrast with the gradient boosting or bagging algorithm, when you use this algorithm to train a BART model, trees are not added to the ensemble. Instead, the Gibbs sampler successively modifies each tree in the ensemble. In BART models, you achieve regularization by using a regularization prior that places greater prior probability on simpler tree structures and applies shrinkage to the leaf parameters through the choice of their prior mean and variance. The algorithm obtains predictions from the final BART model by averaging the predictions from the posterior samples of the ensemble, and you can assess uncertainty in the model predictions on the basis of variability of the posterior samples. For more information about sampling from the posterior for a normal response model, see the section "Sampling the Posterior" on page 156.

When you train a model by using either the GRADBOOST or FOREST procedure, you must specify hyperparameters that control the process of training the individual trees. You can tune the hyperparameters manually, or you can use the AUTOTUNE statement to search for a best configuration of hyperparameter values. Training a BART model requires you to specify a prior for the sum-of-trees ensemble and rules for sampling modifications to the tree structures. As discussed by Chipman, George, and McCulloch (2010), the default BART prior can produce models that, in terms of model fit, are comparable to models that you produce by using alternative techniques, including random forests and gradient boosted trees, which are implemented using hyperparameter tuning.

Using CAS Sessions and CAS Engine Librefs

SAS Cloud Analytic Services (CAS) is the analytic server and associated cloud services in SAS Viya. This section describes how to create a CAS session and set up a CAS engine libref that you can use to connect to the CAS session. It assumes that you have a CAS server already available; contact your system administrator if you need help starting and terminating a server. This CAS server is identified by specifying the host on which it runs and the port on which it listens for communications. To simplify your interactions with this CAS server, the host information and port information for the server are stored as SAS option values that are retrieved automatically whenever this CAS server needs to be accessed. You can examine the host and port values for the server at your site by using the following statements:

```
proc options option=(CASHOST CASPORT);
run;
```

In addition to starting a CAS server, your system administrator might also have created a CAS session and a CAS engine libref for your use. You can define your own sessions and CAS engine librefs that connect to the CAS server as shown in the following statements:

```
cas mysess;
libname mylib cas sessref=mysess;
```

The CAS statement creates the CAS session named mysess, and the LIBNAME statement creates the mylib CAS engine libref that you use to connect to this session. It is not necessary to explicitly name the CASHOST and CASPORT of the CAS server in the CAS statement, because these values are retrieved from the corresponding SAS option values.

If you have created the mysess session, you can terminate it by using the TERMINATE option in the CAS statement as follows:

```
cas mysess terminate;
```

For more information about the CAS and LIBNAME statements, see the section "Introduction to Shared Concepts" on page 6 in Chapter 2, "Shared Concepts."

Getting Started: BART Procedure

(View the complete code for this example at https://github.com/sascommunities/doc-supplement-casstat/tree/main/Examples/bartgs1.sas.)

NOTE: Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections "Using CAS Sessions and CAS Engine Librefs" on page 7 and "Loading a SAS Data Set onto a CAS Server" on page 7 in Chapter 2, "Shared Concepts."

This section illustrates some of the basic features of the BART procedure by analyzing data about city-cycle fuel efficiency—that is, the gas mileage of vehicles driven in urban conditions. The data can be downloaded from the UCI Machine Learning Repository (Asuncion and Newman 2007). The following DATA step creates the table autompg in your CAS session:

```
title 'Automobile MPG Study';
data mylib.Autompg;
   input MPG Cylinders Displacement Horsepower Weight
         Acceleration Year Origin Name $35.;
   datalines;
18.0 8 307.0   130.0   3504   12.0   70  1  Chevrolet Chevelle Malibu
15.0 8 350.0   165.0   3693   11.5   70  1  Buick Skylark 320
18.0 8 318.0   150.0   3436   11.0   70  1  Plymouth Satellite
16.0 8 304.0   150.0   3433   12.0   70  1  AMC Rebel SST

   ... more lines ...

44.0 4 97.00   52.00   2130   24.6   82  2  VW Pickup
32.0 4 135.0   84.00   2295   11.6   82  1  Dodge Rampage
28.0 4 120.0   79.00   2625   18.6   82  1  Ford Ranger
31.0 4 119.0   82.00   2720   19.4   82  1  Chevy S-10
;
```

These statements assume that your CAS engine libref is named mylib, but you can substitute any appropriately defined CAS engine libref.

The data table contains nine variables. The response variable MPG is city-cycle mileage per gallon (MPG). Seven predictor variables (Cylinders, Displacement, Horsepower, Weight, Acceleration, Year, and Origin) provide vehicle attributes. Among them, Cylinders, Year, and Origin are categorical variables. The last variable, Name, contains the specific name of each vehicle model.

The following statements fit a Bayesian additive regression trees (BART) model:

```
proc bart data= mylib.autompg seed=246810;
   class cylinders year origin;
   model mpg = cylinders displacement horsepower
               weight acceleration year origin;
run;
```

The MODEL statement is required when you use the BART procedure to fit a model. If you specify a CLASS statement, it must precede the MODEL statement. The output from this analysis is presented in Figure 4.1 through Figure 4.6. Note that this example is run on a single machine with a single chain. If you run PROC BART in a distributed mode, multiple chains are used by default, and the model fit depends on the number of chains that you run.

The "Model Information" in Figure 4.1 summarizes important information about the model that you fit. Information includes the input data source, response (target) variable, bin method, missing rule, simulation size, thinning rate, and random number seed.

Figure 4.1 Model Information

Automobile MPG Study

The BART Procedure

Model Information	
Data Source	AUTOMPG
Target Variable	MPG
Distribution	Normal
Number of Trees	200
Number of Bins	50
Bin Method	Quantiles
Missing Rule	Separate
Burn-In Size	100
Simulation Size	1000
Thinning Rate	1
Random Number Seed	246810

Figure 4.2 displays the "Class Level Information" and "Number of Observations" tables. The "Class Level Information" table lists the levels of the classification variables that you specify in the CLASS statement and the ordering of the levels. The "Number of Observations" table displays the number of observations that are used in the analysis.

Figure 4.2 Class Level Information and Number of Observations

Class Level Information		
Class	**Levels**	**Values**
Cylinders	5	3 4 5 6 8
Year	13	70 71 72 73 74 75 76 77 78 79 80 81 82
Origin	3	1 2 3

Number of Observations Read	398
Number of Observations Used	398

Figure 4.3 displays the "Prior Information" table. The table summarizes the parameters that define the BART prior for the tree structure and the variance of the response variable. The BART prior parameters typically do not require hyperparameter tuning, because the default BART model tends to perform well in terms of model fit, and this performance does not change significantly if there are small changes in the prior parameters.

Figure 4.3 Prior Information

Prior Information	
Depth Zero Node Split Probability	0.95
Node Depth Power	2
Split Operation Probability	0.5
Prune Operation Probability	0.5
Leaf Variance Multiplier	2
Variance Prior Degrees of Freedom	3
Variance Prior Scale	1.522001
Variance Prior Quantile Level	0.9
Initial Variance Estimate	7.517741

Figure 4.4 displays the "Sample Information" table. This table summarizes basic information about the posterior samples of the sum-of-trees ensemble, such as the number of samples saved, the proportion of tree-modifying operations accepted, and information about the size of the trees.

Figure 4.4 Sample Information

Sample Information	
Number of Samples Kept	1000
Proportion of Burn-In Operations Accepted	0.22735
Proportion of Sampling Operations Accepted	0.21987
Minimum Number of Tree Nodes	1
Maximum Number of Tree Nodes	15
Average Number of Tree Nodes	3.5521
Minimum Tree Depth	0
Maximum Tree Depth	5
Minimum Number of Leaves	1
Maximum Number of Leaves	8
Average Number of Leaves	2.27605
Minimum Leaf Size	5
Maximum Leaf Size	398
Average Leaf Size	174.8643

Figure 4.5 displays the fit statistics for the fitted model. In this case, there is only one: the average square error (ASE).

Figure 4.5 Fit Statistics

Fit Statistics	
Average Square Error	2.89472

Finally, PROC BART displays the table shown in Figure 4.6, which shows the amount of time (in seconds) that it took to perform the various tasks in the analysis.

Figure 4.6 Procedure Timing

Task Timing		
Task	**Seconds**	**Percent**
Parsing	0.01	0.08%
Data Preparation	0.05	0.84%
Burn-In Iterations	0.55	9.07%
MCMC Sampling	5.42	89.81%
Miscellaneous	0.01	0.19%
Cleaning Up	0.00	0.01%
Total	6.04	100.00%

Syntax: BART Procedure

The following statements are available in the BART procedure:

PROC BART < *options* > **;**
BY *variables* **;**
CLASS *variable*. . . < *variable* > **;**
DISPLAY < *table-list* > < */options* > **;**
DISPLAYOUT *table-spec-list* < */options* > **;**
FREQ *variable* **;**
GAUSSPRIOR *prior-options* < */option* > **;**
MARGIN '*name*' *variable* **=** '*value*' < . . . *variable* **=** '*value*' > < / **LABEL =** '*label*' > **;**
MARGINDIFF EVENT='*margin-name*' **REF=**'*margin-name*' / < **NAME=***name* > < **LABEL=***label* > **;**
MODEL *response*=*variables* < */model-options* > **;**
OUTPUT OUT=*libref.data-table*
< **SAMPLEPRED** >< **ALPHA=***number* > < **COPYVARS=(***variables***)** >
< *keyword* < **=***name* > >. . . < *keyword* < **=***name* > > **;**
OUTPUTMARGINS *libref.data-table* **;**
PARTITION *partition-options* **;**
SAMPSUMMARY OUT=*libref.data-table* < *keyword*=*name* >. . . < *keyword*=*name* > **;**
STORE < **OUT=** >*libref.data-table* **;**
TREEPRIOR < *prior-options* > **;**

The PROC BART statement and one MODEL statement are required; you can specify only one MODEL statement. The CLASS statement can appear multiple times. If you specify a CLASS statement, it must

precede the MODEL statement. The following sections describe the PROC BART statement and then describe the other statements in alphabetical order.

PROC BART Statement

PROC BART < *options* > **;**

The PROC BART statement invokes the procedure. Table 4.1 summarizes the available options in the PROC BART statement by function. The options are described fully after the table.

Table 4.1 PROC BART Statement Options

Option	Description
Basic Options	
ALPHA=	Specifies the level for the construction of equal-tail credible intervals
APPLYROWORDER	Uses group and order information from the input data table
DATA=	Specifies the input data table
LEAFSIGMAK=	Specifies the value used to determine the prior variance for the leaf parameter
MAXTRAINTIME=	Specifies the maximum time (in seconds) for training
NOCLPRINT	Limits or suppresses the display of classification variable levels
NTREE=	Specifies the number of trees in the sum-of-trees model
RESTORE=	Specifies the input analytic store
SEED=	Sets the seed for pseudorandom number generation
MCMC Options	
NBI=	Specifies the number of burn-in iterations
NDISTCHAINS=	Specifies the number of distributed chains to run
NMC=	Specifies the number of MCMC iterations, excluding the burn-in iterations
NMCDIST=	Specifies the number of MCMC iterations for each chain, excluding the burn-in iterations, when chains are distributed
NTHIN=	Specifies the thinning rate
Memory Options	
MAPINMEM	Stores a mapping of each observation to terminal nodes in memory when the model is trained
TRAININMEM	Stores data in memory when the model is trained
Splitting Options	
ASSIGNMISSING=	Specifies how to handle missing values in predictor variables
BINMETHOD=	Specifies the method to use for binning continuous variables
MINLEAFSIZE=	Specifies the minimum number of observations per leaf
NBINS=	Specifies the number of bins to use for binning continuous variables
ORDERSPLIT=	Specifies the minimum cardinality of a categorical input for which to use splitting rules based on level ordering

ALPHA=*number*

specifies the level for the construction of equal-tail credible limits. The level is 1 – *number*. The value of *number* must be between 0 and 1, exclusive. The level applies to the credible limits for all predictive margins. The level also applies to credible limits for predictions in the output data table unless you specify a different level by using the ALPHA= option in the OUTPUT statement.

By default, ALPHA=0.05.

APPLYROWORDER

uses group and order information from the input data table. You can add this information to the input table by using the **`partition`** action in the **`table`** action set. For more information, see the section "The APPLYROWORDER Option" on page 107 in Chapter 2, "Shared Concepts." The APPLYROWORDER option is applicable to models that require the simulation of latent variables, and by default it is not enabled. If you specify this option but the table does not contain group and order information, the procedure terminates with an error.

ASSIGNMISSING=MACBIG | MACSMALL | NONE | SEPARATE

specifies how to handle missing values during the training and scoring phases.

During the training phase, PROC BART samples a primary splitting rule for a node as described in the section "Splitting Rules" on page 158. During the scoring phase, observations are assigned to a node in a tree according to the primary splitting rule if it can be applied. If the primary rule cannot be applied because of missing values or unknown categorical predictor levels, a default splitting rule is applied that assigns an observation to the larger branch of the split. An unknown level of a categorical predictor is a level that was not encountered in the training data but is encountered during scoring. The default splitting rule enables all data to be scored.

You can specify one of the following values:

MACBIG during the training phase, treats missing values for continuous predictors as the largest machine value and treats missing values for categorical predictors as a separate level. In the scoring phase, observations that have missing continuous predictor values are assigned to the right branch of the split, and observations that have an unknown categorical predictor level are assigned to the larger branch of the split.

MACSMALL during the training phase, treats missing values for continuous predictors as the smallest machine value and treats missing values for categorical predictors as a separate level. In the scoring phase, observations that have missing continuous predictor values are assigned to the left branch of the split, and observations that have an unknown categorical predictor level are assigned to the larger branch of the split.

NONE during the training phase, excludes all observations that have a missing predictor value. In the scoring phase, observations that have missing values or observations whose unknown categorical predictor level is unknown are assigned to the larger branch of the split.

SEPARATE during the training phase, treats missing values for continuous predictors as a separate group and treats missing values for categorical predictors as a separate level. In the training phase, when a split operation is sampled for a continuous predictor and there are observations that have a missing value of the splitting

variable on the node, a primary rule for routing missing values is sampled before the primary splitting rule for nonmissing values is sampled. If a continuous predictor does not have a missing value on the node that you are splitting, a primary rule for routing missing values is not sampled. In the scoring phase, observations that have an unknown categorical predictor level or have a missing continuous predictor value for a node without a primary rule for routing missing values are assigned to the larger branch of the split.

By default, ASSIGNMISSING=SEPARATE.

BINMETHOD=EVEN | QUANTILE
specifies how to bin continuous input variables before training the model. The number of bins that are created is determined by the value of the NBINS= option.

You can specify one of the following values:

EVEN discretizes continuous input variables into fixed-width bins. The width of each bin for a particular variable is calculated by subtracting the smallest value among all observations from the largest value among all observations, and then dividing the result by the number of bins.

QUANTILE discretizes continuous input variables into bins according to their quantile. The width of the bins for a particular variable is not fixed, but the number of observations in the various bins is approximately equal.

By default, BINMETHOD=QUANTILE.

DATA=*libref.data-table*
names the input data table for PROC BART to use. The default is the most recently created data table. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to the data, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 134.

data-table specifies the name of the input data table.

LEAFSIGMAK=*number*
specifies the *number* to be used to determine the prior variance for the leaf parameters. For more information about the tree prior, see the section "Tree Prior" on page 155.

By default, LEAFSIGMAK=2.

MAPINMEM

OBSLEAFMAPINMEM
stores a mapping of each observation to terminal nodes, or leaves, in memory when the model is trained. Storing the observation-to-leaf map in memory eliminates the need to route observations through trees when tree-modifying operations are sampled, and it can improve performance. However,

the observation-to-leaf map requires memory allocations that are, in total, proportional to the number of observations times the value of the NTREE= option.

By default, the observation-to-leaf map is not stored in memory.

MAXTRAINTIME=*number*
specifies the maximum time (in seconds) to use for model training.

By default, there is no time limit.

MINLEAFSIZE=*number*
specifies the minimum *number* of observations that each child of a split must contain in the training data in order for the split to be considered.

By default, MINLEAFSIZE=5.

NBI=*number*
specifies the *number* of burn-in iterations to perform before the procedure starts to save samples for prediction.

By default, NBI=100.

NBINS=*number*
specifies the *number* of bins to use for binning the continuous input variables.

By default, NBINS=50.

NDISTCHAINS=*number*
specifies a distributed mode that divides the MCMC sampling in a grid environment. This mode distributes training data to workers so that the specified *number* of workers have a full copy of the training data and run a separate chain. Using this option can reduce the cost of communication over the grid and allows multiple chains to be run in parallel. However, distributed mode requires more memory space for each worker. This option is not applicable when you are in single-machine mode. When NDISTCHAINS=0, a single chain is run, and each worker node is assigned a portion of the training data. Note that the fit of a BART model depends on the number of chains that you run.

By default, the *number* of chains is equal to the number of worker nodes.

NMC=*number*
specifies the *number* of iterations in the main simulation loop. This is the MCMC sample size if NTHIN=1. This option is ignored if you specify the NMCDIST= option.

By default, NMC=1000.

NMCDIST=*number*
specifies the number of MCMC iterations for each chain, excluding the burn-in iterations, when multiple chains are used. This option is not applicable when you are in single-machine mode or when NDISTCHAINS=0.

By default, the number of MCMC iterations is evenly distributed across chains, and the total number is equal to the value of the NMC= option.

NOCLPRINT<=*number*>
suppresses the display of the "Class Level Information" table if you do not specify *number*. If you specify *number*, the values of the classification variables are displayed for only those variables whose number of levels is less than *number*. Specifying a *number* helps reduce the size of the "Class Level Information" table if some classification variables have a large number of levels.

NTHIN=*number*
specifies the thinning rate of the simulation. PROC BART keeps every *number*th simulation sample and discards the rest. All the posterior statistics are calculated using the thinned samples.

By default, NTHIN=1.

NTREE=*number*
specifies the *number* of trees in a sample of the sum-of-trees ensemble.

By default, NTREE=200.

ORDERSPLIT=*number*
specifies the minimum *number* of levels for which a categorical input uses splitting rules according to level ordering. For more information about the sampling of splitting rules, see the section "Splitting Rules" on page 158.

By default, ORDERSPLIT=50.

RESTORE=*libref.data-table*
specifies the name of the analytic store that contains a model that is fitted and stored from a previous analysis. The analytic store is created by a STORE statement from a previous PROC BART call or by a **`store`** parameter that is specified in a previous **`bartGauss`** or **`bartProbit`** action call.

You can use the previously fitted model to either score data or compute predictive margins. For each task, you use the DATA= option to specify the input data table to be used. To score the input data, you specify the OUTPUT statement to create a data table that contains the observationwise predictions and statistics. To compute predictive margins relative to the covariate distribution in the input data, you specify one or more MARGIN statements. When you compute predictive margins, you can also use the MARGINDIFF statement to specify a contrast between predictive margins that is defined by different MARGIN statements. You can use the OUTPUTMARGINS statement to create an output data table that contains the predicted marginal mean for each saved sample of the ensemble.

Because the previously fitted model is stored, you cannot use statements and options specific to training a model. In particular, the CLASS, FREQ, GAUSSPRIOR, MODEL, PARTITION, STORE, and TREEPRIOR statements are not available when you specify the RESTORE= option. When you use this option, you cannot specify both the OUTPUT statement and the MARGIN statement. You must use separate PROC BART calls to score data and compute predictive margins when you use a previously fitted model.

For more information about using a previously fitted model to score new data or compute predictive margins, see "Example 4.1: Storing and Scoring" on page 164.

SEED=*number*

specifies the initial seed for random number generation for model building. The value of *number* must be an integer. If you do not specify a seed or you specify a *number* less than or equal to 0, the seed is generated by reading the time of day from the computer's clock.

TRAININMEM

stores data in memory when the model is trained. Storing the data in memory can improve performance, but it requires memory allocations that are, in total, proportional to the number of observations times the size of a row of predictors.

By default, the data are not stored in memory.

BY Statement

BY *variables* **;**

You can specify a BY statement in PROC BART to obtain separate analyses of observations in groups that are defined by the values of the BY variables. If you specify more than one BY statement, only the last one specified is used. For more information, see the discussion of BY-group processing in the "Grouping Data" section of *SAS Programmers Guide: Essentials*.

CLASS Statement

CLASS *variable. . .* < *variable* > **;**

The CLASS statement names the classification variables to be used as explanatory variables in the analysis.

DISPLAY Statement

DISPLAY < *table-list* > < / *options* > **;**

The DISPLAY statement enables you to specify a list of display tables to display or exclude. This statement is similar to the ODS SELECT, ODS EXCLUDE, and ODS TRACE statements. However, the DISPLAY statement can improve performance when a large number of tables could be generated (such as in BY-group processing). The procedure processes the DISPLAY statement on a CAS server and thus sends only a subset of ODS tables to the SAS client. Because ODS statements are processed on a SAS client, first all the generated display tables are sent to the client, and then the client creates a subset.

If you use both DISPLAY and ODS statements together, the DISPLAY statement takes precedence over the ODS statements. Note that the ODS EXCLUDE statement processes tables that are sent to the client after they have been filtered by the DISPLAY statement. In some cases, it might appear that the ODS EXCLUDE statement is taking precedence because it can further filter the tables. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

You can specify the *table-list* as a list of table names, paths, partial pathnames, and regular expressions.

The table names that you can specify are listed in the section "ODS Table Names" on page 163. A path is a table name that is prefixed with dot-separated grouping information. For example, a

SelectionSummary table that a procedure produces during a selection routine might have the path *Bygroup1.Summary.SelectionSummary*. A partial pathname does not include all groups; for example, *SelectionSummary* and *Summary.SelectionSummary* are partial pathnames for *Bygroup1.Summary.SelectionSummary*.

When you specify a table name or partial pathname, all display tables whose paths end in the specified name are selected for display or exclusion. For example, both *SelectionSummary* and *Summary.SelectionSummary* select *Bygroup1.Summary.SelectionSummary*.

A regular expression is enclosed in forward slashes (/). For example, specifying "/tions/" selects all pathnames that contain the substring "tions"; in particular, the *Bygroup1.Summary.SelectionSummary* table is selected. Specifying "!/tions/" selects all pathnames that do not contain the substring "tions"; in particular, the *Bygroup1.Summary.SelectionSummary* table is not selected.

You can specify the following *options* after a slash (/):

CASESENSITIVE
performs a case-sensitive comparison of table names in the *table-list* to display table names when tables are subsetted for display. To preserve case, you must enclose table names in the *table-list* in quotation marks.

EXCLUDE
displays all display tables except those that you specify in the *table-list*.

EXCLUDEALL
suppresses display of all tables. This option takes precedence over the other options.

TRACE
displays the display table names, labels, and paths.

DISPLAYOUT Statement

DISPLAYOUT *table-spec-list* < / *options* > **;**

The DISPLAYOUT statement enables you to create output tables from your displayed output. This statement is similar to the ODS OUTPUT statement. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

The *table-spec-list* specifies a list of output tables to create. Each entry in the list has either a *key=value* format or a *key* format:

key=value specifies *key* as the ODS table name, path, or partial pathname, and specifies *value* as the output table name.

key specifies *key* as the ODS table name and also as the output table name.

The ODS table names that you can specify are listed in the section "ODS Table Names" on page 163. You cannot specify the ODS table named OutputCasTables in the *table-spec-list*.

Table names and partial pathnames are discussed under the DISPLAY statement. The DISPLAYOUT statement does not support regular expressions.

You can specify the following *options* after a slash (/):

INCLUDEALL
creates output tables for all display tables. The name of the created output table is the same as the name of the corresponding display table. If you specify this option, the *table-spec-list* specification is ignored.

NOREPLACE
does not replace any existing output table of the same name.

REPEATED
replicates all output tables on all nodes.

FREQ Statement

FREQ *variable* **;**

The *variable* in the FREQ statement identifies a numeric variable in the input data table that contains the frequency of occurrence of each observation. PROC BART treats each observation as if it appears f times, where f is the value of the FREQ *variable* for the observation. If f is not an integer, it is truncated to an integer. If f is less than 1 or missing, the observation is not used in the analysis. When you omit the FREQ statement, each observation is assigned a frequency of 1.

GAUSSPRIOR Statement

GAUSSPRIOR *prior-options* < */option* > **;**

The GAUSSPRIOR statement enables you to specify the prior parameters for a model that is fitted to a normally distributed response variable. Bayesian additive regression trees (BART) models for normal data use an inverse chi square prior for the variance parameter. For more information about the BART prior for a normal data, see the section "Variance Prior" on page 156. You can use the following *prior-options* to specify the prior for the variance parameter and the initial variance estimate:

SIGMADF=*number*
specifies the degrees of freedom of the scaled inverse chi-square prior for the variance parameter. The value of *number* must be greater than or equal to 1.

By default, SIGMADF=3.

SIGMALAMBDA=*number*
specifies the scale parameter of the scaled inverse chi-square prior for the variance parameter. The value of *number* must be positive. You cannot specify both this option and the SIGMAQUANTILE= option.

By default, the scale parameter value is determined by the value of the SIGMAQUANTILE= option.

SIGMAQUANTILE=*number*
specifies the quantile level to be used to determine the scale parameter of the scaled inverse chi-square prior for the variance parameter. The value of *number* must be between 0 and 1, exclusive. You cannot specify both this option and the SIGMALAMBDA= option.

By default, SIGMAQUANTILE=0.9.

VAREST=*number*
specifies the initial value for the variance. The value of *number* must be positive.

By default, the initial variance value is determined by a least squares regression of the response variable on the predictor variables.

You can also specify the following *option*:

VARAUTOCORR=*numeric-list*
specifies the autocorrelation lags for the variance parameter.

By default, autocorrelations are computed at lags of 1, 5, 10, and 50.

MARGIN Statement

MARGIN '*name*' *variable*='*value*' < . . . *variable* = '*value*' > < / **LABEL=**'*label*' > **;**

The MARGIN statement computes a predictive margin (Lane and Nelder 1982). A predictive margin is obtained by fixing the *value* of one or more input *variables* and averaging the predicted response values over the distribution of covariates in an input data table.

The elements of the MARGIN statement are as follows:

name
identifies the predictive margin. The *name* is used in output tables, to specify event and reference levels in a MARGINDIFF statement, and to name variables in an output data table that is created by a OUTPUTMARGIN statement.

variable **=** '*value*'
identifies an input *variable* and the *value* that it is set to when you compute the predictive margin. The *variable* can be either continuous or categorical. For a categorical variable, you must specify the *value* in quotes. You can specify multiple input *variables* and the *values* that they are set to when computing the predictive margin.

In addition to the required *name* for the predictive margin, you can provide an optional *label* that is displayed in output tables when you specify the LABEL= option after a slash (/).

If you specify both the MARGIN and MODEL statements, then the predictive margins are computed by averaging the predictions from the fitted model for all observations in the training data. If you specify the MARGIN statement and the RESTORE= option in the PROC BART statement, then the predictive margins are computed by averaging the predictions from the previously fitted model that is saved in the specified analytic store for all observations in the input data table that you specify in the DATA= option in the PROC BART statement. For more information about using a previously fitted model to score and compute predictive margins, see "Example 4.1: Storing and Scoring" on page 164.

MARGINDIFF Statement

MARGINDIFF EVENT='*margin-name*' **REF=**'*margin-name*' / < **LABEL=***label* > < **NAME=***name* > **;**

The MARGINDIFF statement computes the difference between two predictive margins. You must specify the event predictive margin by using the EVENT= option and specify the reference predictive margin by using the REF= option. You identify the event and reference predictive margins by using a *margin-name* that you specify in a MARGIN statement.

You can specify the following options after a slash (/):

LABEL=*label*
labels the difference between predictive margins in output tables.

NAME=*name*
names the difference in predictive margins. A name for the difference is required if you use the OUTPUTMARGINS statement to create an output data table of the predictive margins and their differences.

MODEL Statement

MODEL *response*=*variables* < / *model-options* > **;**

The MODEL statement specifies the *response* (dependent or target) variable and the predictor (independent or explanatory) *variables*. You can specify only one MODEL statement.

You can specify the following *model-options*:

DIST=*keyword*
DISTRIBUTION=*keyword*
specifies the response distribution for the model. The *keywords* and their associated distributions are shown in Table 4.2.

Table 4.2 Built-In Distribution Functions

DISTRIBUTION=	Distribution Function
BINARY | BERNOULLI	Binary
NORMAL | GAUSSIAN | GAUSS	Normal

If you do not specify a response distribution, the BART procedure determines the response distribution as follows. If you specify a GAUSSPRIOR statement, then a Gaussian response model is used. If the response variable is a character variable or included in the CLASS statement, then a binary response model is used. A Gaussian response model is used by default if none of these conditions are met.

OFFSET=*variable*
specifies a *variable* to use as an offset for the binary response model. For more information about the BART model for binary response data, see the section "BART Model for Binary Data" on page 159.

OUTPUT Statement

OUTPUT OUT=*libref.data-table*
< **SAMPLEPRED** >< **ALPHA=***number* > < **COPYVARS=(***variables***)** >
< *keyword* < **=***name* > >. . . < *keyword* < **=***name* > > **;**

The OUTPUT statement creates a data table that contains observationwise statistics that PROC BART computes after fitting the model. To avoid data duplication when you have large data tables, the variables in the input data table are *not* included in the output data table unless you specify them in the COPYVARS= option.

The computation of the output statistics is based on the final fitted estimates. If an error occurs in fitting the model, then the output data table is not created.

You must specify the following option:

OUT=*libref.data-table*
names the output data table for PROC BART to use. You must specify this option before any other options. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to where the data table is to be stored, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 134.

data-table specifies the name of the output data table.

You can also specify the following syntax elements:

ALPHA=*number*
specifies the level for the construction of equal-tail credible limits. The level is 1 – *number*. The value of *number* must be between 0 and 1, exclusive.

By default, ALPHA=0.05.

COPYVAR=*variable*
COPYVARS=(*variables***)**
transfers one or more *variables* from the input data table to the output data table.

INTO< (*cutpoint***) >=<** *name* **>**
names the variable that contains the level of the response into which an observation is classified for binary response models. The default *name* is _INTO_. If the predicted probability of an observation equals or exceeds the *cutpoint*, the observation is classified as an event; otherwise it is classified as a nonevent. You can specify the *cutpoint* value as a number between 0 and 1, exclusive. The default value is 0.5.

LCL<=*name*>
computes the equal-tail lower credible limit. The default *name* is Lcl. You can set the credible limit by specifying the ALPHA= option.

PRED<=*name*>

PREDICTED<=*name*>

P<=*name*>
computes predicted values for the response variable. For observations in which the response variable is missing, the predicted values are computed even though these observations do not affect the model fit. The default *name* is Pred.

RESIDUAL<=*name*>

RESID<=*name*>

R<=*name*>
computes the raw residual, $y - \mu$, where μ is the estimate of the predicted mean. The default *name* is Residual.

ROLE<=*name*>
specifies the numeric variable that indicates the role that each observation plays in fitting the model. The default *name* is _ROLE_. Table 4.3 shows how this variable is interpreted for each observation.

Table 4.3 Role Interpretation

Value	Observation Role
0	Not used
1	Training
3	Testing

If you do not partition the input data by specifying a PARTITION statement, then the role variable value is 1 for observations that are used to fit the model and 0 for observations that are not used.

SAMPLEPRED
produces predicted values and raw residuals for each posterior sample saved for prediction if you specify the PRED and RESIDUAL keywords.

UCL<=*name*>
computes the equal-tail upper credible limit. The default *name* is Ucl. You can set the credible level by specifying the ALPHA= option.

OUTPUTMARGIN Statement

OUTPUTMARGIN *libref.data-table* **;**

The OUTPUTMARGIN statement creates an output data table that contains the predicted marginal means and differences for each posterior sample that is saved for prediction. Each row of the table contains the predicted marginal means and differences for a saved sample of the ensemble. The column name for each predictive margin is the name that you specify in a MARGIN statement, and the column name for each difference is the name that you specify in a MARGINDIFF statement.

libref.data-table is a two-level name, where *libref* refers to the library, and *data-table* specifies the name of the output data table. For more information about this two-level name, see the section "Using CAS Sessions and CAS Engine Librefs" on page 134.

PARTITION Statement

PARTITION *partition-options* **;**

The PARTITION statement specifies how to logically partition observations in the input data set into disjoint subsets for model training and testing. For more information, see the section "Using Validation and Test Data" on page 101 in Chapter 2, "Shared Concepts." Either you can designate a variable in the input data table and a set of formatted values of that variable to determine the role of each observation, or you can specify proportions to use for randomly assigning observations to each role.

You must specify exactly one of the following *partition-options*:

FRACTION(< TEST=*fraction* **>< SEED=***number* **>)**
randomly assigns specified proportions of the observations in the input data table to the roles. You specify the proportions for testing by using the TEST= suboption. The specified *fraction* must be between 0 and 1. The remaining fraction is the proportion of the observations that are assigned to the training role. The SEED= option specifies an integer that is used to start the pseudorandom number generator for random partitioning of data for training, testing, and validation. If you omit the SEED= option, or if the SEED= option value is less than or equal to 0, the seed is generated by reading the time of day from the computer's clock.

ROLE=*variable***(< TEST='***value***' >< TRAIN='***value***' >)**

ROLEVAR=*variable***(< TEST='***value***' >< TRAIN='***value***' >)**
names the *variable* in the input data table whose values are used to assign roles to each observation. You cannot also use this variable as an analysis variable in other statements or options. The TEST= and TRAIN= suboptions specify the formatted values of this variable that are used to assign observation roles. If you omit the TRAIN= suboption, then all observations whose role is not determined by the TEST= suboption are assigned to the training role. If you omit both the TEST= and TRAIN= suboptions, data partitioning is not performed.

SAMPSUMMARY Statement

SAMPSUMMARY OUT=*libref.data-table* < *keyword*=*name*>. . . < *keyword*=*name*> ;

The SAMPSUMMARY statement creates a data table that contains a summary of the sum-of-trees ensemble samples.

You must specify the following option:

OUT=*libref.data-table*
names the output data table for PROC BART to use. You must specify this option before any other options. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to where the data table is to be stored, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 134.

data-table specifies the name of the output data table.

You can also specify the following *keywords*:

AVGNODE< =*name* **>**
names the variable that contains the average number of nodes per tree in the sample. The default *name* is AvgNode.

PROPACCEPTED< =*name* **>**
names the variable that contains the proportion of accepted tree modifications for the sample. The default *name* is PropAccepted.

SAMPSAVED< =*name* **>**
names the variable that contains an indicator for whether the sample is saved for prediction. The default *name* is SampSaved.

VARIANCE< =*name* **>**
names the variable that contains the sampled variance for the normal response model. The default *name* is Variance.

STORE Statement

STORE < **OUT=** >*libref.data-table* **;**

The STORE statement saves the context and results of the statistical analysis. The resulting analytic store has a binary file format that you cannot modify. An example of analytic store technology is to perform a time-consuming analysis and to store its results by using the STORE statement. Later you can perform specific statistical analysis tasks that are based on the saved results of the previous analysis, without having to fit the model again. You can process the contents of the analytic store by using the RESTORE= option, the ASTORE procedure, the **`bartScore`** and **`bartScoreMargin`** actions in the **`bart`** action set, or the **`score`** action in the **`aStore`** action set.

libref.data-table is a two-level name, where *libref* refers to the library, and *data-table* specifies the name of the output data table. For more information about this two-level name, see the section "Using CAS Sessions and CAS Engine Librefs" on page 134.

TREEPRIOR Statement

TREEPRIOR < *prior-options* > **;**

The TREEPRIOR statement enables you to specify the regularization prior for the sum-of-trees ensemble. For more information about the regularization prior, see the section "Tree Prior" on page 155. You can use the following *prior-options* to specify the regularization prior for the sum-of-trees ensemble:

DEPTHBASE=*number*
specifies the base probability for splitting an internal node as a function of its depth from the root. A larger base probability value makes splitting a node more likely. The value of *number* must be between 0 and 1, exclusive.

By default, DEPTHBASE=0.95.

DEPTHPOWER=*number*
specifies the power parameter to be used to compute the probability of splitting an internal node as a function of its depth from the root. A larger depth power value decreases the probability of splitting a node. The value of *number* must be positive.

By default, DEPTHPOWER=2.

PPRUNE=*number*
specifies the probability of sampling the operation of pruning a terminal node for the tree sampling algorithm. The value of *number* must be between 0 and 1, exclusive, and if you specify both the PSPLIT= and PPRUNE= options, their values must sum to 1. If you specify the PSPLIT= option but not the PPRUNE= option, the PPRUNE= option is set to 1 minus the PSPLIT= option value.

By default, PPRUNE=0.5.

PSPLIT=*number*

specifies the probability of sampling the operation of splitting a terminal node for the tree sampling algorithm. The value of *number* must be between 0 and 1, exclusive, and if you specify both the PSPLIT= and PPRUNE= options, their values must sum to 1. If you specify the PPRUNE= option but not the PSPLIT= option, the PSPLIT= option is set to 1 minus the PPRUNE= option value.

By default, PSPLIT=0.5.

Details: BART Procedure

Missing Values

Any observation that has missing values for the response or frequency variable is excluded from training and evaluating the model. The ASSIGNMISSING= option determines how PROC BART handles observations that have missing input values. By default, during the training phase missing values for a categorical predictor are treated as a separate level. When a node is split on a continuous predictor, by default if there are observations in which the splitting variable has missing values, a separate rule for routing missing values is sampled before the splitting rule for nonmissing values is sampled. An alternative strategy is to exclude observations that have missing input values from the training phase, or to treat a continuous predictor that has a missing value as either the smallest or largest machine value.

In the scoring phase, observations whose values are not covered by a node's primary splitting rule are assigned to the larger branch of the split. An observation can have a value that is not covered by a primary splitting rule for the following reasons. For a node that is split by using a categorical predictor, an observation that has a level that did not occur in the training data is always uncovered by the splitting rule. In addition, when a node is split on a categorical predictor by using an unordered splitting rule, any observation with a level of the splitting variable that did not occur in the training data on that node is uncovered by the splitting rule. When ASSIGNMISSING=NONE, any observation that has a missing predictor value is uncovered by the splitting rule.

For a node that is split by using a continuous predictor, when ASSIGNMISSING=SEPARATE, an observation that has a missing continuous predictor value is uncovered by the splitting rule if there is no separate rule for missing values. A separate rule for missing values is not sampled when, in the training data, all observations on the node have nonmissing values of the splitting variable.

For more information about how splitting rules are sampled, see the section "Splitting Rules" on page 158.

BART Model for Normal Data

For a response y assumed to follow a normal distribution,

$$y = f(\boldsymbol{x}) + \epsilon \quad \epsilon \sim N(0, \sigma^2)$$

a Bayesian additive regression trees (BART) model uses a sum-of-trees ensemble to approximate $f(\boldsymbol{x})$, the conditional mean of y given the vector of predictors $\boldsymbol{x}$ (Chipman, George, and McCulloch 2010). In addition to requiring you to specify the vector of predictors and the number of trees to use in the ensemble, the BART model requires you to specify a prior for the tree structure, the tree parameters, and the variance parameter σ^2.

The following definitions describe components of the BART prior:

- m denotes the number of trees in the ensemble.
- T_j denotes the structure of the jth tree.
- M_j denotes the set of terminal node, or leaf, parameters in the jth tree.
- b_j denotes the number of terminal nodes in the jth tree.
- $\mu_{\ell j}$ denotes the terminal node parameter associated with the ℓth terminal node in the jth tree.

The BART prior is assumed to be of the following form:

$$\begin{aligned} p\left(T_1, M_1, \ldots, T_m, M_m, \sigma^2\right) &= \prod_{j=1}^{m} p\left(T_j, M_j\right) p(\sigma^2) \\ &= \prod_{j=1}^{m} p\left(M_j \mid T_j\right) p\left(T_j\right) p(\sigma^2) \\ &= \prod_{j=1}^{m} \prod_{\ell=1}^{b_j} p\left(\mu_{\ell j} \mid T_j\right) p\left(T_j\right) p(\sigma^2) \end{aligned}$$

Priors of this form assume independence between the tree structures and conditionally independent terminal node parameters, given the tree structure. The priors for the tree structure $p\left(T_j\right)$, the terminal node parameters $p\left(\mu_{\ell j} \mid T_j\right)$, and the variance term $p\left(\sigma^2\right)$ are chosen to form a regularization prior that limits the contribution of a single tree to the model fit. The use of a regularization prior enforces the so-called weak learner property on the individual trees (Chipman, George, and McCulloch 2010).

Tree Prior

The tree prior $p\left(T_j\right)$ is based on the recommendations of Chipman, George, and McCulloch (1998). For a node η that has depth d from the root node, the probability of the node being split is given by

$$\alpha(1+d)^{-\beta} \quad \alpha \in (0,1),\ \beta \in [0, \infty)$$

As part of the regularization prior, the base split probability α and depth power parameter β are chosen to assign higher prior probability to smaller trees. The recommended distribution for sampling the split

rule is determined by two components: first, a distribution for selecting which variable is used to split the node; and second, a distribution for selecting the split criterion, given the selected variable. When you train the model, you can restrict the set of feasible splitting rules by requiring each terminal node to contain a minimum number of observations. The BART procedure uses a uniform distribution to sample both the splitting variable and the splitting criterion. You can specify the parameters of the tree prior by using the TREEPRIOR statement.

The leaf parameters $p\left(\mu_{\ell j} \mid T_j\right)$ use a normal prior that has mean μ_μ and variance σ_μ^2. Given the assumed form of the BART prior, the conditional mean of the response, given the predictor variables, $f(\boldsymbol{x}) = E\left[Y \mid \boldsymbol{X}\right]$, is the sum of m independent and identically distributed (iid) normal random variables and has mean $m\mu_\mu$ and variance $m\sigma_\mu^2$. As described by Chipman, George, and McCulloch (2010), the parameters μ_μ and σ_μ^2 are chosen to assign high probability to values within the range of the training data. In particular, given a value of $k > 0$ the parameters μ_μ and σ_μ^2 are chosen to satisfy $y_{\max} = m\mu_\mu + k\sqrt{m}\sigma_\mu$ and $y_{\min} = m\mu_\mu - k\sqrt{m}\sigma_\mu$. For $k = 2$, these parameters result in an approximately 95% prior probability for the event $f(\boldsymbol{x}) \in (y_{\min}, y_{\max})$.

For a specified value of k, you can simplify identifying the values of μ_μ and σ_μ^2 by considering a linear transformation of the response variable that results in a new range of $(-0.5, 0.5)$. After such a transformation, the leaf parameter prior would be given by a normal prior that has mean 0 and variance $\frac{0.5}{k\sqrt{m}}$. As noted by Chipman, George, and McCulloch (2010), these choices of parameter values limit the contribution of a single tree and result in a decreasing prior variance for $\mu_{\ell j}$ as the number of trees in the ensemble m or the choice of parameter k increases. You can specify the value of k that is used to determine the prior variance of the leaf parameter by using the LEAFSIGMAK= option in the PROC BART statement.

Variance Prior

The prior for the variance $p\left(\sigma^2\right)$ is chosen to follow a scaled inverse chi-square distribution that has degrees of freedom υ and scale parameter λ. As described by Chipman, George, and McCulloch (2010), the choice of parameters υ and λ can be informed by the training data. Namely, for specified degrees of freedom υ, quantile level q, and estimate for the variance $\hat{\sigma}^2$, you can choose the scale parameter to satisfy $p\left(\sigma^2 < \hat{\sigma}^2\right) = q$. You can specify the parameters that are used to determine the prior for the variance parameter by using the GAUSSPRIOR statement.

Sampling the Posterior

Given a training data set, a Bayesian backfitting MCMC algorithm is used to draw samples from the posterior distribution (Chipman, George, and McCulloch 2010). This algorithm is a form of Gibbs sampling that can be described as follows. Let $\boldsymbol{y}$ denote the vector of training data response values, and let $\boldsymbol{T}_{-j}$ denote the tree structures for the $m-1$ trees in the ensemble, excluding the jth tree. Similarly, let $\boldsymbol{M}_{-j}$ denote the set of all leaf parameters, excluding the jth tree parameters. Each iteration of the sampling algorithms takes m consecutive draws, $j = 1, \ldots, m$, from

$$T_j, M_j \mid \boldsymbol{T}_{-j}, \boldsymbol{M}_{-j}, \boldsymbol{y}, \sigma^2$$

followed by a draw from

$$\sigma^2 \mid T_1, M_1, \ldots, T_m, M_m, \boldsymbol{y}$$

As described by Chipman, George, and McCulloch (2010), the draws for the tree structures T_j and leaf parameters M_j are simplified by observing that their conditional distribution depends on $\boldsymbol{T}_{-j}$, $\boldsymbol{M}_{-j}$, and $\boldsymbol{y}$

only through the partial residuals for the fit excluding the jth tree. Moreover, the choice of the conjugate normal prior for the leaf parameters allows for the sampling of T_j and M_j to be carried out in two steps. Let $\boldsymbol{r}_j$ denote the partial residuals that are based on the fit, excluding the jth tree. You then obtain the samples for T_j and M_j by taking successive draws from

$$T_j \mid \boldsymbol{r}_j, \sigma^2$$
$$M_j \mid T_j, \boldsymbol{r}_j, \sigma^2$$

The draws for the tree structure T_j are obtained by using the Metropolis-Hastings sampling algorithm described by Chipman, George, and McCulloch (1998). This algorithm considers sampling from four operations that modify the tree structure—namely, pruning a pair of terminal nodes, splitting a terminal node, changing the splitting rule of an internal node, or swapping a splitting rule between a parent node and a child node. The BART procedure considers only pruning and splitting operations. As described in Pratola et al. (2014), although the changing and swapping operations significantly improve the fit of single-tree models, these operations have less impact on the sum-of-trees ensemble BART models. Limiting the tree modifications to only pruning and splitting operations was observed to have a limited effect on model fit while providing a significant computational benefit.

Let T_{new} denote the sampled modification to the tree structure T_j, and let $q(\cdot \mid \cdot)$ denote the proposal density. The proposed modification is then accepted with the probability

$$\alpha = \min\left\{ \frac{q(T_j \mid T_{\text{new}})\, p\left(\boldsymbol{r}_j \mid T_{\text{new}}, \sigma^2\right)\, p\left(T_{\text{new}}\right)}{q\left(T_{\text{new}} \mid T_j\right)\, p\left(\boldsymbol{r}_j \mid T_j, \sigma^2\right)\, p\left(T_j\right)}, 1 \right\}$$

After the proposed operation is accepted or rejected, a draw is taken for the set of leaf parameters M_j, and the partial residuals $\boldsymbol{r}_{j+1}$ are then updated for sampling the $j+1$th tree structure T_{j+1} and the $j+1$th set of leaf parameters M_{j+1}.

To begin the sampling, the BART procedure initializes the ensemble with m single-node trees, or stumps. By default, 100 burn-in iterations are performed before the main simulation loop of 1,000 iterations. Samples from the main simulation loop are saved for prediction and computing posterior statistics. You can use the NTHIN= option to specify a thinning rate for the main simulation. Thinning is often used to reduce the correlation among posterior sample draws. When you apply thinning, the number of samples that are saved for prediction is $\left\lceil \frac{\text{NMC}}{\text{NTHIN}} \right\rceil$. For more information about Markov chain sample size, burn-in, and thinning, see the section "Burn-In, Thinning, and Markov Chain Samples" (Chapter 8, *SAS/STAT User's Guide*) in Chapter 8, "Introduction to Bayesian Analysis Procedures" (*SAS/STAT User's Guide*).

When you run PROC BART on a cluster of machines, you can use a distributed mode that divides the MCMC samples. By default, each worker node in the cluster has a full copy of the training data and runs a separate chain. You can specify the number of separate chains by using the NDISTCHAINS= option in the PROC BART statement. Specifying NDISTCHAINS=0 requests the use of a single chain in which each worker node is assigned a portion of the training data. Each chain performs the number of burn-in iterations that you specify in the NBI= option in the PROC BART statement. By default, the size of the main simulation loop for each chain is set so that the total number of samples that are saved across chains is equal to $\left\lceil \frac{\text{NMC}}{\text{NTHIN}} \right\rceil$, the number of samples that would be saved if you used a single chain. Alternatively, you can use the NMCDIST= option in the PROC BART statement to specify the main simulation loop size for each chain. Thinning is applied separately to each chain. Note that the fit of a BART model depends on the number of chains that you run.

Predictions from the trained model are obtained by averaging predictions from each of the saved posterior samples. Uncertainty in the model predictions can be assessed on the basis of variability of the posterior samples.

Splitting Rules

PROC BART samples proposed splitting operations by first sampling a terminal node to split, then sampling a variable to split on, and finally sampling a splitting rule for that variable. At each step, the samples are taken uniformly from the set of feasible options. Feasibility of a split is determined by the MINLEAFSIZE= option in the PROC BART statement, which specifies the minimum number of training observations per leaf. The following sections describe how splitting rules for continuous and categorical predictors are sampled.

Continuous Predictors

Splitting rules for a continuous predictor x are of the form $x < c$. PROC BART bins continuous predictors and samples the values of c from the right bin endpoints. You specify the number of bins and the method that is used to bin continuous predictors by using the NBINS= and BINMETHOD= options, respectively, in the PROC BART statement. Observations that have missing values do not contribute to the computation of the bin endpoints.

The ASSIGNMISSING= option in the PROC BART statement determines how to handle missing values during the training and scoring phases. During the training phase, by default observations with missing values are treated as a separate group. If the node that is being split has observations with missing values of the split variable x, a separate splitting rule is sampled for the missing group before the splitting rule for nonmissing values is sampled. During the scoring phase, if an observation has a missing value and the node has a rule for routing the missing group, that rule is applied. If the node does not have a separate rule for routing the missing group, observations with missing values are assigned to the larger branch of the split. Alternatively, observations with missing values could be treated as the largest machine value or the smallest machine value, or they could be excluded from the training phase. For more information about how PROC BART handles missing values, see the section "Missing Values" on page 154.

Categorical Predictors

Splitting rules for categorical predictors x can be either ordered or unordered. Which type of splitting rules the procedure uses for a categorical predictor is determined by the cardinality of the predictor and the value of the ORDERSPLIT= option in the PROC BART statement. Categorical predictors whose cardinality is greater than or equal to the value of the ORDERSPLIT= option use ordered splitting rules, and predictors whose cardinality is less than the value of the ORDERSPLIT= option use unordered splitting rules.

Ordered splitting rules are based on the sorted level ordering of the variable as determined by the levelization process. These rules are of the form $x < l$, where x denotes the observed level order and l is an integer value. Unordered splitting rules are of the form $x \in C$, where C denotes a set of levels. The set C is determined by randomly assigning levels to the left or right branch of a split. Assuming that a categorical predictor has n levels on a node, each with a frequency greater than the minimum leaf size, there are $n - 1$ possible ordered splitting rules and 2^{n-1} unordered splitting rules. The use of ordered or unordered splitting rules is determined by the cardinality of the variable, not the number of levels that occur on the node that is being split. The use of ordered splitting rules can significantly decrease the amount of memory that is needed to represent the fitted model when there are high-cardinality input variables.

The ASSIGNMISSING= option in the PROC BART statement determines how to handle missing values during the training and scoring phases. By default, during the training phase, missing values are treated as a separate level. During the scoring phase, an observation whose level is unknown is assigned to the larger branch of the split. An observation has an unknown level if the level does not occur in the training data or does not occur in the training data on a node that has an unordered splitting rule. Alternatively, observations that have missing values could be excluded from the training phase. For more information about how missing values are handled, see the section "Missing Values" on page 154.

BART Model for Binary Data

As described by Chipman, George, and McCulloch (2010), a probit BART model can be used to analyze binary response data.

The probit BART model has the following components:

- Φ denotes the standard normal cumulative distribution function
- $g_j(\boldsymbol{x})$ denotes the prediction function that corresponds to the jth tree defined by T_j and M_j
- $G(\boldsymbol{x})$ denotes the sum-of-trees prediction function
- c denotes an offset variable

The BART probit model assumes that the probability of an event is given by

$$P\left(Y = 1 \mid \boldsymbol{x}\right) = \Phi\left(G(\boldsymbol{x}) + c\right)$$

The regularization prior for the probit model inherits many properties of the prior for the Gaussian model. For more information about the regularization prior for Gaussian BART models, see the section "BART Model for Normal Data" on page 155.

Like the Gaussian BART model, the probit BART model assumes independent tree structures and conditionally independent terminal node parameters, given the tree structure. The prior probability of a node being split is also given by the same function of the node's depth. For more information about the tree prior, see the section "Tree Prior" on page 155. The differences between the Gaussian and probit BART models are related to the variance parameter σ^2, the prior that is used for the leaf parameters $\mu_{\ell j}$, the presence of an offset variable c, and the algorithm that is used to generate posterior samples of the sum-of-trees ensemble. The remainder of this section elaborates on these differences.

As described by Chipman, George, and McCulloch (2010), samples of the sum-of-trees ensemble for probit models are produced by a Gibbs sampling algorithm that is similar to the algorithm that is used for Gaussian models but involves the addition of latent variables $Z_i, i = 1, \ldots, n$. The latent variables Z_i are assumed to be independent and identically distributed (iid) normal random variables with mean $G(\boldsymbol{x}_i) + c$ and variance 1. Moreover, the latent variables are assumed to be related to the observed response y_i by

$$y_i = 1 \text{ if } Z_i > 0 \quad \text{and} \quad y_i = 0 \text{ if } Z_i \leq 0$$

Conditional on the observed response values, the latent variables Z_i therefore follow the truncated normal distributions

$$Z_i \mid y_i = 1 \sim \max\left\{N(G(\boldsymbol{x}_i + c), 1), 0\right\} \quad \text{and} \quad Z_i \mid y_i = 0 \sim \min\left\{N(G(\boldsymbol{x}_i + c), 1), 0\right\}$$

With these assumptions, the latent variables are used to sample updates to the tree structures and leaf parameters. The algorithm for sampling these updates follows the same process that is used for Gaussian models, but the observed response values are replaced by the sampled values of the latent variables. For more information about the sampling algorithm that is used for Gaussian models, see the section "Sampling the Posterior" on page 156.

For the latent variables, the probit BART model assumes a prior of the form

$$p\left(T_1, M_1, \ldots, T_m, M_m\right) = \prod_{j=1}^{m} \prod_{\ell=1}^{b_j} p\left(\mu_{\ell j} \mid T_j\right) p\left(T_j\right)$$

Like the Gaussian model, the probit model uses a normal prior for the leaf parameters μ_{ℓ_j} that enforces the weak learner property. But in contrast to the Gaussian model, the prior mean and variance do not depend on the range of the observed data. For a value of $k > 0$, the prior mean and variance for the probit model leaf parameters are

$$\mu_\mu = 0 \quad \text{and} \quad \sigma_\mu = \frac{3}{k\sqrt{m}}$$

This choice for the prior mean and variance places a greater prior probability on values of $G(\boldsymbol{x})$ within the range $(-3, 3)$ and applies regularization to $G(\boldsymbol{x})$, shrinking its values toward 0. You can specify the value of k that determines the prior variance of the leaf parameter by using the LEAFSIGMAK= option in the PROC BART statement.

Each iteration of the Gibbs sampling algorithm for probit models begins by sampling values of $z_i \mid y_i$ for each observation $i = 1, \ldots, n$. Let $\boldsymbol{z}$ denote the vector of sampled values, and let $\boldsymbol{T}_{-j}$ denote the tree structures for the $m - 1$ trees in the ensemble, excluding the jth tree. Similarly, let $\boldsymbol{M}_{-j}$ denote the set of all leaf parameters, excluding the jth tree parameters. Then, for each tree T_j and set of leaf parameters M_j, $j = 1, \ldots, m$, samples are taken from

$$T_j, M_j \mid \boldsymbol{T}_{-j}, \boldsymbol{M}_{-j}, \boldsymbol{z}$$

As with the algorithm that is used for Gaussian models, the updates for the tree structure T_j and M_j can be made in two steps. First, an update to the tree structure is sampled, and then an update to the leaf parameters, given the updated tree structure, is sampled. The proposal density for modifying the tree structure and the probability of accepting a proposed modification to the tree structure are determined by using the same expressions that are described in the section "Sampling the Posterior" on page 156.

By default, for probit models, the BART procedure determines the value of the offset variable c on the basis of the proportion of events $y_i = 1$ that occur in the training data. Alternatively, you can use the OFFSET= option in the MODEL statement to specify an offset variable, in which case the offset values can be different for each observation.

Computational Resources

The Bayesian additive regression trees (BART) algorithm is computationally intensive. For each posterior sample, the modifications to each tree are sampled sequentially, and each tree modification requires at least one pass through the data. As a result, for large data sets the time that PROC BART spends generating the posterior samples and training the model can be significant. You can use the MAXTRAINTIME= option in the PROC BART statement to specify a maximum time (in seconds) for training to model.

PROC BART also supports options that you can use to reduce the time that it takes to train the model. However, each of these options can require a significant amount of memory and might not be suitable for very large data sets.

You can use the TRAININMEM option in the PROC BART statement to store the data in memory when the model is trained. Storing the data in memory can reduce the time that it takes to access data during the sampling of tree modifications. This option requires memory allocations that are in total proportional to the number of observations times the size of an input row. By default, the data are not stored in memory during model training.

You can use the MAPINMEM option in the PROC BART statement to store a mapping of each observation to terminal nodes, or leaves, in memory during model training. Storing a mapping of each observation to the terminal nodes that it corresponds to removes the need to route an observation through a tree when you are sampling updates to a tree's structure and its leaf parameters. This option requires memory allocations that are in total proportional to the number of observations times the number of trees. By default, the observation-to-leaf mapping is not stored in memory during model training.

When PROC BART is run on a cluster of machines, you can run multiple chains and divide the MCMC samples across worker nodes. In addition to allowing for multiple chains to be run in parallel, distributed mode can reduce the time that it takes for communications across the cluster of machines. This mode requires distributing the training data to each worker so that each worker that runs a separate chain has a full copy of the training data. This option is not applicable in single-machine mode. By default, distributed mode is used when PROC BART is run on a cluster of machines. You can use the NDISTCHAINS= option in the PROC BART statement to specify the number of distributed chains to use. When NDISTCHAINS=0, a single chain is run, and each worker node is assigned a portion of the training data. Note that the fit of a BART model depends on the number of chains that you run.

Displayed Output

The following sections describe the output that PROC BART produces. The output is organized into various tables, which are discussed in order of their appearance.

Model Information

The "Model Information" table displays basic information about the model, such as the input data, response variable, frequency variable, MCMC burn-in size, simulation size, thinning rate, and number of trees in the ensemble samples.

Class Information

The "Class Level Information" table lists the levels of every variable that you specify in the CLASS statement. You should check this information to make sure that it is correct. You can suppress the "Class Level Information" table completely or partially by specifying the NOCLPRINT= option in the PROC BART statement.

Number of Observations

The "Number of Observations" table displays the number of observations that are read from the input data table and the number of observations that are used in the analysis. If you specify a FREQ statement, the table displays the sum of the frequencies that are read and used.

Response Profile

The "Response Profile" table displays the ordered value from which the BART procedure determines the probability that is being modeled as an event in a binary model. For each response category level, the frequency that is used in the analysis is reported. For a binary model, the note that follows the "Response Profile" table indicates which outcome is modeled as the event. If you specify a PARTITION statement, the table displays the values for each role.

Prior Information

The "Prior Information" table displays basic information about the model prior, such as the parameters for the tree prior and the prior for the variance parameter for a normal response variable.

Chain Information

The "Chain Information" table displays information about the number of MCMC samples that are saved from each chain. This table is produced only when you run multiple chains.

Sample Information

The "Sample Information" table displays basic information about the posterior samples of the sum-of-trees ensemble, such as the number of samples saved, the proportion of operations accepted, and information about the size of the trees.

Variance Autocorrelations

The "Variance Autocorrelations" table displays the first-order autocorrelations of the posterior samples for the variance parameter of a normally distributed response. By default, PROC BART displays lag 1, 5, 10, and 50 estimates for the autocorrelations. You can request different autocorrelations by specifying the VARAUTOCORR= option in the GAUSSPRIOR statement.

Fit Statistics

The "Fit Statistics" table displays the average square error (ASE) that is computed on the basis of the final selected model. If you specify a PARTITION statement, the table displays the ASE for each role.

Classification

The "Classification" table displays information about both the observed and predicted values for a binary response. The table includes columns for the observed response, the predicted event count, the predicted nonevent count, and the total number of events and nonevents. If you specify a PARTITION statement, the table displays the values for each role.

Margins

The "Margins" table displays predictive margin estimates. Each row of the table corresponds to a predictive margin that you define in a MARGIN statement.

MarginDiffs

The "Margin Differences" table displays differences between predictive margin estimates. Each row of the table corresponds to a difference between predictive margins that you specify in a MARGINDIFF statement.

Timing

The "Timing" table displays the amount of time (in seconds) that it took PROC BART to perform different tasks in the analysis.

OutputCasTables Table

The OutputCasTables table is a special table that has information about each CAS table that is created during a CAS action execution. The information for each CAS table consists of the CAS table name, the caslib in which the table resides, and the number of columns and rows in the CAS table. Because this table is not a typical ODS table that contains analytical results, you cannot include it in the *table-spec-list* in the DISPLAYOUT statement.

ODS Table Names

Each table that the BART procedure creates has a name that is associated with it. You must use this name to refer to the table when you use ODS statements. Table 4.4 lists the name of each table and a short description of its contents.

Table 4.4 ODS Tables Produced by PROC BART

Table Name	Description	Statements	Option
ChainInfo	Information about samples per chain	Default for distributed chains	
Classification	Classification matrix	Default output for binary data	

Table 4.4 *continued*

Table Name	Description	Statements	Option
ClassInfo	Level information from the CLASS statement	CLASS	
FitStatistics	Fit statistics	Default output	
MarginDiffs	Differences in predictive margins	MARGINDIFF	
Margins	Predictive margins	MARGIN	
ModelInfo	Information about the modeling environment	Default output	
NObs	Number of observations read and used	Default output	
PriorInfo	Information about the model prior	Default output	
OutputCasTables	See the section "OutputCasTables Table" on page 163	OUTPUT DISPLAYOUT SAMPSUMMARY	OUT=
ResponseProfile	Response categories for binary data	Default output for binary data	
ScoreInfo	Information about the stored model used for scoring	PROC BART	RESTORE=
SampleInfo	Information about the posterior samples	Default output	
Timing	Absolute and relative times for tasks performed by the procedure	Default output	
VarAutoCorr	Autocorrelation statistics for normal response variance	Default output	

Examples: BART Procedure

The examples in this chapter are available in the GitHub repository located at `https://github.com/sascommunities/doc-supplement-casstat`.

Example 4.1: Storing and Scoring

(View the complete code for this example (bartex1.sas) in the example repository.)

When you use PROC BART to fit a model, you can use the STORE statement to save the model. For example, you can score new observations or define interventions on predictors and score marginal means for the interventions. This example uses the RESTORE= option in PROC BART to do subsequent scoring. Note that because scoring a new observation for a Bayesian additive regression trees (BART) model requires routing the observation through each tree for all the saved samples of the sum-of-trees ensemble, scoring large data sets can be time-consuming.

The following statements generate the data table inputData, which consists of 10,000 observations on a continuous response variable (y) and 40 continuous variables (x1–x40), in your CAS session:

```
data mylib.inputData / single =yes;
   drop  j w1-w40;

  array x{40};
  array w{40};
  call streaminit(6524);
  pi=constant("pi");

   do i=1 to 10000;
      u = rand("Uniform");
      do j=1 to dim(x);
         w{j} = rand("Uniform");
         x{j} = (w{j} + u)/2;
      end;

      f1 = sin(pi * x1 * x2 );
      f2 = (x3-0.5)**2;
      f3 = x4;
      f4 = x5;
      fb = 10*f1 +20*f2+10*f3+5*f4;

      y = fb +  rand("Normal");
      output;
   end;

run;
```

For the simulation, only the variables x1–x5 affect the outcome y, and the variables x1–x40 are all positively correlated.

In the following program, because the input data are not large, the TRAININMEM and MAPINMEM options store the data and elements of the model in memory when the model is trained:

```
proc bart data=mylib.inputData seed=9181 trainInMem mapInMem;
   model y = x1-x40;
   store mylib.modelFit;
run;
```

The mylib.modelFit store contains a representation of the model that you can use to score new observations. The fit statistics for the fitted model are displayed in the "Fit Statistics" table in Output 4.1.1.

Output 4.1.1 Fit Statistics

The BART Procedure

Fit Statistics	
Average Square Error	0.97290

The following statements create a new data table of observations to score using the same data-generating process that generated the training data:

```
data mylib.toScoreData / single =yes;
   drop  j w1-w40;

  array x{40};
  array w{40};
  call streaminit(1972);
  pi=constant("pi");

   do i=1 to 1000;
      u = rand("Uniform");
      do j=1 to dim(x);
         w{j} = rand("Uniform");
         x{j} = (w{j} + u)/2;
      end;

      f1 = sin(pi * x1 * x2 );
      f2 = (x3-0.5)**2;
      f3 = x4;
      f4 = x5;
      fb = 10*f1 +20*f2+10*f3+5*f4;

      y = fb +  rand("Normal");
      output;
   end;

run;
```

The following statements show how to use PROC BART to score the new data by using the previously fitted model. The saved model mylib.modelFit is specified in the RESTORE= option. You use the DATA= option to specify the data table of observations to score, and you use the OUTPUT statement to designate the output data table that contains the results. The PRED= keyword names the variable in the output data table that contains the predicted response value. Note that because the toScoreData data table also contains the response variable y, residuals for the predicted values can also be computed. The RESID= keyword names the variable in the output data table that contains the residuals. The RESTORE= option and OUTPUT statement are both required when you use PROC BART to score data by using a previously fitted model.

```
proc bart data=mylib.toScoreData restore=mylib.modelFit;
   output out = mylib.scoredData pred = predResp resid = residual;
run;
```

The following statements compute the average square error (ASE) for the scored observations:

```
data fitCheck;
   set mylib.scoredData;
   SquareError = residual * residual;
run;

proc means data=fitCheck mean;
   var SquareError;
run;
```

The ASE for the scored observations in Output 4.1.2 is similar to the ASE for the training data in Output 4.1.1,

indicating that the fitted model generalizes well.

Output 4.1.2 ASE for the Scored Data

The MEANS Procedure

Analysis Variable : SquareError
Mean
1.0628698

You can also use the saved model mylib.modelFit to compute predictive margins. You can use the MARGIN statement to define predictive margins that intervene on one or more of the input variables. Observations in an input data table are scored according to the specified interventions, and means of the predicted values are reported along with equal-tail credible intervals. When you want to compare the difference between predicted margins, you can use the MARGINDIFF statement to specify a contrast of predictive margins that are defined in MARGIN statements.

The following statements show how you can compute predictive margins by using PROC BART and the previously fitted model that is saved in the analytic store mylib.modelFit. The saved model mylib.modelFit is specified in the RESTORE= option, and the data table of observations to score is specified in the DATA= option. Five predictive margins are specified in the MARGIN statement. The first scenario intervenes on x2 and sets its value to 0.25, and the second scenario intervenes on both x2 and x3 and sets their values to 0.25 and 0.5, respectively. The remaining three predictive margins intervene on x1 and set its level to 0.25, 0.5, and 0.75, respectively. These three predictive margins are then used in MARGINDIFF statements to request differences between these predictive margins.

```
proc bart restore = mylib.modelFit data=mylib.inputData;
   margin "Scenario1" x2 = 0.25;
   margin "Scenario2" x2 = 0.25 x3 = 0.5;
   margin "x1Ref"     x1 = 0.25;
   margin "x1Evt1"    x1 = 0.5;
   margin "x1Evt2"    x1 = 0.75;
   margindiff event = "x1Evt1" ref = "x1Ref" / label= "x1:0.5 - 0.25";
   margindiff event = "x1Evt2" ref = "x1Ref" / label= "x1:0.75 - 0.25";
run;
```

The "Score Information" table in Output 4.1.3 displays information about the saved model, the data table that is used to train the model, the data table of scored observations, and the number of scored observations.

Output 4.1.3 Score Information

The BART Procedure

Scoring Information	
Store	MODELFIT
Scoring Table	INPUTDATA
Training Table	INPUTDATA
Target Variable	y
Missing Rule	Separate
MCMC Sample Size	1000
Number of Observations	10000

The "Predictive Margins" table in Output 4.1.4 displays the means for the predicted margin estimates and the equal-tail credible intervals for the interventions that are specified by a MARGIN statement.

Output 4.1.4 Predictive Margins

Predictive Margins

Description	Estimate	95% Equal-Tail Interval	
Scenario1	12.14185	11.97984	12.31817
Scenario2	11.36423	11.20431	11.55277
x1Ref	12.22552	12.07488	12.42200
x1Evt1	15.13663	15.01155	15.25285
x1Evt2	16.54028	16.39544	16.66414

The "Predictive Margin Differences" table in Output 4.1.5 displays the mean difference between predictive margins that is specified in the MARGINDIFF statement along with equal-tail credible intervals.

Output 4.1.5 Predictive Margin Differences

Predictive Margin Differences

Description	Estimate	95% Equal-Tail Interval	
x1:0.5 - 0.25	2.9111	2.6998	3.1283
x1:0.75 - 0.25	4.3148	4.0424	4.5075

Example 4.2: Probit Model for Binary Data

(View the complete code for this example (bartex2.sas) in the example repository.)

This example uses the junk email available in the Sashelp.JunkMail data set. The data set comes from a study that classifies whether an email is junk email (coded as 1) or not (coded as 0). The data were collected by Hewlett-Packard Labs and donated by George Forman. The data table, which is specified in the following DATA step, contains 4,601 observations, with two binary variables and 57 continuous explanatory variables. The response variable, Class, is a binary indicator of whether an email is considered spam or not. The partitioning variable, Test, is a binary indicator that is used to divide the data into training and testing parts. The 57 explanatory variables are continuous variables that represent the frequencies of some common words and characters and the lengths of uninterrupted sequences of capital letters in emails.

```
data mylib.JunkMail;
   set Sashelp.JunkMail;
run;
```

In the following program, the PARTITION statement divides the data into two parts: the training data have a Test value of 0 and contain about two-thirds of the data, and the rest of the data are used to evaluate the fit. You use the DIST= option in the MODEL statement to indicate that the response is binary.

```
proc bart data=mylib.JunkMail seed=210124;
   model Class = Make Address All _3d Our Over Remove
         Internet Order Mail Receive Will People Report Addresses Free Business
         Email You Credit Your Font _000 Money HP HPL George _650 Lab Labs
         Telnet _857 Data _415 _85 Technology _1999 Parts PM Direct CS Meeting
         Original Project RE Edu Table Conference Semicolon Paren Bracket
         Exclamation Dollar Pound CapAvg CapLong CapTotal
   / dist=binary;
   partition rolevar=Test(train='0' test='1');
run;
```

The "Number of Observations" and "Response Profile" tables in Output 4.2.1 are divided into training and testing columns.

Output 4.2.1 Partitioned Counts

The BART Procedure

Number of Observations Read	4601
Number of Observations Used	4601
Number of Observations Used for Training	3065
Number of Observations Used for Testing	1536

Response Profile		
Class	**Training**	**Testing**
1	1218	595
0	1847	941

Probability modeled is Class = 0.

The fit statistics for the selected model are displayed in the "Fit Statistics" table in Output 4.2.2. The misclassification rate and average square error are similar for the training and testing data, indicating that the fitted model generalizes well.

Output 4.2.2 Partitioned Fit Statistics

Fit Statistics		
	Training	**Testing**
Average Square Error	0.03995	0.04744
Misclassification Rate	0.04698	0.06055

To enable you to further assess the performance of the model, the "Classification Matrix" table in Output 4.2.3 displays information about the observed and predicted values for both the training and test data.

Output 4.2.3 Partitioned Classification Matrix

Classification Matrix						
	Training Prediction			Testing Prediction		
Observed	0	1	Total	0	1	Total
0	1790	57	1847	901	40	941
1	87	1131	1218	53	542	595
Total	1877	1188	3065	954	582	1536

References

Asuncion, A., and Newman, D. J. (2007). "UCI Machine Learning Repository." http://archive.ics.uci.edu/ml/.

Breiman, L. (1996). "Bagging Predictors." *Machine Learning* 24:123–140.

Breiman, L. (2001). "Random Forests." *Machine Learning* 45:5–32.

Chipman, H. A., George, E. I., and McCulloch, R. E. (1998). "Bayesian CART Model Search." *Journal of the American Statistical Association* 93:935–948. https://doi.org/10.2307/2669832.

Chipman, H. A., George, E. I., and McCulloch, R. E. (2010). "BART: Bayesian Additive Regression Trees." *Annals of Applied Statistics* 4:266–298. https://doi.org/10.1214/09-AOAS285.

Friedman, J. H. (2001). "Greedy Function Approximation: A Gradient Boosting Machine." *Annals of Statistics* 29:1189–1232.

Lane, P. W., and Nelder, J. A. (1982). "Analysis of Covariance and Standardization as Instances of Prediction." *Biometrics* 38:613–621.

Pratola, M. T., Chipman, H. A., Gattiker, J. R., Higdon, D. M., McCulloch, R. E., and Rust, W. N. (2014). "Parallel Bayesian Additive Regression Trees." *Journal of Computational and Graphical Statistics* 23:830–852. https://doi.org/10.1080/10618600.2013.841584.

Chapter 5
BINNING Procedure

Contents

Overview: BINNING Procedure **172**
- Bucket Binning 172
- Winsorized Binning 172
- Quantile Binning 172
- Cutpoint Binning 172
- Tree-Based Binning 173
- PROC BINNING Features 173
- Using CAS Sessions and CAS Engine Librefs 173

Getting Started: BINNING Procedure **174**
- Bucket Binning 174

Syntax: BINNING Procedure **176**
- PROC BINNING Statement 176
- CODE Statement 181
- FREQ Statement 181
- INPUT Statement 181
- OUTPUT Statement 182
- TARGET Statement 183
- WEIGHT Statement 184

Details: BINNING Procedure **184**
- Binning Computation and Formulas 184
- Tree-Based Binning 186
- Computing the Weight of Evidence and Information Value 186
- Displayed Output 187
- ODS Table Names 188

Examples: BINNING Procedure **189**
- Example 5.1: Quantile Binning 189
- Example 5.2: Winsorized Binning 191
- Example 5.3: Bucket Binning and Weight-of-Evidence Computation 193
- Example 5.4: Cutpoint Binning 195
- Example 5.5: Tree-Based Binning 196

Overview: BINNING Procedure

The BINNING procedure performs binning in SAS Viya. Binning is a common step in the data preparation stage of the model-building process. You can use binning to classify missing variables, reduce the impact of outliers, and generate multiple effects. The generated effects are useful and contain certain nonlinear information about the original variables.

The BINNING procedure supports several binning methods, which are described in the following subsections. The BINNING procedure can also calculate the weight of evidence (WOE) and information value (IV) based on binning results.

Bucket Binning

Bucket binning creates equal-length bins and assigns each observation in the data to one of these bins. You can choose the number of bins during the binning; the default number of bins (the binning level) is 16. Only interval input variables are supported for this binning method.

Winsorized Binning

Winsorized binning is similar to bucket binning except that both tails are cut off to obtain a smooth binning result. This technique is often used to remove outliers during the data preparation stage. Only interval input variables are supported for this binning method.

Quantile Binning

PROC BINNING calculates the quantile (or percentile) cutpoints and uses them as the lower bound and upper bound in creating bins. As a result, each bin should have a similar number of observations. Because PROC BINNING always assigns observations that have the same value to the same bin, quantile binning might create unbalanced bins if any variable has tied values. Only interval input variables are supported for this binning method. For more information, see the section "Binning Computation and Formulas" on page 184.

Cutpoint Binning

The cutpoint binning method enables you to create bins by specifying the bin upper bound. Only interval input variables are supported for this binning method.

Tree-Based Binning

The tree-based binning method enables you to create bins by creating decision trees or regression trees. This technique requires you to specify a target variable. If the target variable is nominal, PROC BINNING creates decision trees; if the target variable is interval, PROC BINNING creates regression trees.

PROC BINNING Features

The BINNING procedure has the following features:

- provides a bucket (equal-length) binning method
- provides a Winsorized binning method and Winsorized statistics
- provides a quantile binning method
- provides a cutpoints binning method
- provides a tree-based binning method
- calculates the weight of evidence (WOE) and information value (IV) based on binning results

Using CAS Sessions and CAS Engine Librefs

SAS Cloud Analytic Services (CAS) is the analytic server and associated cloud services in SAS Viya. This section describes how to create a CAS session and set up a CAS engine libref that you can use to connect to the CAS session. It assumes that you have a CAS server already available; contact your system administrator if you need help starting and terminating a server. This CAS server is identified by specifying the host on which it runs and the port on which it listens for communications. To simplify your interactions with this CAS server, the host information and port information for the server are stored as SAS option values that are retrieved automatically whenever this CAS server needs to be accessed. You can examine the host and port values for the server at your site by using the following statements:

```
proc options option=(CASHOST CASPORT);
run;
```

In addition to starting a CAS server, your system administrator might also have created a CAS session and a CAS engine libref for your use. You can define your own sessions and CAS engine librefs that connect to the CAS server as shown in the following statements:

```
cas mysess;
libname mylib cas sessref=mysess;
```

The CAS statement creates the CAS session named mysess, and the LIBNAME statement creates the mylib CAS engine libref that you use to connect to this session. It is not necessary to explicitly name the CASHOST and CASPORT of the CAS server in the CAS statement, because these values are retrieved from the corresponding SAS option values.

If you have created the mysess session, you can terminate it by using the TERMINATE option in the CAS statement as follows:

```
cas mysess terminate;
```

For more information about the CAS and LIBNAME statements, see the section "Introduction to Shared Concepts" on page 6 in Chapter 2, "Shared Concepts."

Getting Started: BINNING Procedure

Bucket Binning

(View the complete code for this example at https://github.com/sascommunities/doc-supplement-casstat/tree/main/Examples/binning_gs.sas.)

NOTE: Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections "Using CAS Sessions and CAS Engine Librefs" on page 7 and "Loading a SAS Data Set onto a CAS Server" on page 7 in Chapter 2, "Shared Concepts."

The following DATA step creates the data table mylib.bucket, which consists of 1,000 observations of an ID variable (id) and three continuous variables (x1–x3), in your CAS session:

```
data mylib.bucket;
   length id 8;
   call streaminit(12345);
   do id=1 to 1000;
      x1 = rand("Uniform");
      x2 = rand("Uniform") * 10;
      x3 = rand("Uniform") * 100;
      output;
   end;
run;
```

These statements assume that your CAS engine libref is named mylib, but you can substitute any appropriately defined CAS engine libref.

The following statements show how you can use the BINNING procedure to perform bucket binning:

```
proc binning data=mylib.bucket numbin=10 method=bucket;
   input x1-x3;
   output out=mylib.out;
run;
```

The DATA= option specifies the input data table. The NUMBIN= option requests that 10 bins be created for all binning variables. The METHOD= option requests that the bucket binning method be used. The INPUT statement names three continuous variables (x1–x3) as input variables for binning. The OUTPUT statement creates an OUTPUT data table to contain the results of PROC BINNING.

Figure 5.1 displays the "Bin Details" table. This table shows the binning variable, bin ID, bin lower bound, bin upper bound, bin width, number of observations in that bin, and some statistics of that bin (such as mean, standard deviation, minimum, and maximum).

Figure 5.1 Bin Details

The BINNING Procedure

Bin Details

Variable	Bin ID	Lower Bound	Upper Bound	Bin Width	Number of Observations	Mean	Standard Deviation	Minimum	Maximum
x1	Missing				0				
	1	-Infty	0.1004		97	0.0553	0.0279	0.0005	0.0985
	2	0.1004	0.2003	0.0999	106	0.1490	0.0296	0.1006	0.1980
	3	0.2003	0.3002	0.0999	92	0.2497	0.0267	0.2012	0.2987
	4	0.3002	0.4001	0.0999	99	0.3529	0.0266	0.3024	0.3993
	5	0.4001	0.5000	0.0999	89	0.4422	0.0288	0.4007	0.4995
	6	0.5000	0.5998	0.0999	107	0.5437	0.0286	0.5002	0.5931
	7	0.5998	0.6997	0.0999	95	0.6471	0.0294	0.6013	0.6961
	8	0.6997	0.7996	0.0999	93	0.7521	0.0300	0.7003	0.7984
	9	0.7996	0.8995	0.0999	101	0.8446	0.0272	0.8010	0.8994
	10	0.8995	Infty		121	0.9452	0.0288	0.8997	0.9994
x2	Missing				0				
	1	-Infty	1.0044		97	0.4827	0.2799	0.0057	0.9973
	2	1.0044	2.0031	0.9987	104	1.5221	0.2821	1.0067	1.9993
	3	2.0031	3.0018	0.9987	111	2.5116	0.2972	2.0079	2.9934
	4	3.0018	4.0006	0.9987	95	3.5063	0.2880	3.0024	3.9980
	5	4.0006	4.9993	0.9987	91	4.4807	0.3002	4.0113	4.9960
	6	4.9993	5.9980	0.9987	77	5.5351	0.2816	5.0104	5.9959
	7	5.9980	6.9967	0.9987	100	6.5170	0.2783	6.0154	6.9882
	8	6.9967	7.9954	0.9987	111	7.4421	0.2722	7.0002	7.9516
	9	7.9954	8.9941	0.9987	110	8.4127	0.2815	8.0026	8.9929
	10	8.9941	Infty		104	9.5013	0.2805	9.0124	9.9928
x3	Missing				0				
	1	-Infty	10.192		101	4.8474	3.1068	0.2160	10.190
	2	10.192	20.167	9.9757	94	14.880	2.9955	10.211	20.084
	3	20.167	30.143	9.9757	104	25.173	3.0393	20.207	30.141
	4	30.143	40.119	9.9757	85	34.993	2.9221	30.542	40.059
	5	40.119	50.095	9.9757	95	44.885	3.0297	40.312	49.732
	6	50.095	60.070	9.9757	104	54.577	2.8664	50.097	59.895
	7	60.070	70.046	9.9757	94	64.963	2.7889	60.139	69.596
	8	70.046	80.022	9.9757	124	75.166	2.6515	70.065	79.945
	9	80.022	89.998	9.9757	85	84.991	2.8269	80.142	89.789
	10	89.998	Infty		114	94.853	2.9066	90.046	99.973

Syntax: BINNING Procedure

The following statements are available in the BINNING procedure:

PROC BINNING < *options* > **;**
CODE FILE=*filename* **;**
FREQ *variable* **;**
INPUT *variables* < *options* > **;**
OUTPUT OUT=*libref.data-table* < *options* > **;**
TARGET *variable* / < *options* > **;**
WEIGHT *variable* **;**

The PROC BINNING statement and at least one INPUT statement are required. You can specify multiple INPUT statements. The TARGET statement is required to calculate the weight of evidence (WOE) and information value (IV) when the value of the METHOD= option is BUCKET, QUANTILE, WINSOR, or CUTPTS. The TARGET statement is also required when the value of the METHOD= option is TREE.

The following sections describe the PROC BINNING statement and then describe the other statements in alphabetical order.

PROC BINNING Statement

PROC BINNING < *options* > **;**

The PROC BINNING statement invokes the procedure. Table 5.1 summarizes important *options* in the PROC BINNING statement by function.

Table 5.1 PROC BINNING Statement Options

Option	Description
Basic Option	
DATA=	Specifies the input data table
Precision Option	
FUZZYCOMPARE=	Specifies the fuzzy comparison threshold that is used to determine the distinctness of numeric values
Binning Level Options	
DISTINCTCOUNTLIMIT=	Specifies the maximum number of distinct levels for interval variables in order for them to be treated as nominal variables
NUMBIN=	Specifies the global number of bins for all binning variables
Binning Method Option	
METHOD=	Specifies which binning method to use
Tolerance Option	
PERCENTILETOLERANCE=	Specifies the tolerance for percentile computation

Table 5.1 *continued*

Option	Description
Maximum Iterations Option	
PERCENTILEMAXITERATIONS=	Specifies the maximum iterations for percentile computation
Weight-of-Evidence Option	
WOE(WOEADJUST=)	Computes the weight of evidence and information values
Missing Value Handling Options	
BINMISSING=	Specifies whether to use missing values of the input variables for the binning process
MISSINGBINSTATS=	Specifies whether to use the bin that contains the missing values for binning analysis
MISSINGEVALNONEVENT=	Specifies whether to consider missing values of the target variable as nonevents
Format Option	
RAW=	Specifies whether to process variables by using their raw values or formatted values

You can specify the following *options*:

BINMISSING=TRUE | FALSE
specifies whether to use missing values of the input variables for the binning process. You can specify either of the following values:

TRUE places all missing values into a bin whose binId=0.

FALSE ignores all missing values.

By default, BINMISSING=TRUE.

DATA=*libref.data-table*
names the input data table for PROC BINNING to use. The default is the most recently created data table. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to the data, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 173.

data-table specifies the name of the input data table.

FUZZYCOMPARE=*value*
specifies the fuzzy comparison threshold that is used to determine the distinctness of numeric values. The range for this option is 0 to 1E–5.

By default, FUZZYCOMPARE=1E–10.

METHOD=BUCKET | CUTPTS(*numlist***) | QUANTILE | TREE <***options***> | WINSOR(RATE=***rate***)**
specifies which binning method to use.

You can specify one of the following methods:

BUCKET
uses the bucket binning method. Only interval input variables are supported for this binning method.

CUTPTS(*numlist***)**
uses the cutpoints binning method and specifies the lower and upper bounds of the bin. The numbers in *numlist* specify the upper bounds of each bin except the last bin (whose upper bound is positive infinity). Each number in *numlist* also implies the lower bound of the next bin; the lower bound of the first bin is negative infinity. Only interval input variables are supported for this binning method.

QUANTILE
uses the quantile binning method. Only interval input variables are supported for this binning method.

TREE <*options***>**
uses the tree-based binning method. Both interval and nominal input variables are supported for this binning method. You can specify the following *options*:

CRITERIA <*options***>**
specifies the criteria for growing the tree for the tree-based binning method. You can specify the following *options*:

GAIN

ENTROPY
uses the gain in information (decrease in entropy) as the tree-growing criterion. This criterion is available only for a nominal target.

GAINRATIO

ENTROPYRATIO
uses the gain ratio in information (decrease in entropy ratio) as the tree-growing criterion. This criterion is available only for a nominal target. This is the default criterion for a nominal target.

GINI
uses the Gini statistic as the tree-growing criterion. This criterion is available only for a nominal target.

SSE
uses the sum of squared error as the tree-growing criterion. This criterion is available only for an interval target. This is the default criterion for an interval target.

DISTINCTCOUNTLIMIT=*integer*

MAXDISTINCTLEVELS=*integer*
specifies the maximum number of distinct levels that interval variables can have in order for them to be treated as nominal variables, where *integer* can be any integer between 2 and 30,000, inclusive. You can use the LEVEL= option in the INPUT statement to treat interval variables as nominal variables as long as their levels are less than *integer*.

By default, DISTINCTCOUNTLIMIT=5000.

INITBIN=*integer*
specifies the initial number of bins for interval input variables to start the tree-based binning process. The final number of bins created by PROC BINNING might be different than the number you specify. The value of *integer* can be any integer between 1 and 10,000, inclusive. By default, INITBIN=100.

INITMETHOD=BUCKET | QUANTILE
specifies the method to initialize the tree-based binning for interval input variables. You can specify either of the following values:

BUCKET uses the bucket binning method.

QUANTILE uses the quantile binning method.

By default, the bucket binning method is used.

LEAFSIZE=*integer*

MINNOBSINBIN=*integer*
specifies the allowed minimum number of observations per leaf for tree-based binning. The value of *integer* can be any integer between 1 and 30,000. There is no default. If you specify this option, the number of observations in each leaf will be larger than the specified *integer*.

MAXNBINS=*integer*
specifies the desired maximum number of bins for all variables. The value of *integer* can be any integer between 2 and 10,000, inclusive. If this option is specified, PROC BINNING tries to make sure that the final number of bins for all variables is no more than the specified *integer*. By default, MAXNBINS=5.

MINNBINS=*integer*
specifies the desired minimum number of bins for all variables. The value of *integer* can be any integer between 1 and 10,000, inclusive. If this option is specified, PROC BINNING tries to make sure that the final number of bins for all variables is no less than the specified *integer*. By default, MINNBINS=1.

WINSOR(RATE=*rate*)
uses the Winsorized binning method and specifies the *rate* that it uses. You must specify a *rate* between 0 and 0.5, exclusive. Only interval input variables are supported for this binning method.

MISSINGBINSTATS=TRUE | FALSE
specifies whether to use the bin that contains the missing values for binning analysis such as WOE and IV. You can specify either of the following values:

TRUE uses the bin that contains the missing values (binId=0) for binning analysis.

FALSE ignores the bin that contains the missing values.

By default, MISSINGBINSTATS=TRUE.

MISSINGEVALNONEVENT=TRUE | FALSE
specifies whether to consider missing values of the target variable as nonevents. You can specify either of the following values:

TRUE considers missing values of the target variable to be nonevents.

FALSE excludes missing values of the target variable from analysis.

By default, MISSINGEVALNONEVENT=FALSE.

NUMBIN=*integer*
specifies the global number of binning levels for all binning variables. The value of *integer* can be any integer between 2 and 1000, inclusive.

By default, NUMBIN=16.

In tree-based binning, this option is ignored.

PERCENTILETOLERANCE=*value*
specifies the tolerance for percentile computation.

By default, PERCENTILETOLERANCE=1E–5.

PERCENTILEMAXITERATIONS=*value*
specifies the maximum number of iterations for percentile computation.

By default, PERCENTILEMAXITERATIONS=10.

RAW=TRUE | FALSE
specifies whether to process variables by using their raw values or formatted values. You can specify either of the following values:

TRUE processes variables by using their raw values.

FALSE if a variable has format, processes it by using a formatted value.

By default, RAW=TRUE.

WOE(WOEADJUST=*number***)**
computes the weight of evidence and information values. In tree-based binning, this option is ignored. You can specify the following suboption:

WOEADJUST=*number*
specifies the adjustment factor for the weight-of-evidence calculation, where *number* is between 0 and 1, inclusive.

By default, WOEADJUST=0.5.

CODE Statement

CODE FILE=*filename* **;**

The CODE statement generates score code and stores it in a file that can be used for scoring purposes. Only one CODE statement is processed. If you specify multiple CODE statements, only the first one is used.

You must specify the following option:

FILE=*filename*
specifies the filename of the file to write the SAS score code to.

FREQ Statement

FREQ *variable* **;**

The *variable* in the FREQ statement identifies a numeric variable in the input data table that contains the frequency of occurrence of each observation. PROC BINNING treats each observation as if it appears f times, where f is the value of the FREQ *variable* for the observation. If f is not an integer, it is truncated to an integer. If f is less than 1 or missing, the observation is not used in the analysis. When you omit the FREQ statement, each observation is assigned a frequency of 1.

INPUT Statement

INPUT *variables* < / *options* > **;**

The INPUT statement names one or more *variables* as input variables for binning. PROC BINNING does not support duplicate variables. If the INPUT statement contains a duplicate variable, PROC BINNING takes only the first variable and provides a warning message.

When the values of the METHOD= option is BUCKET, QUANTILE, WINSOR, or CUTPTS, the specified *variables* must be interval variables. If nominal variables are provided, PROC BINNING stops and returns an error message.

When the values of the METHOD= option is TREE, the specified *variables* can be either interval or nominal variables.

You can specify the following *options* after a slash (/):

NUMBIN=*integer*

specifies the number of binning levels for all binning variables in the current INPUT statement. The value of *integer* can be any integer between 2 and 1,000, inclusive.

The number of binning levels that you specify in an INPUT statement overwrites the global number of binning levels, which is specified in the NUMBIN= option in the PROC BINNING statement (or is 16 by default).

In tree-based binning, this option is ignored.

LEVEL=INTERVAL | NOMINAL

specifies whether to treat the specified variables as interval or nominal. You can specify the following values:

INTERVAL

INT

treats all variables that are specified in this INPUT statement as interval. If the INPUT statement contains any character variables, PROC BINNING stops and returns an error message.

NOMINAL

NOM

treats all variables that are specified in this INPUT statement as nominal. If you specify LEVEL=NOMINAL and the variables in the INPUT statement are interval when the value of the METHOD= option is BUCKET, QUANTILE, WINSOR, or CUTPTS, PROC BINNING prints a warning and continues to treat the variables in the INPUT statement as interval variables.

By default, LEVEL=INTERVAL.

OUTPUT Statement

OUTPUT OUT=*libref.data-table* < *options* > **;**

The OUTPUT statement enables you to write an output table back to the CAS server by creating an output data table that contains the results of PROC BINNING. For tree-based binning of class inputs, the OUTPUT statement also enables you to write an output bin-mapping data table that contains the following three columns: class input variables, their levels, and their associated bin numbers. If you do not specify the OUTPUT statement, the write-back process is skipped. This is useful when you want only the ODS output. By default, the output table contains the new binned variables. If you specify the COPYVARS= option, the output table contains the "copied variables" and the "binned variables". The output bin-mapping data table contains only three columns, and the COPYVARS= option does not apply to it. If you specify multiple OUTPUT statements, PROC BINNING stops and produces an error message.

NOTE: If an input variable value is missing, then the binning output level value is 0.

You must specify one of the following options:

OUT=*libref.data-table*
names the output data table for PROC BINNING to use. *libref.data-table* is a two-level name, where *libref* refers to the library, and *data-table* specifies the name of the output data table. For more information about this two-level name, see the DATA= option and the section "Using CAS Sessions and CAS Engine Librefs" on page 173.

OUTLEVELBINMAP=*libref.data-table*
names the output bin-mapping data table for PROC BINNING to use. *libref.data-table* is a two-level name, where *libref* refers to the library, and *data-table* specifies the name of the output data table. For more information about this two-level name, see the DATA= option and the section "Using CAS Sessions and CAS Engine Librefs" on page 173.

You can also specify the following *option*:

COPYVAR=*variable*
COPYVARS=(*variables***)**
lists one or more *variables* from the input data table to be transferred to the output data table.

TARGET Statement

TARGET *variable* / < *options* > **;**

When the value of the METHOD= option is BUCKET, QUANTILE, WINSOR, or CUTPTS, the TARGET statement names the *variable* that PROC BINNING uses to calculate the weight of evidence and information value. If you specify multiple TARGET statements, only the first one is used.

When the value of the METHOD= option is TREE, the TARGET statement names the *variable* that PROC BINNING uses to create trees.

You can specify the following *options* after a slash (/):

EVENT=*"category"*
specifies the target event category that PROC BINNING uses to calculate the weight of evidence and information value in a quoted string of characters. You must specify this option if the WOE option is specified in the PROC BINNING statement.

LEVEL=INTERVAL | NOMINAL
specifies whether to treat the target variable as interval or nominal. You can specify the following values:

INTERVAL
INT
treats the target variable as interval.

NOMINAL
NOM
treats the target variable as nominal.

By default, LEVEL=INTERVAL.

WEIGHT Statement

WEIGHT *variable* **;**

The *variable* in the WEIGHT statement is used as a weight to perform a weighted analysis of the data. Observations that have nonpositive or missing weights are not included in the analysis. The values of the WEIGHT variable can be nonintegral and are not truncated. When you omit the WEIGHT statement, all observations that are used in the analysis are assigned a weight of 1.

Details: BINNING Procedure

Binning Computation and Formulas

For variable x, assume that the data are represented by $\{x_i\}$, where $i = 1, 2, \ldots, n$. Let $\min(x) = \min_{i \in \{1,\ldots,n\}} \{x_i\}$, and let $\max(x) = \max_{i \in \{1,\ldots,n\}} \{x_i\}$. The range of the variable is $\text{range}(x) = \max(x) - \min(x)$.

The computations for the various binning methods are as follows, where *numbin* is the value of the NUMBIN= option in the PROC BINNING statement:

- For bucket binning, the length of the bucket is

  $$L = \frac{\max(x) - \min(x)}{n}$$

  The split points are

  $$s_k = \min(x) + L \times k$$

  where $k = 1, 2, \ldots, \textit{numbin} - 1$.

  When the data are evenly distributed on the cloud, the time complexity for bucket binning is $\mathcal{O}(n)/(\textit{nodes} \times \textit{nThreads})$, where n is the number of observations, *nodes* is the number of computer nodes in the cloud, and *nThreads* is the number of CPUs on each node.

- For quantile binning, PROC BINNING calculates a quantile table P. Let $P = \{p_k\}$, where $k = 1, 2, \ldots, \textit{numbin}$. Then p_k is described as

  $$p_k = \begin{cases} 1.0/\textit{numbin} + p_{k-1} & \text{if } 0 < k < \textit{numbin} \\ 1.0 & \text{if } k = \textit{numbin} \end{cases}$$

  Quantile binning often requires data to be sorted in a particular way, and the sorting process usually consumes a significant amount of CPU time and memory. When the input data table is larger than the available memory, the sorting algorithm becomes more complicated. In distributed computing, data communications overhead also increases the sorting challenge. To avoid the time-consuming sorting process, the BINNING procedure uses an iterative projection method for quantile binning, which runs much faster than the sorting-based quantile binning method in most cases.

  After calculating the quantile table, PROC BINNING uses an iterative projection method to compute quantiles (percentiles) and uses these quantiles to create bins.

Quantile binning aims to assign the same number of observations to each bin. As a result, each bin should have a similar number of observations. Because PROC BINNING always assigns observations that have the same value to the same bin, quantile binning might create unbalanced bins if any variable has tied values. For example, if an observation whose value is x is assigned to bin k, then every observation whose value is x is assigned to bin k for this variable, and no observation whose value is x is assigned to the next bin, bin $k + 1$. Therefore, bin k might have more observations than bin $k + 1$, because the tied values at the boundaries between bin k and bin $k + 1$ are all assigned to bin k. That is, tied values at the boundaries between two bins are always assigned to the lower-numbered bin.

- Winsorized binning applies bucket binning to the Winsorized data table. Winsorization involves replacing lower and upper outliers (which are defined according to the RATE= suboption in the METHOD=WINSOR option in the PROC BINNING statement) by the lower and upper thresholds. For variable x, PROC BINNING uses a simple bucket sorting method to obtain the basic information. Let N be the number of buckets. For each bucket B_i, $i = 1, 2, \ldots, N$, PROC BINNING keeps following information:

  - c_i: count of x in B_i
  - $\min_i$: minimum value of x in B_i
  - $\max_i$: maximum value of x in B_i
  - $\sum x_i$: sum of x in B_i
  - $\sum x_i^2$: sum of x^2 in B_i

  The Winsorized statistics are computed first. After the minimum and maximum have been found, the bin lower and upper bounds are calculated the same way as in bucket binning.

  Let the tail count, wc, be $\mathrm{ceil}(\mathrm{Rate} \times n)$, and find the smallest I such that $\sum_{i=1}^{I} c_i \geq \mathrm{wc}$. Then the left tail count is $\mathrm{lwc} = \sum_{i=1}^{I} c_i$. Find the next I_l such that $\sum_{i=1}^{I_l} c_i > \mathrm{lwc}$. Therefore, the minimum value is $\mathrm{WinsorMin} = \min_{I_l}$. Similarly, find the largest I such that $\sum_{i=I}^{N} c_i \geq \mathrm{wc}$. The right tail count is $\mathrm{rwc} = \sum_{i=I}^{N} c_i$. Find the next I_r such that $\sum_{i=I_r}^{N} c_i > \mathrm{rwc}$. Then the maximum value is $\mathrm{WinsorMax} = \max_{I_r}$. The mean is calculated by the formula

$$\mathrm{WinsorMean} = \frac{\mathrm{lwc} \times \mathrm{WinsorMin} + \sum_{i=I_l}^{I_r} x_i + \mathrm{rwc} \times \mathrm{WinsorMax}}{n}$$

  The trimmed mean is calculated by the formula

$$\mathrm{TrimmedMean} = \frac{\sum_{i=I_l}^{I_r} x_i}{n - \mathrm{lwc} - \mathrm{rwc}}$$

NOTE: PROC BINNING reserves bin 0 for missing values.

NOTE: Whenever possible, PROC BINNING always tries to generate the specified number of bins. In some cases, some bins might have no observations in them.

Tree-Based Binning

Tree-based binning is different from other binning methods. The basic idea of tree-based binning is to bin (group) input variables by growing a one-level decision tree (or regression tree, when the target variable is not nominal) for each input variable. In the literature, the difference between a decision tree and a regression tree is often ignored (with some exceptions). Therefore, in the rest of this chapter, the term decision tree can be used to represent either a decision tree or a regression tree. To grow a decision tree, PROC BINNING uses a training data set to split the target variable space into nonoverlapping regions. These regions correspond to the terminal nodes of the tree, which are also known as leaves.

The tree growing is done by splitting; the splitting starts with all the observations, which are represented by the node at the top of the tree. The algorithm splits this parent node into child nodes in such a way that the values (levels) of the target variable within each child region are as similar as possible. The split is determined by finding the best split value that optimizes a specified criterion across the child nodes for each input variable. The goal of maximizing similarity is achieved by minimizing node impurity. For an interval target, the default and only available growing criterion is SSE (sum of squared error), which aims to minimize node impurity. For a nominal target, the default growing criterion is GAINRATIO (or ENTROPYRATIO as the alias), which measures the gain ratio in information (decrease in entropy ratio) with the goal of minimizing node impurity. For a nominal target, the other supported growing criteria are GAIN (or ENTROPY as the alias) and GINI, which measure the gain in information (decrease in entropy) and the Gini statistic with the goal to minimize node impurity.

Computing the Weight of Evidence and Information Value

PROC BINNING can compute the weight of evidence and the information value.

Weight of evidence (WOE) is a measure of how much the evidence supports or undermines a hypothesis. WOE measures the relative risk of an attribute of binning level. The value depends on whether the value of the target variable is a nonevent or an event. An attribute's WOE is defined as follows:

$$\mathrm{WOE}_{\mathrm{attribute}} = \ln\frac{p_{\mathrm{attr(nonevent)}}}{p_{\mathrm{attr(event)}}} = \ln\frac{N_{\mathrm{attr(nonevent)}}/N_{\mathrm{tot(nonevent)}}}{N_{\mathrm{attr(event)}}/N_{\mathrm{tot(event)}}}$$

The definitions of the quantities in the preceding formula are as follows:

- $N_{\mathrm{attr(nonevent)}}$: the number of nonevent records that exhibit the attribute
- $N_{\mathrm{tot(nonevent)}}$: the total number of nonevent records
- $N_{\mathrm{attr(event)}}$: the number of event records that exhibit the attribute
- $N_{\mathrm{tot(event)}}$: the total number of event records

To avoid an undefined WOE, an adjustment factor, x, is used:

$$\mathrm{WOE}_{\mathrm{attribute}} = \ln \frac{(N_{\mathrm{attr(nonevent)}} + x)/N_{\mathrm{tot(nonevent)}}}{(N_{\mathrm{attr(event)}} + x)/N_{\mathrm{tot(event)}}}$$

You can use the WOEADJUST= option to specify a value between [0, 1] for x. By default, x is 0.5.

The information value (IV) is a weighted sum of the WOE of the characteristic's attributes. The weight is the difference between the conditional probability of an attribute for an event and the conditional probability of that attribute for a nonevent. In the following formula of IV, m is the number of bins of a variable:

$$\mathrm{IV} = \sum_{i=1}^{m} \left(\frac{N_{\mathrm{attr(nonevent)}}}{N_{\mathrm{tot(nonevent)}}} - \frac{N_{\mathrm{attr(event)}}}{N_{\mathrm{tot(event)}}} \right) \times \mathrm{WOE}_i$$

An information value can be any real number. Generally speaking, the higher the information value, the more predictive an attribute is likely to be.

Displayed Output

The following sections describe the output that PROC BINNING produces. The output is organized into various tables.

Bin Details

The "Bin Details" table displays some parameters and data information, which includes the binning variable, bin ID, bin lower bound, bin upper bound, bin width, number of observations in that bin, and some statistics of that bin (such as mean, standard deviation, minimum, and maximum). If you specify the WOE option in the PROC BINNING statement, this table also includes the target event count, weight of evidence, and information value of that bin.

Trimmed Statistics

The "Trimmed Statistics" table displays the binning variable, trimmed minimum, maximum, mean, standard error mean, left tail, left tail percentage, right tail, and right tail percentage. PROC BINNING generates this table if you specify the METHOD=WINSOR option in the PROC BINNING statement.

Winsorized Statistics

The "Winsorized Statistics" table displays the binning variable, Winsorized minimum, maximum, mean, standard error mean, left tail, left tail percentage, right tail, and right tail percentage. PROC BINNING generates this table if you specify the METHOD=WINSOR option in the PROC BINNING statement.

Variable Information Value

The "Variable Information Value (IV)" table displays the information value for each variable. PROC BINNING generates this table if you specify the WOE option in the PROC BINNING statement.

Variable Transformation Information

The "Variable Transformation Information" table displays the transformation information for each variable, in tree-based binning. PROC BINNING generates this table if you specify the METHOD=TREE option in the PROC BINNING statement.

Variable Mapping Information

The "Variable Mapping Information" table displays how each binning variable name is mapped to each result variable name. This table includes two columns: Variable and Result Variable.

ODS Table Names

Each table created by the BINNING procedure has a name associated with it. You must use this name to refer to the table when you use ODS statements. The names of each table and a short description of the contents are listed in Table 5.2.

Table 5.2 ODS Tables Produced by PROC BINNING

Table Name	Description	Statement	Options
BinDetails	Basic binning information and parameters	PROC BINNING	Default
InfoValue	Information value for each variable	PROC BINNING	WOE
Trim	Trimmed statistics for the given variables	PROC BINNING	METHOD=WINSOR
VarMapping	Variable mapping information	PROC BINNING	Default
VarTransInfo	Transformation information for the specified variables	PROC BINNING	METHOD=TREE
Winsor	Winsor statistics for the given variables	PROC BINNING	METHOD=WINSOR

Examples: BINNING Procedure

The examples in this chapter are available in the GitHub repository located at https://github.com/sascommunities/doc-supplement-casstat.

NOTE: Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections "Using CAS Sessions and CAS Engine Librefs" on page 7 and "Loading a SAS Data Set onto a CAS Server" on page 7 in Chapter 2, "Shared Concepts."

Example 5.1: Quantile Binning

(View the complete code for this example (binning_ex1.sas) in the example repository.)

The following DATA step generates 1,000,000 observations of an ID variable (id) and two continuous variables (x1–x2). The mylib CAS library reference enables your client machine to communicate with the CAS session. These statements assume that your CAS engine libref is named mylib, but you can substitute any appropriately defined CAS engine libref.

```
data mylib.ex1;
   length id 8;
   call streaminit(12345);
   do id=1 to 1000000;
      x1 = rand("Uniform");
      x2 = rand("Uniform") * 10;
      output;
   end;
run;
```

The following statements demonstrate how to use PROC BINNING to perform the quantile binning:

```
proc binning data=mylib.ex1 numbin=10 method=quantile;
   input x1-x2;
   output out=mylib.out1;
run;
```

The DATA= option specifies the input data table. The NUMBIN= option requests that 10 bins be created for all binning variables. The METHOD= option requests that the quantile binning method be used. The INPUT statement names two continuous variables (x1–x2) as input variables for binning. The OUTPUT statement creates an OUTPUT data table to contain the results of PROC BINNING.

The "Bin Details" table in Output 5.1 shows the binning variable, bin ID, bin lower bound, bin upper bound, bin width, number of observations in that bin, and some statistics of that bin (such as mean, standard deviation, minimum, and maximum). When the binning method is quantile, PROC BINNING assigns the same number of observations to each bin for the input variables if possible.

Output 5.1.1 Bin Details

The BINNING Procedure

Bin Details									
Variable	Bin ID	Lower Bound	Upper Bound	Bin Width	Number of Observations	Mean	Standard Deviation	Minimum	Maximum
x1	Missing				0				
	1	-Infty	0.0995		99999	0.0497	0.0288	795E-9	0.0995
	2	0.0995	0.1998	0.1003	100000	0.1496	0.0289	0.0995	0.1998
	3	0.1998	0.2999	0.1001	100000	0.2498	0.0289	0.1998	0.2999
	4	0.2999	0.3997	0.0998	100000	0.3498	0.0289	0.2999	0.3997
	5	0.3997	0.4999	0.1002	100000	0.4497	0.0290	0.3997	0.4999
	6	0.4999	0.5997	0.0998	100000	0.5498	0.0288	0.4999	0.5997
	7	0.5997	0.6994	0.0997	100000	0.6495	0.0287	0.5997	0.6994
	8	0.6994	0.7996	0.1002	100000	0.7495	0.0289	0.6994	0.7996
	9	0.7996	0.8999	0.1003	100000	0.8497	0.0290	0.7996	0.8999
	10	0.8999	Infty		100001	0.9499	0.0289	0.8999	1.0000
x2	Missing				0				
	1	-Infty	1.0047		99999	0.5034	0.2895	975E-8	1.0047
	2	1.0047	2.0057	1.0011	100000	1.5041	0.2889	1.0047	2.0057
	3	2.0057	3.0011	0.9954	100000	2.5054	0.2874	2.0057	3.0011
	4	3.0011	4.0050	1.0039	100000	3.5041	0.2898	3.0011	4.0050
	5	4.0050	5.0069	1.0019	100001	4.5051	0.2891	4.0050	5.0069
	6	5.0069	6.0046	0.9977	99999	5.5068	0.2882	5.0069	6.0046
	7	6.0046	7.0039	0.9993	99999	6.5056	0.2884	6.0046	7.0039
	8	7.0039	7.9989	0.9950	100001	7.5002	0.2870	7.0039	7.9989
	9	7.9989	9.0017	1.0028	100000	8.5010	0.2898	7.9989	9.0016
	10	9.0017	Infty		100001	9.5010	0.2881	9.0017	10.000

Example 5.2: Winsorized Binning

(View the complete code for this example (binning_ex2.sas) in the example repository.)

The following DATA step generates 10,000 observations of an ID variable (id) and three continuous variables (x1–x3). The mylib CAS library reference enables your client machine to communicate with the CAS session.

```
data mylib.ex2;
   length id 8;
   call streaminit(12345);
   do id=1 to 10000;
      x1 = rand("Uniform");
      x2 = rand("Uniform") * 10;
      x3 = rand("Uniform") * 100;
      output;
   end;
run;
```

For Winsorized binning, PROC BINNING provides bucket binning and basic Winsorized statistical information for the input data.

The following statements include the WINSOR(RATE=0.05) option and generate tables for Winsorized and trimmed statistics:

```
proc binning data=mylib.ex2 numbin=10 method=winsor(rate=0.05);
   input x1-x3;
   output out=mylib.out2;
run;
```

The preceding statements generate Output 5.2.1 through Output 5.2.4.

Output 5.2.1 Bin Details

The BINNING Procedure

Bin Details

Variable	Bin ID	Lower Bound	Upper Bound	Bin Width	Number of Observations	Mean	Standard Deviation	Minimum	Maximum
x1	Missing				0				
	1	-Infty	0.1410		1479	0.0815	0.0304	0.0509	0.1410
	2	0.1410	0.2312	0.0902	849	0.1865	0.0259	0.1411	0.2311
	3	0.2312	0.3214	0.0902	903	0.2759	0.0252	0.2312	0.3214
	4	0.3214	0.4115	0.0902	906	0.3671	0.0258	0.3214	0.4115
	5	0.4115	0.5017	0.0902	875	0.4560	0.0264	0.4118	0.5016
	6	0.5017	0.5918	0.0902	920	0.5453	0.0261	0.5017	0.5918
	7	0.5918	0.6820	0.0902	871	0.6381	0.0254	0.5919	0.6819
	8	0.6820	0.7722	0.0902	892	0.7287	0.0264	0.6820	0.7719
	9	0.7722	0.8623	0.0902	888	0.8184	0.0258	0.7722	0.8621
	10	0.8623	Infty		1417	0.9234	0.0298	0.8625	0.9525
x2	Missing				0				
	1	-Infty	1.3989		1402	0.7818	0.2975	0.4952	1.3982
	2	1.3989	2.3026	0.9037	863	1.8443	0.2552	1.4027	2.3002
	3	2.3026	3.2063	0.9037	865	2.7466	0.2535	2.3047	3.2062
	4	3.2063	4.1100	0.9037	861	3.6638	0.2662	3.2112	4.1081
	5	4.1100	5.0136	0.9037	886	4.5545	0.2525	4.1110	5.0121
	6	5.0136	5.9173	0.9037	901	5.4611	0.2643	5.0139	5.9165
	7	5.9173	6.8210	0.9037	916	6.3894	0.2584	5.9181	6.8203
	8	6.8210	7.7247	0.9037	917	7.2690	0.2565	6.8217	7.7234
	9	7.7247	8.6284	0.9037	921	8.1685	0.2588	7.7248	8.6282
	10	8.6284	Infty		1468	9.2379	0.2974	8.6301	9.5321
x3	Missing				0				
	1	-Infty	14.290		1386	8.1918	2.9856	5.3208	14.283
	2	14.290	23.258	8.9688	874	18.813	2.5927	14.290	23.234
	3	23.258	32.227	8.9688	871	27.778	2.5755	23.275	32.219
	4	32.227	41.196	8.9688	910	36.659	2.5792	32.232	41.193
	5	41.196	50.165	8.9688	887	45.551	2.5481	41.202	50.162
	6	50.165	59.134	8.9688	911	54.660	2.5880	50.173	59.133
	7	59.134	68.102	8.9688	885	63.711	2.5745	59.149	68.093
	8	68.102	77.071	8.9688	946	72.761	2.6008	68.108	77.059
	9	77.071	86.040	8.9688	918	81.526	2.5287	77.071	86.039
	10	86.040	Infty		1412	92.093	2.9715	86.043	95.009

Output 5.2.2 Winsorized Statistics

Winsorized Statistics

Variable	Minimum	Maximum	Mean	Standard Deviation	N Left Tail	Percent Left Tail	N Right Tail	Percent Right Tail
x1	0.0509	0.9525	0.5002	0.2867	499	4.99	500	5.00
x2	0.4952	9.5321	5.0777	2.8743	499	4.99	500	5.00
x3	5.3208	95.009	50.564	28.360	499	4.99	500	5.00

Output 5.2.3 Trimmed Statistics

Trimmed Statistics

Variable	Minimum	Maximum	Mean	Standard Deviation	N Left Tail	Percent Left Tail	N Right Tail	Percent Right Tail
x1	0.0509	0.9525	0.5000	0.2623	499	4.99	500	5.00
x2	0.4952	9.5321	5.0843	2.6290	499	4.99	500	5.00
x3	5.3208	95.009	50.603	25.891	499	4.99	500	5.00

Output 5.2.4 Variable Mapping Information

Variable Mapping Information

Variable	Result Variable
x1	BIN_x1
x2	BIN_x2
x3	BIN_x3

Example 5.3: Bucket Binning and Weight-of-Evidence Computation

(View the complete code for this example (binning_ex3.sas) in the example repository.)

The following DATA step generates a data table that contains 10 observations of a target variable (y), three continuous variables (x0–x2), and some other variables. The mylib CAS library reference enables your client machine to communicate with the CAS session.

```
data mylib.ex3;
   input cl1 $ x0  x1  x2  y $ freq id;
   datalines;
a  2  .  7  n  2  1
a  2  2  6  .  3  2
a  3  0  1  o  0  3
c  2  3  7  y  .  4
c  2  .  4  n  -5 5
a  3  6  7  n  3  6
b  1  4  4  y  4  7
b  2  5  6  y  3  8
b  1  6  4  o  1  9
b  2  3  2  n  3  10
;
```

The following statements show how you can use the BINNING procedure to perform bucket binning and compute the WOE and the information value (IV):

```
proc binning data=mylib.ex3 numbin=5 woe;
   input x1/numbin=4;
   input x2;
   target y/event="y";
   output out=mylib.out3;
run;
```

The DATA= option specifies the input data table. The WOE option enables computation of the weight of evidence and information values with WOEADJUST=0.5 by default. The first INPUT statement names one continuous variable (x1) as the first input variable for binning with four bins. The second INPUT statement names another continuous variable (x2) as the second input variable for binning with five bins, as specified in the NUMBIN= global option. The TARGET statement names the variable (y) that PROC BINNING uses to calculate the weight of evidence, and the EVENT= option specifies the target event category in a quoted string. The OUTPUT statement creates an OUTPUT data table to contain the results of PROC BINNING.

The preceding statements generate Output 5.3.1 through Output 5.3.2.

Output 5.3.1 Bin Details

The BINNING Procedure

Bin Details

Variable	Bin ID	Lower Bound	Upper Bound	Bin Width	Number of Observations	Mean	Standard Deviation	Minimum	Maximum	Event Count	Weight of Evidence	Information Value
x1	Missing				2					0	0.9163	0.3054
	1	-Infty	1.5000		1	0		0	0	0	0.4055	0.0676
	2	1.5000	3	1.5000	1	2		2	2	0	.	.
	3	3	4.5000	1.5000	3	3.3333	0.5774	3	4	2	-1.386	0.6931
	4	4.5000	Infty		3	5.6667	0.5774	5	6	1	0	0
x2	Missing				0					0	.	.
	1	-Infty	2.2000		2	1.5000	0.7071	1	2	0	0.9163	0.3054
	2	2.2000	3.4000	1.2000	0					0	.	.
	3	3.4000	4.6000	1.2000	3	4	0	4	4	1	0	0
	4	4.6000	5.8000	1.2000	0					0	.	.
	5	5.8000	Infty		5	6.6000	0.5477	6	7	2	-0.693	0.2310

Output 5.3.2 Variable Information Value

Variable Information Value

Variable	Information Value
x1	1.0662
x2	0.5365

Example 5.4: Cutpoint Binning

(View the complete code for this example (binning_ex4.sas) in the example repository.)

The following DATA step generates a data table that contains 10 observations of a target variable (y), two continuous variables (x1, x2), and some other variables:

```
data mylib.ex4;
   input cl1 $ cl2  x1  x2  y  freq id;
   datalines;
a     2    3   7  9   2   1
a     2    2   6  8   3   2
a     3    0   1  5   0   3
c     2    3   7  4   .   4
c     2    .   4  8   -5  5
a     3    6   7  5   3   6
b     1    4   4  8   4   7
b     2    5   6  3   3   8
b     1    6   4  8   1   9
b     2    3   2  6   3   10
;
```

The following statements show how you can use the BINNING procedure to perform cutpoint binning:

```
proc binning data=mylib.ex4 numbin=4 method=cutpts;
   input x2/cutpts(2, 2.3, 4.5);
   input x1/numbin=3 cutpts(3.1, 5);
run;
```

The DATA= option specifies the input data table. The METHOD= option requests that the cutpoint binning method be used. The first INPUT statement names one continuous variable x2 as the first input variable for binning, with four bins specified by the NUMBIN= global option. The second INPUT statement names another continuous variable x1 as the second input variable for binning with three bins. For the first input variable x2, 2, 2.3, 4.5 (and infinity) are used as the upper bounds for its four bins. For the second input variable x1, 3.1, 5 (and infinity) are used as the upper bounds for its three bins.

Output 5.4.1 shows the "Bin Details" table.

Output 5.4.1 Bin Details

The BINNING Procedure

Bin Details

Variable	Bin ID	Lower Bound	Upper Bound	Bin Width	Number of Observations	Mean	Standard Deviation	Minimum	Maximum
x2	Missing				0				
	1	-Infty	2		1	1		1	1
	2	2	2.3000	0.3000	1	2		2	2
	3	2.3000	4.5000	2.2000	3	4	0	4	4
	4	4.5000	Infty		5	6.6000	0.5477	6	7
x1	Missing				1				
	1	-Infty	3.1000		5	2.2000	1.3038	0	3
	2	3.1000	5	1.9000	1	4		4	4
	3	5	Infty		3	5.6667	0.5774	5	6

Example 5.5: Tree-Based Binning

(View the complete code for this example (binning_ex5.sas) in the example repository.)

The following statements show how you can use the BINNING procedure to perform tree-based binning:

```
data mylib.hmeq;
    set sampsio.hmeq;
run;

proc binning data=mylib.hmeq method=tree;
 target bad/level=int;
 input mortdue/level = int;
 input job/ level = nom;
 output outlevelbinmap=mylib.outlevel;
run;

proc print data=mylib.outlevel;
run;
```

The DATA= option specifies the input data table. The METHOD= option requests that tree-based binning method be used. The TARGET statement specifies bad as the target variable and requests that it be treated as an interval variable. The first INPUT statement names one variable, mortdue, and requests that it be treated as interval. The second INPUT statement names one variable, job, and requests that it be treated as nominal.

Output 5.5.1 shows the "Bin Details" table. Compared to the "Bin Details" tables in previous examples, this table has an extra column, "N Levels". The reason is that other binning methods can handle only interval variables, but tree-based binning can handle both interval and nominal variables. The "N Levels" column shows the number of levels of this variable that this bin contains. As a result, for interval variables, this column is empty.

Output 5.5.1 Bin Details

The BINNING Procedure

Bin Details

Variable	Bin ID	Lower Bound	Upper Bound	Bin Width	N Levels	Number of Observations	Mean	Standard Deviation	Minimum	Maximum
MORTDUE	Missing					518				
	1	-Infty	37837			933	23701	9087.0	2063	37750
	2	37837	73611	35774		2309	56416	9712.4	37843	73600
	3	73611	105410	31799		1283	87448	9032.3	73624	105328
	4	105410	145158	39749		575	124127	11706	105422	145127
	5	145158	Infty			342	191406	48765	145324	399550
JOB	Missing				1	279				
	1				2	302				
	2				1	767				
	3				1	2388				
	4				1	1276				
	5				1	948				

Output 5.5.2 shows the "Transformation Information" table.

Output 5.5.2 Transformation Information

Transformation Information					
Variable	N	N Miss	N Bins	Importance	Relative Importance
MORTDUE	5442	518	6	8.6977	0.5053
JOB	5681	279	6	17.214	1

Output 5.5.3 shows the "Output Bin Mapping" data table.

Output 5.5.3 Output Bin Mapping Data Table

Obs	Variable	Level	BinId
1	JOB		0
2	JOB	Sales	1
3	JOB	Self	1
4	JOB	Mgr	2
5	JOB	Other	3
6	JOB	ProfExe	4
7	JOB	Office	5

Chapter 6
CAEFFECT Procedure

Contents

Overview: CAEFFECT Procedure **200**
PROC CAEFFECT Features 201
PROC CAEFFECT Compared with Other SAS Procedures 201
Using CAS Sessions and CAS Engine Librefs 202
Getting Started: CAEFFECT Procedure **203**
Syntax: CAEFFECT Procedure **207**
PROC CAEFFECT Statement 208
BY Statement 210
DIFFERENCE Statement 210
DISPLAY Statement 211
DISPLAYOUT Statement 212
FREQ Statement 213
OUTCOMEMODEL Statement 213
OUTCOMEVAR Statement 213
POM Statement 215
TREATVAR Statement 215
WEIGHT Statement 216
Details: CAEFFECT Procedure **216**
Missing Values 216
Potential Outcomes and Effect Definitions 216
Estimation Methods 219
Standard Errors and Confidence Limits 227
Displayed Output 229
ODS Table Names 230
Examples: CAEFFECT Procedure **231**
Example 6.1: Estimation by Inverse Probability Weighting 231
Example 6.2: Estimation by Regression Adjustment 235
Example 6.3: Estimation by Doubly Robust Methods 239
References **243**

Overview: CAEFFECT Procedure

The CAEFFECT procedure implements estimation methods that enable you to study the effect of a treatment variable on an outcome variable. When appropriate causal assumptions are satisfied, the estimates that you obtain by using this procedure correspond to potential outcome mean estimates, and the differences between these estimates have a causal interpretation. For more information about the framework that is used to define causal effects and the necessary assumptions, see the section "Potential Outcomes and Effect Definitions" on page 216.

The estimation methods that PROC CAEFFECT implements are designed primarily for use with observational data (that is, data from nonrandomized or imperfectly randomized experiments), in which confounding variables are likely to be present. In the presence of confounding variables, statistical adjustments are necessary to obtain unbiased estimates of the effect of interest. The procedure enables you to adjust for confounding variables by using models for the treatment variable, the outcome variable, or both. By specifying predicted probabilities for the treatment assignment, also referred to as propensity scores, you can use inverse probability weighting to adjust for confounding. By specifying an outcome model or predicted potential outcomes, you can use regression adjustment to adjust for confounding. Finally, you can combine models for the outcome and treatment variables in so-called doubly robust estimation methods. In particular, you can use the procedure to perform doubly robust estimation by using either augmented inverse probability weighting (AIPW) or targeted maximum likelihood estimation (TMLE). For more information about the estimation methods that this procedure supports, see the section "Estimation Methods" on page 219.

PROC CAEFFECT is a model-agnostic tool for effect estimation. Although the procedure itself does not support the modeling of the treatment or outcome variable, it makes statistical adjustments for confounding variables by using precomputed model predictions that you provide as input. For methods that require an outcome model, you can provide a previously fit model for the outcome in the form of an analytic store, which the procedure uses to obtain the appropriate counterfactual outcome predictions. In this way, the procedure enables you to use the modeling technique of your choice for confounding adjustment.

Although the model-agnostic approach that this procedure supports gives you a flexible tool for estimating potential outcome means and effects, the same flexibility does not extend to variance estimation and statistical inference. In general, to obtain valid statistical inference, you must assess model-dependent conditions that are beyond the scope of this procedure. In this way, PROC CAEFFECT's approach differs from other model-specific approaches to causal effect estimation, including those that are implemented for binary treatments by the CAUSALTRT procedure in SAS/STAT software and the DEEPCAUSAL procedure in SAS Econometrics software. In particular, during variance estimation, these model-specific approaches to effect estimation can account for dependencies that are introduced when the data that you use to estimate the potential outcome means are the same data that you use to estimate the treatment and outcome models. In the model-agnostic approach of PROC CAEFFECT, no similar adjustment can be made, and typically some form of data splitting is required in order to obtain valid statistical inference. For more information about the conditions that are required for valid such inference, see the section "Standard Errors and Confidence Limits" on page 227.

PROC CAEFFECT Features

PROC CAEFFECT provides the following features:

- estimation of potential outcome means by using inverse probability weighting, regression adjustment, augmented inverse probability weighting, and targeted maximum likelihood estimation (TMLE)
- estimation of potential outcome means conditional on a treatment level by using inverse probability weighting or regression adjustment
- estimation of effects on the difference scale
- support for categorical treatments
- support for continuous, categorical, and binomial outcomes
- input of previously fit models for the outcome variable in the form of an analytic store

Because PROC CAEFFECT runs on SAS Cloud Analytic Services (CAS), it also does the following:

- enables you to run on a cluster of machines that distribute the data and the computations
- enables you to run in single-machine mode
- exploits available cores and concurrent threads

PROC CAEFFECT Compared with Other SAS Procedures

The CAUSALTRT procedure in SAS/STAT and the DEEPCAUSAL procedure in SAS Econometrics both support the estimation of total causal effects by analyzing observational data. Both procedures are limited to the analysis of binary treatments. PROC CAUSALTRT adjusts for confounding variables by using models for the treatment variable, the outcome variable, or both. It uses generalized linear models when adjusting for confounding and fits these models itself. PROC DEEPCAUSAL adjusts for confounding variables by using models for the treatment and outcome variables that it fits by using deep neural networks.

The CAEFFECT procedure supports the estimation of total effects by analyzing observational data. The procedure supports the analysis of binary or categorical treatments. The estimation methods that the procedure supports adjust for confounding variables by using models for the treatment variable, the outcome variable, or both. Unlike the CAUSALTRT and DEEPCAUSAL procedures, PROC CAEFFECT does not directly fit any statistical models. Instead, it adjusts for confounding by using predictions from precomputed models. In this way, the procedure provides a model-agnostic tool for effect estimation and enables you to use the modeling technique of your choice for confounding adjustment.

The CAUSALTRT and DEEPCAUSAL procedures both support valid statistical inference for total effect estimates. The procedures are able to support valid inference because they fit the models that are used to adjust for confounding. During the variance estimation for the effect, the procedures can make model-specific adjustments that account for dependencies that are introduced when the same data are used to estimate the models, the potential outcome means, and the effect. The model-agnostic approach to effect estimation

that PROC CAEFFECT implements cannot perform any model-specific adjustments during the variance estimation. Typically, some form of data splitting and the assessment of model-dependent conditions are required in order to obtain valid statistical inference by using PROC CAEFFECT. Assessing model-dependent conditions and ensuring the use of proper splitting are beyond the scope of this procedure.

Using CAS Sessions and CAS Engine Librefs

SAS Cloud Analytic Services (CAS) is the analytic server and associated cloud services in SAS Viya. This section describes how to create a CAS session and set up a CAS engine libref that you can use to connect to the CAS session. It assumes that you have a CAS server already available; contact your system administrator if you need help starting and terminating a server. This CAS server is identified by specifying the host on which it runs and the port on which it listens for communications. To simplify your interactions with this CAS server, the host information and port information for the server are stored as SAS option values that are retrieved automatically whenever this CAS server needs to be accessed. You can examine the host and port values for the server at your site by using the following statements:

```
proc options option=(CASHOST CASPORT);
run;
```

In addition to starting a CAS server, your system administrator might also have created a CAS session and a CAS engine libref for your use. You can define your own sessions and CAS engine librefs that connect to the CAS server as shown in the following statements:

```
cas mysess;
libname mylib cas sessref=mysess;
```

The CAS statement creates the CAS session named mysess, and the LIBNAME statement creates the mylib CAS engine libref that you use to connect to this session. It is not necessary to explicitly name the CASHOST and CASPORT of the CAS server in the CAS statement, because these values are retrieved from the corresponding SAS option values.

If you have created the mysess session, you can terminate it by using the TERMINATE option in the CAS statement as follows:

```
cas mysess terminate;
```

For more information about the CAS and LIBNAME statements, see the section “Introduction to Shared Concepts” on page 6 in Chapter 2, “Shared Concepts.”

Getting Started: CAEFFECT Procedure

(View the complete code for this example at https://github.com/sascommunities/doc-supplement-casstat/tree/main/Examples/caeffectgs1.sas.)

NOTE: Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections "Using CAS Sessions and CAS Engine Librefs" on page 7 and "Loading a SAS Data Set onto a CAS Server" on page 7 in Chapter 2, "Shared Concepts."

This section illustrates some of the basic features of the CAEFFECT procedure. The example uses the School data set available in the examples for the CAUSALTRT procedure in SAS/STAT software. The data for this example are a subset of data from the National Educational Longitudinal Study of 1988 in Murnane and Willett (2011). Demographic information was collected from the students and their families, along with standardized test scores in 1988 and in follow-up years.

As in the PROC CAUSALTRT example, the question of interest in this example is what effect attending a Catholic high school has on a student's performance in mathematics, as measured by the student's score on the mathematics portion of the standardized tests in 1992. The effect of interest is estimated in this example by using the targeted maximum likelihood estimation (TMLE) method, which is doubly robust.

The following DATA step creates the data table School in your CAS session:

```
data mylib.School;
   input
   Income $ FatherEd $ MotherEd $ Math BaseMath  Catholic $;
   datalines;
   Middle HighSchool HighSchool 49.77 50.27 Yes
   Middle Unknown Unknown 59.84 51.52 Yes
   Middle NoHighSchool Postsecondary 50.38 47.56 Yes
   Middle Unknown Unknown 45.03 46.60 Yes
   High HighSchool HighSchool 54.26 60.22 Yes

   ... more lines ...

   High College College 56.60 60.15 Yes
   High College SomeSecondary 56.29 59.53 Yes
   High College Postsecondary 58.16 57.06 Yes
   High College SomeSecondary 63.57 63.51 Yes
;
```

These statements assume that your CAS engine libref is named mylib, but you can substitute any appropriately defined CAS engine libref.

The School data table consists of records for 5,671 high school students whose 1988 total family income was at most $75,000. The variables in the data are as follows:

- BaseMath: student's score on the mathematics portion of a standardized test in 1988
- Catholic: Yes, if a student attended a Catholic high school; No, otherwise
- FatherEd: highest level of education completed by the student's father, with six possible levels

- Income: classification based on total family income, with values of Low, Middle, and High
- Math: student's score on the mathematics portion of a standardized test in 1992
- MotherEd: highest level of education completed by the student's mother, with six possible levels

For this example, it is assumed that the variables BaseMath, FatherEd, MotherEd, and Income make up a valid adjustment set that you can use to obtain an unbiased estimate of the effect of interest. For more information about the causal assumptions that are required for valid effect estimation, see the section "Potential Outcomes and Effect Definitions" on page 216.

The TMLE method adjusts for the confounding variables by using models for both the treatment variable Catholic and the outcome variable Math. PROC CAEFFECT does not fit the required outcome and treatment models. Instead, the procedure performs adjustments that are based on predictions from previously fit models, enabling you to use the modeling technique of your choice to adjust for confounders.

In this example, the model for the treatment variable is fit by using the LOGSELECT procedure, as follows:

```
proc logselect data=mylib.School;
   class FatherEd MotherEd Income;
   model Catholic( event='Yes' ) = FatherEd MotherEd Income;
   output out=mylib.schoolEst pred=pTrt copyvars=(_All_);
run;
```

The PRED= option in the OUTPUT statement is used to name the variable pTrt, which contains the predicted event probability of attending a Catholic high school. The _ALL_ keyword is used in the COPYVARS= option to copy all variables from the input data table to the output data table schoolEst. The following DATA step creates the variable pCnt, which contains the predicted probability of not attending a Catholic high school:

```
data mylib.schoolEst;
   set mylib.schoolEst;
   pCnt = 1 - pTrt;
run;
```

An alternative way to obtain the predicted treatment probabilities is to save the fitted treatment model as an analytic store and use the ASTORE procedure. This approach is demonstrated in "Example 6.1: Estimation by Inverse Probability Weighting" on page 231.

For estimation methods that require modeling of the outcome variable, you can provide a previously fit model for the outcome in the form of an analytic store. PROC CAEFFECT uses this analytic store to obtain the required counterfactual predictions. The saved outcome model that you provide as input must use the confounding variables and the treatment variable as predictors. For this example, PROC BART is used to fit a Bayesian additive regression trees (BART) model for the outcome variable. BART models are nonparametric and use samples of a sum-of-trees ensemble to predict the conditional mean of a response variable. For more information about BART models, see the section "Details: BART Procedure" on page 154 in Chapter 4, "BART Procedure." The following statements use PROC BART to fit an outcome model that is saved in an analytic store named bartOutMod:

```
proc bart data=mylib.schoolEst nMC=200 nTree=100 seed=2455;
   class Income FatherEd MotherEd Catholic;
   model Math = BaseMath Income FatherEd MotherEd Catholic;
   store out=mylib.bartOutMod;
```

```
run;
```

The following statements use PROC CAEFFECT to estimate potential outcome means and their difference by using the TMLE method:

```
proc caeffect data=mylib.schoolEst;
   treatvar Catholic;
   outcomevar Math;
   outcomemodel restore=mylib.bartOutMod predname=P_Math;
   pom treatlev='Yes' treatprob=pTrt;
   pom treatlev='No'  treatprob=pCnt;
   difference evtLev='Yes';
run;
```

When you use PROC CAEFFECT, the TREATVAR statement and at least one POM statement are required. In the TREATVAR statement, you identify the treatment variable for the analysis. You use a POM statement to specify a potential outcome of interest and to specify any variables in the input data table that contain predicted values that the estimation method requires. You specify the outcome variable and any outcome variable options by using the OUTCOMEVAR statement. The TMLE method requires the observed outcome values.

You must use the TREATLEV= option in each POM statement. You use this option to specify the level of treatment that defines the potential outcome of interest. For estimation methods that require predicted treatment probabilities, you specify the variables that contain these values by using the TREATPROB= option.

For this example, you use the OUTCOMEMODEL statement to specify the BART model that has previously been fit to the data. PROC CAEFFECT uses the saved model to compute the predicted counterfactual outcome values for the outcome variable Math that the TMLE method requires. When you use the OUTCOMEMODEL statement, you use the RESTORE= option to specify the model that is saved as an analytic store, and you use the PREDNAME= option to specify the name of the variable that contains the model predictions. If you do not know the name of the variables that are created by the analytic store, you can use the DESCRIBE statement in PROC ASTORE and check the "Output Variables" table that it creates to find the variable names. As an alternative to using the OUTCOMEMODEL statement, you can use the PREDOUT= option in the POM statement to specify variables in the input data table that contain precomputed predictions for the counterfactual outcomes. An example of how you can use the alternative syntax is provided by "Example 6.2: Estimation by Regression Adjustment" on page 235.

The DIFFERENCE statement requests a difference between the potential outcome mean estimates. In this example, a single difference is requested, and the EVTLEV= option specifies that the treatment level 'Yes' is used as the event level. In general, you can specify multiple DIFFERENCE statements, and in each DIFFERENCE statement you can specify reference levels, event levels, or both.

The output from this analysis is presented in Figure 6.1 through Figure 6.4.

The "Model Information" table in Figure 6.1 summarizes important information about the analysis that you perform. It includes the name of the input data source, the treatment variable, the estimation method, and the name of the analytic store that is used to score the predicted counterfactual outcomes.

Figure 6.1 Model Information

The CAEFFECT Procedure

Model Information	
Data Source	SCHOOLEST
Treatment Variable	Catholic
Outcome Variable	Math
Outcome Type	Continuous
Estimation Method	TMLE
Outcome Model Store	BARTOUTMOD

Figure 6.2 displays two tables. The "Number of Observations" table displays the number of observations that are used in the analysis. The "Treatment Profile" table displays information about the levels of the treatment variable that appear in the input data table and the number of observations in each level.

Figure 6.2 Model Information

Number of Observations Read	5671
Number of Observations Used	5671

Treatment Profile	
Catholic	**Total Frequency**
No	5079
Yes	592

Figure 6.3 also displays two tables. The "POM Estimates" table indicates the level of treatment that defines the potential outcome and the estimate of its mean. The "POM Differences" table shows the requested difference in potential outcome mean estimates. The table also includes nonprinting columns that contain the potential outcome mean estimates for the event and reference levels; these columns are accessible if you save the table as an output data set by using the ODS OUTPUT statement or if you modify the table's template. If the necessary causal assumptions are satisfied, then the difference in the potential outcome mean estimates is an unbiased estimate of the average treatment effect (ATE). The ATE estimate of 1.467 indicates that attending a Catholic high school has a small positive effect on math scores. For more information about the assumptions that are required for valid causal effect estimation within a potential outcomes framework, see the section "Potential Outcomes and Effect Definitions" on page 216.

Figure 6.3 Potential Outcome Mean and Difference Estimates

POM Estimates	
Treatment Level	**Estimate**
Yes	52.37357
No	50.90671

POM Differences		
Treatment Levels		
Event	**Reference**	**Difference**
Yes	**No**	1.4669

Finally, PROC CAEFFECT displays the table shown in Figure 6.4, which shows the amount of time (in seconds) that it took to perform the various tasks in the analysis.

Figure 6.4 Procedure Timing

Task Timing

Task	Seconds	Percent
Parsing	0.03	1.79%
Data Preparation	0.03	1.63%
Outcome Prediction	1.53	90.20%
Targeting	0.10	5.66%
Estimation	0.00	0.13%
Miscellaneous	0.00	0.21%
Cleaning Up	0.01	0.36%
Total	1.70	100.00%

Syntax: CAEFFECT Procedure

The following statements are available in the CAEFFECT procedure:

PROC CAEFFECT < *options* > **;**
BY *variables* **;**
DIFFERENCE < **REFLEV=(***reference-list* **)** > < **EVTLEV=(***event-list* **)** > **;**
DISPLAY < *table-list* > < */options* > **;**
DISPLAYOUT *table-spec-list* < */options* > **;**
FREQ *variable* **;**
OUTCOMEMODEL RESTORE=*libref.data-table* **PREDNAME=***prediction-name* **;**
OUTCOMEVAR *outcome-variable* < **(EVENT='***event-level***'**) > < */ outcome-options* > **;**
OUTCOMEVAR *events / trials* < */ outcome-options* > **;**
POM TREATLEV='*treatment-level***'**
< **PREDOUT=***predicted-outcome-value* > < **TREATPROB=***predicted-treatment-probability* > **;**
TREATVAR *treatment-variable* < */treatment-options* > **;**
WEIGHT *variable* **;**

The PROC CAEFFECT statement, TREATVAR statement, and at least one POM statement are required. You can specify multiple POM and DIFFERENCE statements. The following sections describe the PROC CAEFFECT statement and then describe the other statements in alphabetical order.

PROC CAEFFECT Statement

PROC CAEFFECT < *options* > **;**

The PROC CAEFFECT statement invokes the procedure. Table 6.1 summarizes the available options in this statement. The options are described fully after the table.

Table 6.1 PROC CAEFFECT Statement Options

Option	Description
DATA=	Specifies the input data table
INFERENCE	Computes standard errors and confidence limits
METHOD=	Specifies the method to use for estimating potential outcome means
POMCOV	Displays the estimated covariance matrix of the potential outcome means
POMINFO	Displays a table that summarizes the potential outcome specifications
SCALEDIPWFLAG=	Specifies a value to be used to flag large weights

DATA=*libref.data-table*

names the input data table for PROC CAEFFECT to use. The default is the most recently created data table. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to the data, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 202.

data-table specifies the name of the input data table.

INFERENCE

computes standard errors and confidence intervals for the potential outcome mean and causal effect estimates. Note that because the CAEFFECT procedure takes a model-agnostic approach to causal effect estimation, the statistical validity of the inference depends on model-specific conditions that you must evaluate and that the procedure cannot verify. For more information about the conditions that are required for valid statistical inference, see the section "Standard Errors and Confidence Limits" on page 227.

METHOD=AIPW | IPW | REGADJ | TMLE

specifies the method to use for estimating potential outcome means (POMs). The required inputs for the procedure depend on the estimation method that you specify.

You can specify one of the following methods:

AIPW
estimates POMs by augmented inverse probability weighting. This method requires the observed values of the outcome, predicted treatment probabilities, and predicted counterfactual outcome values. For more information about the AIPW estimation method, see the section "Augmented Inverse Probability Weighting" on page 223.

IPW
estimates POMs by inverse probability weighting. This method requires the observed values of the outcome and predicted treatment probabilities. For more information about the IPW estimation method, see the section "Inverse Probability Weighting" on page 220.

REGADJ
estimates POMs by regression adjustment. This method requires predicted counterfactual outcome values. For more information about the REGADJ estimation method, see the section "Regression Adjustment" on page 222.

TMLE
estimates POMs by targeted maximum likelihood estimation. This method requires the observed values of the outcome, predicted treatment probabilities, and predicted counterfactual outcome values. For more information about the TMLE method, see the section "Targeted Maximum Likelihood Estimation" on page 225.

A default estimation method is determined by the input that you provide to the procedure. If you provide predicted treatment probabilities and predicted counterfactual outcomes, by using either an OUTCOMEMODEL statement or the PREDOUT= option in a POM statement, the TMLE method is used by default. If you provide predicted treatment probabilities and no predicted counterfactual outcomes, the IPW method is used by default. If you provide predicted counterfactual outcomes and no predicted treatment probabilities, the REGADJ method is used by default.

POMCOV
displays a covariance matrix of the potential outcome mean estimates. Note that because the CAEFFECT procedure takes a model-agnostic approach to causal effect estimation, the statistical validity of the inference depends on model-specific conditions that you must evaluate and that the procedure cannot verify. For more information about the conditions that are required for valid statistical inference, see the section "Standard Errors and Confidence Limits" on page 227.

POMINFO
displays the "Potential Outcome Mean Information" table. This table summarizes the levels of treatment for which potential outcome means are estimated and the names of variables that contain precomputed treatment probabilities or counterfactual outcomes that are used for estimation.

SCALEDIPWFLAG=number
specifies a multiple of the expected inverse probability weight of an observation that is used to flag observations that have large weights.

By default, SCALEDIPWFLAG=10.

BY Statement

BY *variables* **;**

You can specify a BY statement in PROC CAEFFECT to obtain separate analyses of observations in groups that are defined by the values of the BY variables. If you specify more than one BY statement, only the last one specified is used. For more information, see the discussion of BY-group processing in the "Grouping Data" section of *SAS Programmers Guide: Essentials*.

DIFFERENCE Statement

DIFFERENCE < **REFLEV=(***reference-list* **)** > < **EVTLEV=(***event-list* **)** > **;**

The DIFFERENCE statement specifies causal effects to estimate on the difference scale. Causal effects on the difference scale are computed by taking a potential outcome mean that is defined by an event level and subtracting a potential outcome mean that is defined by a reference level.

You can use the following options to specify the treatment levels for causal effect estimates:

EVTLEV= (*event-list* **)**
specifies event levels for causal effect estimates.

REFLEV= (*reference-list* **)**
specifies reference levels for causal effect estimates.

The levels of treatment that you specify in the *reference-list* and *event-list* must also be used to define a potential outcome in a POM statement.

If you specify both a *reference-list* and a *event-list*, causal effects are computed for each unique pair of treatment levels between the two lists. If you specify a *reference-list* but not an *event-list*, a causal effect is computed for each treatment level in the *reference-list* and all other levels of treatment that you specify in a POM statement that do not appear in the *reference-list*. If you specify an *event-list* but not a *reference-list*, a causal effect is computed for each treatment level in the *event-list* and all levels of treatment that you specify in a POM statement that do not appear in the *event-list*. You can specify more than one DIFFERENCE statement.

If the treatment variable is a character variable, each treatment level that you specify in the *reference-list* and *event-list* should be enclosed in quotes.

DISPLAY Statement

DISPLAY < *table-list* > < / *options* > **;**

The DISPLAY statement enables you to specify a list of display tables to display or exclude. This statement is similar to the ODS SELECT, ODS EXCLUDE, and ODS TRACE statements. However, the DISPLAY statement can improve performance when a large number of tables could be generated (such as in BY-group processing). The procedure processes the DISPLAY statement on a CAS server and thus sends only a subset of ODS tables to the SAS client. Because ODS statements are processed on a SAS client, first all the generated display tables are sent to the client, and then the client creates a subset.

If you use both DISPLAY and ODS statements together, the DISPLAY statement takes precedence over the ODS statements. Note that the ODS EXCLUDE statement processes tables that are sent to the client after they have been filtered by the DISPLAY statement. In some cases, it might appear that the ODS EXCLUDE statement is taking precedence because it can further filter the tables. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

You can specify the *table-list* as a list of table names, paths, partial pathnames, and regular expressions.

The table names that you can specify are listed in the section "ODS Table Names" on page 230. A path is a table name that is prefixed with dot-separated grouping information. For example, a SelectionSummary table that a procedure produces during a selection routine might have the path *Bygroup1.Summary.SelectionSummary*. A partial pathname does not include all groups; for example, *SelectionSummary* and *Summary.SelectionSummary* are partial pathnames for *Bygroup1.Summary.SelectionSummary*.

When you specify a table name or partial pathname, all display tables whose paths end in the specified name are selected for display or exclusion. For example, both *SelectionSummary* and *Summary.SelectionSummary* select *Bygroup1.Summary.SelectionSummary*.

A regular expression is enclosed in forward slashes (/). For example, specifying "/tions/" selects all pathnames that contain the substring "tions"; in particular, the *Bygroup1.Summary.SelectionSummary* table is selected. Specifying "!/tions/" selects all pathnames that do not contain the substring "tions"; in particular, the *Bygroup1.Summary.SelectionSummary* table is not selected.

You can specify the following *options* after a slash (/):

CASESENSITIVE
performs a case-sensitive comparison of table names in the *table-list* to display table names when tables are subsetted for display. To preserve case, you must enclose table names in the *table-list* in quotation marks.

EXCLUDE
displays all display tables except those that you specify in the *table-list*.

EXCLUDEALL
suppresses display of all tables. This option takes precedence over the other options.

TRACE
displays the display table names, labels, and paths.

DISPLAYOUT Statement

DISPLAYOUT *table-spec-list* < / *options* > **;**

The DISPLAYOUT statement enables you to create output tables from your displayed output. This statement is similar to the ODS OUTPUT statement. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

The *table-spec-list* specifies a list of output tables to create. Each entry in the list has either a *key*=*value* format or a *key* format:

key=*value* specifies *key* as the ODS table name, path, or partial pathname, and specifies *value* as the output table name.

key specifies *key* as the ODS table name and also as the output table name.

The ODS table names that you can specify are listed in the section "ODS Table Names" on page 230. You cannot specify the ODS table named OutputCasTables in the *table-spec-list*.

Table names and partial pathnames are discussed under the DISPLAY statement. The DISPLAYOUT statement does not support regular expressions.

You can specify the following *options* after a slash (/):

INCLUDEALL
creates output tables for all display tables. The name of the created output table is the same as the name of the corresponding display table. If you specify this option, the *table-spec-list* specification is ignored.

NOREPLACE
does not replace any existing output table of the same name.

REPEATED
replicates all output tables on all nodes.

FREQ Statement

FREQ *variable* **;**

The *variable* in the FREQ statement identifies a numeric variable in the input data table that contains the frequency of occurrence of each observation. PROC CAEFFECT treats each observation as if it appears f times, where f is the value of the FREQ *variable* for the observation. If f is not an integer, it is truncated to an integer. If f is less than 1 or missing, the observation is not used in the analysis. When you omit the FREQ statement, each observation is assigned a frequency of 1.

OUTCOMEMODEL Statement

OUTCOMEMODEL RESTORE=*libref.data-table* **PREDNAME=***prediction-name* **;**

The OUTCOMEMODEL statement enables you to use a previously fit model for the outcome variable to compute counterfactual outcomes. PROC CAEFFECT uses the model to obtain predicted counterfactual outcomes for each treatment level that you specify in a POM statement.

When you use the OUTCOMEMODEL statement, you must specify the following options:

RESTORE=*libref.data-table*
specifies the saved model in the form of an analytic store.

PREDNAME=*prediction-name*
identifies the variable that contains the outcome of interest when data are scored using the analytic store.

The OUTCOMEMODEL statement is ignored if the method that you use to estimate potential outcome means does not require modeling of the outcome variable. If you specify this statement, any predicted counterfactual outcomes that you specify by using the PREDOUT= option in a POM statement are ignored.

OUTCOMEVAR Statement

OUTCOMEVAR *outcome-variable* < **(EVENT=**'*event-level*'**)** > < */outcome-options* > **;**

OUTCOMEVAR *events* / *trials* < */outcome-options* > **;**

The OUTCOMEVAR statement specifies information about the outcome variable. Information about the outcome variable is required for all estimation methods except the regression adjustment method. You can specify the outcome as a single variable or as a ratio of two variables that are denoted as *events*/*trials*. The first form applies to response variables that are continuous or categorical; the second form applies only to summarized binomial response data.

When you have binomial data, the *events* variable contains the number of positive responses (or events) and the *trials* variable contains the number of trials. The values of both *events* and (*trials* – *events*) must be nonnegative, and the value of *trials* must be positive. For binomial response data, the potential outcomes correspond to the event probability.

When you have categorical data, you must specify the following option:

EVENT='*event-level***'**
specifies the outcome *event-level* of interest.

For categorical response data, the potential outcomes correspond to the probability of the event level occurring.

You can specify the following *outcome-options* after a slash (/):

COUNTMISSING
treats missing values (".", ".A", . . . , ".Z" for numeric variables and blanks for character variables) as valid values of a categorical outcome variable.

LEVELIZERAW
bases the levelization of a categorical outcome variable on raw values.

MAXLEV=*number*
specifies the maximum number of levels for a categorical outcome variable. The *number* 0 means an unlimited number of levels.

TYPE=BINOMIAL | CATEGORICAL | CONTINUOUS
specifies the type of the outcome variable.

You can specify one of the following types:

BINOMIAL
specifies a binomial outcome. For a binomial outcome, the event probability is modeled, and predicted outcome values must be between 0 and 1, inclusive. If the specified estimation method requires the observed outcome values, you must use event/trial syntax. Event variable values must be nonnegative, trial variable values must be positive, and the number of events must be less than or equal to the number of trials.

CATEGORICAL
specifies a categorical outcome. For a categorical outcome, the probability of a designated event level is modeled, and the observed outcome and predicted outcome values must be between 0 and 1, inclusive. If the specified estimation method requires the observed outcome values, you must specify the event level by using the EVENT= option.

CONTINUOUS
specifies a continuous outcome. For a continuous outcome, the observed outcome and predicted outcome values can be any real values.

POM Statement

POM TREATLEV='*treatment-level*' < **PREDOUT=***predicted-outcome-value* > < **TREATPROB=***predicted-treatment-probability* > **;**

The POM statement specifies a potential outcome mean to estimate. You must specify the following option in a POM statement:

TREATLEV='*treatment-level*'

TRTLEV='*treatment-level*'

specifies the level of the treatment variable that defines the potential outcome. If the treatment variable is a character variable, specify the *treatment-level* in quotes.

For more information about the definition of potential outcomes and the counterfactual framework to causal effect estimation, see the section "Potential Outcomes and Effect Definitions" on page 216.

Depending on the estimation method that you use, you can use the following options to specify variables in the input data table that contain precomputed model predictions:

PREDOUT=*predicted-outcome-value*

specifies the input variable that contains the predicted counterfactual outcome values for the specified treatment level. Predicted counterfactual outcomes are not used for the IPW estimation method. If you use the OUTCOMEMODEL statement to specify a previously fit model to use in computing counterfactual outcomes, the PREDOUT= option is ignored.

TREATPROB=*predicted-treatment-probability*

TRTPROB=*predicted-treatment-probability*

specifies the input variable that contains the predicted probability of having the specified level of treatment. Predicted treatment probabilities are not used for the REGADJ estimation method.

TREATVAR Statement

TREATVAR *treatment-variable* < */treatment-options* > **;**

The TREATVAR statement specifies information about the treatment variable. For all the estimation methods, you must specify the name of the *treatment-variable*.

You can specify the following *treatment-options* after a slash (/):

CONDEVENT='*treatment-level*'

EVENT='*treatment-level*'

specifies the estimation of conditional potential outcome means and the observed *treatment-level* to condition on. This *treatment-level* must appear in a potential outcome specification. Conditional potential outcome mean estimation is supported only for the IPW and REGADJ estimation methods. For more information about the definition of causal effects, see the section "Potential Outcomes and Effect Definitions" on page 216.

COUNTMISSING
treats missing values (“.”, “.A”, . . . , “.Z” for numeric variables and blanks for character variables) as valid values of the treatment variable.

LEVELIZERAW
bases the levelization of the treatment variable on raw values.

MAXLEV=*number*
specifies the maximum number of levels for the treatment variable. The *number* 0 means an unlimited number of levels.

WEIGHT Statement

WEIGHT *variable* **;**

The *variable* that you specify in the WEIGHT statement is used as a weight to perform a weighted analysis of the data. Observations that have nonpositive weights or missing weights are excluded from the analysis. The values of the WEIGHT variable can be nonintegral and are not truncated. When you omit this statement, all observations that are used in the analysis are assigned a weight of 1.

Details: CAEFFECT Procedure

Missing Values

Any observation that has missing values for a predicted counterfactual outcome, predicted treatment probability, treatment variable, frequency variable, or weight variable is excluded from the analysis; however, missing values are valid for the treatment variable if you specify the COUNTMISSING option in the TREATVAR statement. If the estimation method requires the observed outcome values, any observation that has a missing value for the outcome is excluded from the analysis; however, missing values are valid for the categorical outcome variable if you specify the COUNTMISSING option in the OUTCOMEVAR statement.

Potential Outcomes and Effect Definitions

This section describes the Neyman-Rubin potential outcomes framework (Rubin 1980, 1990), which the CAEFFECT procedure uses to define causal effects. This framework provides an approach that you can use to clearly articulate causal assumptions and to define causal effects of a treatment T on an outcome Y. Note that although the effects are referred to as treatment effects, this is done only for ease of exposition, because the variable T could also represent an intervention or exposure. For example, T could indicate enrollment in a job training program, and Y could indicate subsequent employment.

Potential outcomes, which are also referred to as counterfactual outcomes, are used to describe an idealized source of data where you can observe a subject’s response to all possible treatment assignments. The potential outcome $Y(t)$ is defined to be the value of the outcome variable that would occur if the subject were to receive

treatment level t (possibly contrary to the observed assignment). For example, if you consider a categorical treatment T with p levels, there are p potential outcomes $Y(t_i)$, for $i = 1, \ldots, p$. With the potential outcomes, it is natural to define a unit-level treatment effect between treatment levels t_1 and t_0 as the difference between $Y(t_1)$ and $Y(t_0)$.

Although the individual-level treatment effects that are defined using potential outcomes are intuitive, a number of assumptions must be satisfied for them to be well-defined quantities. The stable unit treatment value assumption (SUTVA), which is described by Rubin (1980), is one assumption that you typically make when using potential outcomes to define causal effects. This assumption ensures that the potential outcomes that you hypothesize are well-defined intrinsic values associated with a subject. SUTVA has two components:

- a no-interference condition, which assumes that a subject's potential outcomes are not affected by the treatment assignment of the other subjects
- an assumption of no hidden levels of treatment, meaning that the potential outcomes capture all relevant variability in the treatment

In a more refined statement, VanderWeele (2009) proposed a less restrictive version of the second component of SUTVA. Essentially, VanderWeele's version requires that the potential outcomes remain unchanged within a certain degree of treatment variation for each treatment level. This version is also known as the treatment variation irrelevance assumption. (See also Cole and Frangakis 2009.)

To connect the potential outcomes to observed data, you make a consistency assumption that states that the observed outcome Y is equal to the potential outcome that is defined by the observed level of the treatment T. To express this assumption mathematically, let $\mathbb{1}(T = t)$ denote the indicator function for the event in which $T = t$. You can then write the consistency assumption as

$$Y = \sum_{i=1}^{p} \mathbb{1}(T = t_i) Y(t_i)$$

The consistency assumption makes clear that you can observe at most one potential outcome for each subject. Therefore, you are typically unable to reliably estimate all potential outcomes and the unit-level treatment effects that they define for a subject. Instead, you typically consider the estimation of the potential outcome means, $E\left[Y(t)\right]$. You can also consider average treatment effects for the entire population. For example, for the designated event level t_1 and reference level t_0, the average treatment effect (ATE) on the difference scale is given by

$$E\left[Y(t_1)\right] - E\left[Y(t_0)\right]$$

The ATE is interpreted as the mean difference between the outcomes when everyone in the population receives treatment level t_1 and when everyone receives treatment level t_0.

In addition to estimating potential outcome means and average treatment effects for the entire population, you can consider estimating potential outcome means and treatment effects conditional on subjects receiving a particular level of treatment. That is, for a designated level of treatment t_{trt}, you can estimate $E\left[Y(t) \mid T = t_{\text{trt}}\right]$. For a binary treatment variable T, these conditional potential outcome mean estimates define the average treatment effect for the treated (ATT). Assuming that $T = t_1$ is the designated treatment level and $T = t_0$ is the reference level, the ATT is given by

$$E\left[Y(1) \mid T = t_1\right] - E\left[Y(0) \mid T = t_1\right]$$

Here, the ATT is interpreted as the average effect of treatment on those subjects who receive it.

Identification of Causal Effects

As described by Rubin (1990), it is implied in this potential outcomes approach to defining causal effects that if the SUTVA assumption holds, the potential outcomes are not affected by the treatment assignment mechanism that is used to generate the observed data. That is, whether you conduct a fully randomized experiment or an observational study, if you can assume SUTVA and you have a well-defined causal problem, you can define the same potential outcomes and average treatment effects.

Data that are generated from a fully randomized experiment and from an observational study differ in what you can assume about the relationship between the potential outcomes and the observed treatment assignment. This difference determines whether it is feasible to obtain unbiased estimates of the effects of interest and what estimation methods you can use to do so.

When the data are generated from a fully randomized experiment, external randomization enables you to safely assume that the potential outcomes and treatment assignment are independent. The observed outcome Y still depends on the treatment assignment through the consistency assumption. Assuming independent potential outcomes and treatment assignment, you can obtain an unbiased estimate of $E[Y(t)]$ by taking the average of the observed outcome Y for those subjects in which treatment $T = t$.

When the data are generated from a nonrandomized or observational study, there can be a number of pretreatment characteristics that are associated with both the potential outcomes and the treatment assignment. These pretreatment variables are often referred to as confounding variables or confounding covariates. To enable the unbiased estimation of potential outcome means in the presence of confounding, you must assume that there is a set of measured confounders $\boldsymbol{X}$ such that the potential outcomes and treatment are conditionally independent given $\boldsymbol{X}$. This condition is often articulated as an ignorability or conditional exchangeability assumption. A set of confounders $\boldsymbol{X}$ that satisfies this conditional independence property is often referred to as a valid adjustment set, because a statistical adjustment of these confounders can produce an unbiased estimate of the potential outcome means.

The assumption that a set of confounders $\boldsymbol{X}$ is a valid adjustment set typically is not empirically verifiable, because the assumption involves the unobserved potential outcomes. Moreover, the potential outcome definitions do not naturally lend themselves to representing assumptions of why a particular set of confounders $\boldsymbol{X}$ might provide a valid adjustment set. Graphical causal models (Spirtes, Glymour, and Scheines 2001; Pearl 2009a, b; Elwert 2013) are one approach that you can use to depict assumptions about the data-generating process and the association between variables. As described by Shpitser, VanderWeele, and Robins (2010), graphical causal models and potential outcomes can be related through an equivalence between a set of confounders $\boldsymbol{X}$, which satisfy the conditional exchangeability assumption, and a certain graphical identification criterion. You can therefore use graphical causal models as a way to represent the causal assumptions about a data-generating process that would justify the use of a particular adjustment set. For more information about graphical causal models and how you can use them to assess the identifiability of treatment effects, see Chapter 37, "The CAUSALGRAPH Procedure" (*SAS/STAT User's Guide*).

An additional assumption that you typically make when estimating a causal effect is the positivity assumption: that is, each subject should have a nonzero probability of receiving any treatment level. This assumption can be expressed as

$$0 < \Pr(T = t \mid \boldsymbol{X} = \mathbf{x}) < 1 \quad \text{for all } t = 1, \ldots, p$$

The positivity assumption ensures that the causal effects are practically well defined for the population of interest.

Adjustment for Confounding

You can adjust for a set of confounding variables $\boldsymbol{X}$ by modeling the treatment variable T, the outcome variable Y, or both. PROC CAEFFECT enables you to use a model for the treatment variable T to estimate potential outcome means through inverse probability of treatment weighting. When you use a model for the outcome variable Y, the procedure can estimate potential outcome means by using regression adjustment (also called a response surface approach). Finally, you can combine predicted treatment probabilities and an outcome model in so-called doubly robust approaches. An estimation method is considered doubly robust if it provides unbiased estimates of the potential outcome means even if one of the models is misspecified. PROC CAEFFECT can perform doubly robust estimation by using either augmented inverse probability weighting or targeted maximum likelihood estimation. For more information about the estimation methods that this procedure implements, see the section "Estimation Methods" on page 219.

For more information about defining and estimating causal effects by using a potential outcomes framework, see Rubin (1980), Rubin (1990), Neyman, Dabrowska, and Speed (1990), Imbens and Rubin (2015), Hernán and Robins (2020), and references therein.

Estimation Methods

PROC CAEFFECT implements several methods of estimating potential outcome means and their differences. The causal interpretation of these estimates depends on the plausibility of the causal assumptions that are described in the section "Potential Outcomes and Effect Definitions" on page 216.

These estimation methods differ primarily in how they make statistical adjustments for confounding variables. In general, you can adjust for confounding variables by modeling the treatment variable, the outcome variable, or both. PROC CAEFFECT is a model-agnostic tool for effect estimation. That is, the procedure itself does not support modeling of the treatment and outcome variables. Instead, as input to the procedure, you provide the quantities necessary for your chosen estimation method. Thus, you can adjust for the effects of confounding variables by using predictions from a previously fit model for the treatment variable, a previously fit model for the outcome variable, or models for both the treatment and outcome variables. The required inputs for each method that PROC CAEFFECT supports are summarized in Table 6.2.

Table 6.2 Required Inputs of Different Estimation Methods

Method	Treatment Model	Outcome Model	Observed Outcome
Inverse probability weighting	X		X
Regression adjustment		X	
Augmented inverse probability weighting	X	X	X
Targeted maximum likelihood estimation	X	X	X

The inverse probability weighting (IPW) method requires a model for the treatment variable. To use this method, you must specify variables in the input data table that contain the precomputed treatment assignment probabilities. For an example of how you can perform IPW estimation, see "Example 6.1: Estimation by Inverse Probability Weighting" on page 231. For more information about estimation by inverse probability weighting, see the section "Inverse Probability Weighting" on page 220.

The regression adjustment method requires a model for the outcome variable. To use this method, you have two choices for the input that you must provide: you can specify variables in the input data table that contain the predicted potential outcomes for each specified level of treatment, or you can provide a previously fit model in the form of an analytic store by using the OUTCOMEMODEL statement. In the second case, the procedure uses the analytic store to compute the relevant predicted outcome values. When you provide an analytic store as input, the model must include the specified treatment variables as a predictor, and all the required predictor variables must appear in the input data table. For an example that shows how you can perform estimation by regression adjustment by providing the procedure with either an analytic store or precomputed predicted values, see "Example 6.2: Estimation by Regression Adjustment" on page 235. For more information about estimation by regression adjustment, see the section "Regression Adjustment" on page 222.

The remaining two methods, augmented inverse probability weighting (AIPW) and targeted maximum likelihood estimation (TMLE), are so-called doubly robust estimation methods. An estimation method is considered doubly robust if it provides unbiased estimates whenever at least one of the models is correctly specified. For these methods, you must specify a model for both the treatment variable and the outcome variable. For an example of how you can use PROC CAEFFECT to perform doubly robust estimation, see "Example 6.3: Estimation by Doubly Robust Methods" on page 239. For more information about estimation by augmented inverse probability weighting, see the section "Augmented Inverse Probability Weighting" on page 223. For more information about targeted maximum likelihood estimation, see the section "Targeted Maximum Likelihood Estimation" on page 225.

Supported Variable Types

PROC CAEFFECT supports the estimation of treatment effects for categorical treatment variables and for continuous, categorical, or binomial outcomes. In the case of categorical outcomes, you must specify a designated event level by using the EVENT= option in the OUTCOMEVAR statement, and the outcome model (specified as either predicted potential outcomes or an analytic store) must represent the probability of that event level. Similarly, for binomial data, the outcome model must represent the event probability. If you are modeling binomial data and the estimation method that you use requires observed outcome values, you must use the event/trial syntax in the OUTCOMEVAR statement. This syntax requires that you specify two variables that contain the counts from a binomial experiment. The event variable contains the number of positive responses, or events, and the trial variable contains the number of trials that are conducted.

Inverse Probability Weighting

When you estimate potential outcome means by inverse probability weighting (IPW), the observed outcomes for each subject are assigned weights inversely proportional to the probability of having received the observed level of treatment. Incorporating the IPW weights creates a pseudopopulation that corrects for the bias that the confounding variables introduce. You can obtain unbiased estimates of the potential outcome means from the weighted pseudopopulation. For more information about IPW estimation, see Lunceford and Davidian (2004), Hernán and Robins (2020), and references therein.

When estimating ℓ potential outcome means by IPW, you must provide as input ℓ variables that contain the predicted treatment probabilities of a subject being assigned to the specified treatment levels. You specify the variables that contain these precomputed probabilities by using the TREATPROB= option in each potential outcome specification. For an example of using PROC CAEFFECT to implement the IPW estimation method, see "Example 6.1: Estimation by Inverse Probability Weighting" on page 231.

The following list defines components of the estimating equations that you solve to obtain the IPW estimates of the potential outcome means:

- T denotes the treatment variable.
- y_i denotes the observed outcome for subject i.
- $\mathbb{1}(T_i = t)$ denotes the indicator function for subject i being assigned to treatment level t.
- e_{ti} denotes the conditional probability of subject i being assigned to treatment level t; that is, $\Pr(T_i = t \mid \boldsymbol{X} = \boldsymbol{x}_i)$.

The IPW estimates of the potential outcome means solve the equations

$$\mathbf{S}_{\text{ipw}}(\boldsymbol{\mu}) = \sum_{i=1}^{n} \mathbf{S}_{\text{ipw},i}(\boldsymbol{\mu}) = 0$$

where

$$\mathbf{S}_{\text{ipw},i}(\boldsymbol{\mu}) = \begin{bmatrix} \mathbb{1}(T_i = t_1)\left(\frac{y_i - \mu_1}{e_{1i}}\right) \\ \vdots \\ \mathbb{1}(T_i = t_\ell)\left(\frac{y_i - \mu_\ell}{e_{\ell i}}\right) \end{bmatrix}$$

The solution to these equations has a closed-form expression given by

$$\hat{\mu}_{t_j} = \left(\sum_{i=1}^{n} \frac{\mathbb{1}(T_i = t_j)}{e_{t_j i}}\right)\left(\sum_{i=1}^{n} \frac{\mathbb{1}(T_i = t_j) y_i}{e_{t_j i}}\right) \quad \text{for } j = 1, \ldots, \ell$$

You can also use the IPW estimation method to estimate potential outcome means conditional on subjects receiving a particular treatment assignment. You specify the level of treatment to condition on by using the CONDEVENT= option in the TREATVAR statement. The level of treatment that you specify for the conditioning event must also be used in a potential outcome specification in a POM statement.

Let t_{trt} denote the level of treatment that you specify to condition on for computing conditional potential outcome mean estimates. The IPW estimates of the potential outcome means conditional on $T = t_{\text{trt}}$ solve a similar set of estimating equations, where the weights that are used to estimate the unconditional means (one over the observed treatment assignment probability) are multiplied by $e_{t_{\text{trt}}}$, the probability of being assigned to the treatment level t_{trt}. The score equation for each subject is then given by

$$\mathbf{S}_{\text{ipw},i}(\boldsymbol{\mu}) = \begin{bmatrix} \mathbb{1}(T_i = t_1) e_{t_{\text{trt}} i}\left(\frac{y_i - \mu_1}{e_{1i}}\right) \\ \vdots \\ \mathbb{1}(T_i = t_\ell) e_{t_{\text{trt}} i}\left(\frac{y_i - \mu_\ell}{e_{\ell i}}\right) \end{bmatrix}$$

The solution to the estimating equations for the conditional potential outcome means is then given by

$$\hat{\mu}_{t_j | T = t_{\text{trt}}} = \left(\sum_{i=1}^{n} \frac{\mathbb{1}(T_i = t_j) e_{t_{\text{trt}} i}}{e_{t_j i}}\right)\left(\sum_{i=1}^{n} \frac{\mathbb{1}(T_i = t_j) e_{t_{\text{trt}} i} y_i}{e_{t_j i}}\right) \quad \text{for } j = 1, \ldots, \ell$$

If you specify a weight variable by using the WEIGHT statement, the score equation for each subject is multiplied by the value of this variable, and the solutions to the estimating equations are updated accordingly. When you use the IPW estimation method, an observation from the input data table is not used if any of the following conditions occur:

- A predicted treatment assignment probability is missing.
- The value of the outcome variable is missing.
- The observed treatment level is not used in a potential outcome specification.
- The value of a frequency or weight variable, if specified, is missing or nonpositive.

Regression Adjustment

When you estimate potential outcome means by regression adjustment, you use predicted potential outcome values that are obtained from a model of the outcome variable. The predicted potential outcome values for each subject are obtained by replacing the observed treatment assignment with the treatment assignments that define the potential outcomes of interest and then by evaluating the outcome model for each treatment assignment in turn. The averages of the predicted values are used to estimate the average of the potential outcomes. For more information about estimation by regression adjustment, see Hernán and Robins (2020) and references therein.

To estimate ℓ potential outcome means by regression adjustment, you must provide as input either ℓ variables that contain the predicted potential outcome values or a previously fit model for the outcome in the form of an analytic store. If the input that you provide consists of predicted potential outcome values, you must use the PREDOUT= option for each potential outcome specification to specify the variable that represents the predicted potential outcomes. If the input that you provide consists of a previously fit model, you use the OUTCOMEMODEL statement, specify the name of the analytic store by using the RESTORE= option, and specify the name of the variable that contains the model predictions by using the PREDNAME= option. When you specify a previously fit model, PROC CAEFFECT uses the model to compute the potential outcome predictions. For an example of using this procedure to implement the regression adjustment estimation method, see "Example 6.2: Estimation by Regression Adjustment" on page 235.

To describe the estimating equations, you solve to obtain the regression adjustment estimates of the potential outcome means. Let $\hat{y}_{ti}$ denote the predicted counterfactual outcome value that you obtain by replacing the observed treatment assignment with $T = t$. The regression adjustment estimates of the potential outcome means solve the equations

$$\mathbf{S}_{\text{regadj}}(\boldsymbol{\mu}) = \sum_{i=1}^{n} \mathbf{S}_{\text{regadj},i}(\boldsymbol{\mu}) = 0$$

where

$$\mathbf{S}_{\text{regadj},i}(\boldsymbol{\mu}) = \begin{bmatrix} \hat{y}_{1i} - \mu_1 \\ \vdots \\ \hat{y}_{\ell i} - \mu_\ell \end{bmatrix}$$

The solution to these equations has a closed-form expression given by

$$\hat{\mu}_{t_j} = \frac{1}{n} \sum_{i=1}^{n} \hat{y}_{t_j i} \quad \text{for } j = 1, \ldots, \ell$$

You can also use the regression adjustment estimation method to estimate potential outcome means conditional on subjects receiving a particular treatment assignment. You specify the level of treatment to condition on by using the CONDEVENT= option in the TREATVAR statement. The level of treatment that you specify for the conditioning event must also be used in a potential outcome specification in a POM statement.

The regression adjustment estimates of the potential outcome means conditional on $T = t_{\text{trt}}$ solve a similar set of estimating equations, where predicted counterfactual values are averaged only for subjects that have the treatment assignment that you are conditioning on. The following list defines components of the estimating equations that you solve to obtain the regression adjustment estimates of the potential outcome means conditional on a treatment level:

- t_{trt} denotes the level of treatment that you specify to define the conditioning event $T = t_{\text{trt}}$.
- $\mathbb{1}(T_i = t_{\text{trt}})$ denotes the indicator function for subject i being assigned to treatment level t_{trt}.
- $n_{t_{\text{trt}}}$ denotes the number of subjects that have the observed treatment assignment $T_i =_{\text{trt}}$.

The score equation for each subject is then given by

$$\mathbf{S}_{\text{regadj},i}(\boldsymbol{\mu}) = \begin{bmatrix} \mathbb{1}(T_i = t_{\text{trt}})(\hat{y}_{1i} - \mu_1) \\ \vdots \\ \mathbb{1}(T_i = t_{\text{trt}})(\hat{y}_{\ell i} - \mu_\ell) \end{bmatrix}$$

The solution to these equations has a closed-form expression given by

$$\hat{\mu}_{t_j|T=t_{\text{trt}}} = \frac{1}{n_{t_{\text{trt}}}} \sum_{i=1}^{n} \mathbb{1}(T_i = t_{\text{trt}}) \hat{y}_{t_j i} \quad \text{for } j = 1, \ldots, \ell$$

If you specify a weight variable by using the WEIGHT statement, the score equation for each subject is multiplied by the value of this variable, and the solutions to the estimating equations are updated accordingly. When you use the regression adjustment estimation method, an observation from the input data table is not used if either of the following conditions occurs:

- A predicted potential outcome value is missing.
- The value of a frequency or weight variable, if specified, is missing or nonpositive.

In addition to these conditions, if you are estimating potential outcome means conditional on a particular treatment assignment, only observations that have the specified treatment assignment are used.

Augmented Inverse Probability Weighting

When you estimate potential outcome means by augmented inverse probability weighting (AIPW), predictions from a model for the outcome variable and predictions from a model for the treatment variable are combined to form a doubly robust estimation method. An estimation method is said to be doubly robust if it provides unbiased estimates whenever at least one of the models that you use is correctly specified. The model for the outcome variable is used in the AIPW estimation method to predict potential outcome values for each subject by replacing the observed treatment assignment with the treatment assignments that define the potential outcomes of interest. The model for the treatment variable is used to compute weights that are inversely

proportional to the probability that a subject has the observed treatment assignment. For more information about the AIPW estimation method, see Lunceford and Davidian (2004), Hernán and Robins (2020), and references therein.

To estimate ℓ potential outcome means by AIPW, you must provide as input ℓ variables that contain the predicted probability of a subject being assigned to the specified treatment levels. You specify the variables that contain the precomputed treatment assignment probabilities by using the TREATPROB= option in each potential outcome specification. Also, you must specify either ℓ variables that contain the predicted potential outcome values or a previously fit model for the outcome in the form of an analytic store. If the input that you provide consists of predicted potential outcome values, you must use the PREDOUT= option for each potential outcome specification to specify the variable that represents the predicted potential outcomes. If the input that you provide consists of a previously fit model, you use the OUTCOMEMODEL statement, specify the name of the analytic store by using the RESTORE= option, and specify the name of the variable that contains the model predictions by using the PREDNAME= option. When you specify a previously fit model, PROC CAEFFECT uses the model to compute the potential outcome predictions. For an example of using this procedure to implement the AIPW estimation method, see "Example 6.3: Estimation by Doubly Robust Methods" on page 239.

The following list defines components of the estimating equations that you solve to obtain the AIPW estimates of the potential outcome means:

- T denotes the treatment variable.
- y_i denotes the observed outcome for subject i.
- $\hat{y}_{ti}$ denotes the predicted potential outcome value that you obtain by replacing the observed treatment assignment with $T = t$.
- $\mathbb{1}(T_i = t)$ denotes the indicator function for subject i being assigned to treatment level t.
- e_{ti} denotes the conditional probability of subject i being assigned to treatment level t; that is, $\Pr(T_i = t \mid \boldsymbol{X} = \boldsymbol{x}_i)$.

The AIPW estimates of the potential outcome means solve the equations

$$\mathbf{S}_{\text{aipw}}(\boldsymbol{\mu}) = \sum_{i=1}^{n} \mathbf{S}_{\text{aipw},i}(\boldsymbol{\mu}) = 0$$

where

$$\mathbf{S}_{\text{aipw},i}(\boldsymbol{\mu}) = \begin{bmatrix} \frac{\mathbb{1}(T_i=t_1)y_i}{e_{1i}} - \hat{y}_{1i}\left(\frac{\mathbb{1}(T_i=t_1)-e_{1i}}{e_{1i}}\right) - \mu_1 \\ \vdots \\ \frac{\mathbb{1}(T_i=t_\ell)y_i}{e_{\ell i}} - \hat{y}_{\ell i}\left(\frac{\mathbb{1}(T_i=t_\ell)-e_{\ell i}}{e_{\ell i}}\right) - \mu_\ell \end{bmatrix}$$

The solution to these equations has a closed-form expression given by

$$\hat{\mu}_{t_j} = \frac{1}{n}\sum_{i=1}^{n} \frac{\mathbb{1}(T_i = t_j)y_i}{e_{t_j i}} - \hat{y}_{t_j i}\left(\frac{\mathbb{1}(T_i = t_j) - e_{t_j i}}{e_{t_j i}}\right) \quad \text{for } j = 1, \ldots, \ell$$

If you specify a weight variable by using the WEIGHT statement, the score equation for each subject is multiplied by the value of this variable, and the solutions to the estimating equations are updated accordingly. When you use the AIPW estimation method, an observation from the input data table is not used if any of the following conditions occur:

- A predicted treatment assignment probability is missing.
- The value of the outcome variable is missing.
- A predicted counterfactual value is missing.
- The value of a frequency or weight variable, if specified, is missing or nonpositive.

Targeted Maximum Likelihood Estimation

When you estimate potential outcome means by targeted maximum likelihood estimation (TMLE), predictions from a model for the outcome variable and predictions from a model for the treatment variable are combined to form a doubly robust estimation method. An estimation method is said to be doubly robust if it provides unbiased estimates whenever one of the models that you use is correctly specified. The model for the outcome variable is used in the TMLE method to predict potential outcome values for each subject by replacing the observed treatment assignment with the treatment assignments that define the potential outcomes of interest. The model for the treatment variable is used to compute covariates that a targeting step uses to reduce finite sample bias and model misspecification bias in the outcome model. For more information about the TMLE method, see Gruber and Van der Laan (2009), Van der Laan (2010a), Van der Laan (2010b), and Van der Laan and Rose (2011).

To estimate ℓ potential outcome means by TMLE, you must provide as input ℓ variables that contain the predicted probability of a subject being assigned to the specified treatment levels. You specify the variables that contain the precomputed treatment assignment probabilities by using the TREATPROB= option in each potential outcome specification. Also, you must specify either ℓ variables that contain the predicted potential outcome values or a previously fit model for the outcome in the form of an analytic store. If the input that you provide consists of predicted potential outcome values, you must use the PREDOUT= option for each potential outcome specification to specify the variable that represents the predicted potential outcomes. If the input that you provide consists of a previously fit model, you use the OUTCOMEMODEL statement, specify the name of the analytic store by using the RESTORE= option, and specify the name of the variable that contains the model predictions by using the PREDNAME= option. When you specify a previously fit model, PROC CAEFFECT uses the model to compute the potential outcome predictions. For an example of using this procedure to implement the TMLE method, see "Example 6.3: Estimation by Doubly Robust Methods" on page 239.

The following list defines components of the equations that you use to obtain the TMLE estimates of the potential outcome means:

- T denotes the treatment variable.
- y_i denotes the observed outcome for subject i.
- $\hat{y}_{ti}$ denotes the predicted potential outcome value that you obtain by replacing the observed treatment assignment with $T = t$.

- $\tilde{y}_i$ denotes the predicted potential outcome value that corresponds to the observed treatment value.
- y^*_{ti} denotes the predicted potential outcome value that you obtain by applying the targeting step to the initial predicted potential outcome value that corresponds to treatment level $T = t$.
- $\mathbb{1}(T_i = t)$ denotes the indicator function for subject i being assigned to treatment level t.
- e_{ti} denotes the conditional probability of subject i being assigned to treatment level t; that is, $\Pr(T_i = t \mid \boldsymbol{X} = \boldsymbol{x}_i)$.

The TMLE algorithm starts with the predicted potential outcomes that are obtained from the outcome model that you provide as input. The algorithm then uses a targeting step to refine those predictions. The targeting step is performed by regressing the observed outcome variable against a carefully chosen set of covariates; the initial predicted potential outcomes are specified as a regression offset.

There is one covariate, H_{t_j}, for each potential outcome mean that you want to estimate ($j = 1, \ldots, \ell$). When you estimate the potential outcome means in the population, the values of the clever covariates are computed for each observation by the equation

$$\hat{h}_{t_j i} = \frac{\mathbb{1}(T_i = t_j)}{e_{t_j i}} \quad \text{for } j = 1, \ldots, \ell$$

Note that for each individual, only one of the ℓ covariates is nonzero. The covariates are used in the regression equation

$$\operatorname{logit}(y) = \operatorname{logit}(\tilde{y}) + \sum_{j=1}^{\ell} \epsilon_j \hat{h}_{t_j}$$

In the targeting step, the TMLE algorithm computes estimates $\hat{\epsilon}_j$ of the regression parameters ϵ_j and then uses these estimates to update the predicted potential outcomes. These updated predictions are computed by the formula

$$\operatorname{logit}(y^*_{t_j i}) = \operatorname{logit}(\hat{y}_{t_j i}) + \frac{\hat{\epsilon}_j}{e_{t_j i}}$$

When the outcome variable is binary, the estimated regression parameters $\hat{\epsilon}_j$ are computed by logistic regression. When the outcome variable is continuous or binomial, the regression parameters are computed by a generalized linear model that has a binomial distribution and a logit link. In the latter case, you must rescale the observed outcomes and the predicted potential outcomes so that all values are contained in the open interval (0,1). In both cases, the regression is computed with no intercept and with $\operatorname{logit}(\tilde{y})$ as the offset.

After the targeting step, the potential outcome means are given by

$$\hat{\mu}_{t_j} = \frac{1}{n} \sum_{i=1}^{n} y^*_{t_j i} \quad \text{for } j = 1, \ldots, \ell$$

If you specify a weight variable by using the WEIGHT statement, this weight variable is applied to the regression model that defines the targeting step, and final computation of the potential outcome means is updated accordingly. When you use the TMLE estimation method, an observation from the input data table is not used if any of the following conditions occur:

- A predicted treatment assignment probability is missing.
- The value of the outcome variable is missing.
- A predicted potential outcome value is missing.
- The value of a frequency or weight variable, if specified, is missing or nonpositive.

Standard Errors and Confidence Limits

You can specify the INFERENCE option in the PROC CAEFFECT statement to request standard errors and confidence limits for the potential outcome mean estimates.

Note that the statistical validity of the standard error estimates and confidence limits that the CAEFFECT procedure computes depends on how the treatment and outcome models are estimated. Because the procedure takes a model-agnostic approach to estimating potential outcome means, it cannot assess these model-dependent conditions, and you must determine whether the modeling techniques that you are using will result in valid inference.

The following sections discuss how the CAEFFECT procedure performs variance estimation for the different estimation methods and what conditions are required for these estimates to produce statistically valid inference. These conditions are specific to the statistical validity of the variance and interval estimation only. The statistical validity of the point estimation of potential outcome means and their differences is not affected by these conditions.

Estimating Equation Methods

For the inverse probability weighting (IPW), regression adjustment, and augmented inverse probability weighting (AIPW) estimation methods, standard errors and confidence limits for the potential outcome mean estimates are computed by using the sandwich covariance formula that is based on asymptotic theory. This approach considers the estimating equations as an M-estimation problem (Stefanski and Boos 2002) that is given by

$$S(\boldsymbol{\mu}) = \sum_{i=1}^{n} S_i(\boldsymbol{\mu}) = 0$$

The robust, or sandwich, covariance matrix $\hat{\boldsymbol{V}}(\hat{\boldsymbol{\mu}})$ for the estimates $\hat{\boldsymbol{\mu}}$ is given by

$$\hat{\boldsymbol{V}}(\hat{\boldsymbol{\mu}}) = n^{-1}\boldsymbol{A}_n(\hat{\boldsymbol{\mu}})^{-1}\boldsymbol{B}_n(\hat{\boldsymbol{\mu}})\boldsymbol{A}_n(\hat{\boldsymbol{\mu}})^{-T}$$

where n is the sample size, $\boldsymbol{A}^{-T}$ denotes $\left(\boldsymbol{A}^{-1}\right)^{T}$, and

$$\boldsymbol{A}_n(\hat{\boldsymbol{\mu}}) = n^{-1}\sum_{i=1}^{n} -\frac{\partial S_i(\hat{\boldsymbol{\mu}})}{\partial \boldsymbol{\mu}}$$

$$\boldsymbol{B}_n(\hat{\boldsymbol{\mu}}) = n^{-1}\sum_{i=1}^{n} S_i(\hat{\boldsymbol{\mu}})S_i(\hat{\boldsymbol{\mu}})^{T}$$

Note that the validity of this estimate for the covariance matrix and the rate of convergence that it assumes depend on a number of factors that relate to the modeling of the treatment and outcome variables. In particular, because PROC CAEFFECT takes a model-agnostic approach to estimating potential outcome means, these expressions treat all predicted treatment probabilities and predicted counterfactual outcome values as fixed quantities. If you use the same data (that is, you do not use data splitting) to estimate the potential outcome means and to estimate the underlying treatment and outcome models, you can introduce dependencies that are unaccounted for in the covariance matrix estimate. Moreover, if the modeling technique that you use for the treatment or outcome variable converges at a rate slower than the square root of n, variability in the predicted quantities might lead to a rate of convergence for the potential outcome mean estimates that is slower than the rate that this approach to inference assumes. Because PROC CAEFFECT considers only the estimation of potential outcome means and does not directly model the treatment or outcome variable, you must assess whether the modeling techniques that you are using will result in valid inference when you use this expression.

A special case when the expression for $\hat{\boldsymbol{V}}(\hat{\boldsymbol{\mu}})$ can provide valid inference, even if data splitting is not used, is IPW estimation with predicted treatment probabilities estimated by using maximum likelihood estimation. In that case, the expression for $\hat{\boldsymbol{V}}(\hat{\boldsymbol{\mu}})$ that treats the predicted treatment assignment probabilities as fixed is known to be conservative and to produce standard errors larger than the errors that you would obtain if you were to adjust for the estimation of the treatment model (Lunceford and Davidian 2004). Note that the same is not true of estimation by regression adjustment when you use maximum likelihood estimation for the outcome model and no data splitting.

Targeted Maximum Likelihood Estimation

For the targeted maximum likelihood estimation (TMLE) method, standard errors and confidence limits are computed by using the influence function for the estimator that is based on semiparametric efficiency theory.

The TMLE estimator is asymptotically linear, so

$$\hat{\boldsymbol{\mu}} - \boldsymbol{\mu} = \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{\phi}_i + o\left(\frac{1}{\sqrt{n}}\right)$$

where $\boldsymbol{\phi}_i$ is the influence function of the estimator for the ith observation. By definition, the mean of the influence function is the zero vector (Tsiatis 2006). Thus the covariance matrix $\hat{\boldsymbol{V}}(\hat{\boldsymbol{\mu}})$ for the estimates $\hat{\boldsymbol{\mu}}$ is given by

$$\hat{\boldsymbol{V}}(\hat{\boldsymbol{\mu}}) = \frac{1}{n^2}\sum_{i=1}^{n}\hat{\boldsymbol{\phi}}_i\hat{\boldsymbol{\phi}}_i^T$$

where n is the sample size and

$$\hat{\boldsymbol{\phi}}_i = \begin{bmatrix} \hat{h}_{t_1 i}\left(y_i - y^*_{1i}\right) + y^*_{1i} - \hat{\mu}_{t_1} \\ \vdots \\ \hat{h}_{t_\ell i}\left(y_i - y^*_{\ell i}\right) + y^*_{\ell i} - \hat{\mu}_{t_\ell} \end{bmatrix}$$

is an estimate of the influence function for each observation.

As with the methods based on estimating equations, the validity of this estimate for the covariance matrix and the rate of convergence that it assumes depend on a number of factors related to the modeling of the treatment and outcome variables. In particular, if you use the same data (that is, you do not use data splitting)

to compute the predicted treatment probabilities or predicted potential outcome means as well as to perform the targeting step and compute the influence functions, this can introduce dependencies that are unaccounted for in the covariance matrix estimate. Moreover, if the modeling technique that you use for the treatment or outcome variable converges at a rate slower than the square root of n, variability in the predicted quantities might lead to a rate of convergence for the potential outcome mean estimates that is slower than the rate that this approach to inference assumes. Because PROC CAEFFECT considers only the estimation of potential outcome means and does not directly model the treatment or outcome variable, you must assess whether the modeling techniques that you are using will result in valid inference when you use this expression.

As an alternative to sample splitting, ensemble modeling strategies based on cross validation, such as superlearning (Van der Laan and Rose 2011), have proved very effective when combined with TMLE for effect estimation.

Displayed Output

The following sections describe the output that PROC CAEFFECT produces. The output is organized into various tables, which are discussed in order of their appearance.

Model Information

The "Model Information" table displays basic information about the model, such as the input data, the treatment variable, and the estimation method.

Potential Outcome Mean Information

The "Potential Outcome Mean Information" table displays information about the potential outcome means that are estimated, such as the level of treatment that defines the potential outcomes and the source of predicted values that are used for their estimation.

Number of Observations

The "Number of Observations" table displays the number of observations that are read from the input data table and the number of observations that are used in the analysis. If you specify a FREQ statement, the table displays the sum of the frequencies that are read and used. If you specify a WEIGHT statement, the table displays the sum of the weighted frequencies that are read and used.

Treatment Profile

The "Treatment Profile" table displays the levels of the treatment variable that appear in the input data table and the number of observations in each level.

Response Profile

The "Response Profile" table displays the levels of the outcome variable that appear in the input data table and the number of observations in each level.

Potential Outcome Mean Covariance Matrix

The "Potential Outcome Mean Covariance Matrix" displays the estimated covariance matrix for the potential outcome means. Because of the model-agnostic approach that PROC CAEFFECT uses to perform estimation, the statistical validity of this estimate depends on model-specific conditions that you must evaluate and that the procedure cannot verify. For more information about how the procedure estimates the potential outcome mean covariance matrix and the conditions that are required for valid statistical inference, see the section "Standard Errors and Confidence Limits" on page 227.

Potential Outcome Mean Estimates

The "Potential Outcome Mean Estimates" table displays the estimated potential outcome means.

Potential Outcome Mean Differences

The "Potential Outcome Mean Differences" table displays the requested differences in potential outcome mean estimates.

Timing

The "Timing" table displays the amount of time (in seconds) that it took PROC CAEFFECT to perform different tasks in the analysis.

OutputCasTables Table

The OutputCasTables table is a special table that has information about each CAS table that is created during a CAS action execution. The information for each CAS table consists of the CAS table name, the caslib in which the table resides, and the number of columns and rows in the CAS table. Because this table is not a typical ODS table that contains analytical results, you cannot include it in the *table-spec-list* in the DISPLAYOUT statement.

ODS Table Names

Each table that the CAEFFECT procedure creates has a name that is associated with it. You must use this name to refer to the table when you use ODS statements. Table 6.3 lists the name of each table and a short description of its contents.

Table 6.3 ODS Tables Produced by PROC CAEFFECT

Table Name	Description	Statement	Option
ModelInfo	Information about the modeling environment	Default output	
NObs	Number of observations read and used	Default output	
OutputCasTables	See the section "OutputCasTables Table" on page 230	DISPLAYOUT	
POMCov	Potential outcome mean covariance matrix	PROC CAEFFECT	POMCOV

Table 6.3 *continued*

Table Name	Description	Statement	Option
POMDiffs	Potential outcome mean differences	DIFFERENCE	
POMEstimates	Potential outcome mean estimates	Default output	
POMInfo	Potential outcome mean information	PROC CAEFFECT	POMINFO
ResponseProfile	Response levels for categorical outcomes	Default output	
Timing	Absolute and relative times for tasks performed by the procedure	Default output	
TreatmentProfile	Treatment levels	Default output	

Examples: CAEFFECT Procedure

The examples in this chapter are available in the GitHub repository located at `https://github.com/sascommunities/doc-supplement-casstat`.

Example 6.1: Estimation by Inverse Probability Weighting

(View the complete code for this example (caeffectIPWEx1.sas) in the example repository.)

This example uses the SmokingWeight data set available in the examples for the CAUSALTRT procedure in SAS/STAT software. These data are a subset of data from the NHANES I Epidemiologic Follow-Up Study (NHEFS) in Hernán and Robins (2020). For the study, medical and behavioral information were collected during an initial physical examination and again at follow-up interviews about 10 years later. As in the PROC CAUSALTRT examples, the question of interest here is what effect does quitting smoking, as recorded in the variable Quit, have on an individual's change in weight (measured in kilograms), as recorded in the variable Change.

To create the data table SmokingWeight in your CAS session, you can modify the DATA step that is used to create the data set in PROC CAUSALTRT to create the data table in your CAS engine libref. For this example, it is assumed that your CAS engine libref is named mylib, but you can substitute any appropriately defined CAS engine libref.

In addition to the treatment variable Quit and outcome variable Change, the data include the following variables:

- Activity: level of daily activity, with values 0, 1, and 2
- Age: age in 1971
- BaseWeight: weight in kilograms in 1971
- Education: level of education, with values 0, 1, 2, 3, and 4
- Exercise: amount of regular recreational exercise, with values 0, 1, and 2

- PerDay: number of cigarettes smoked per day in 1971
- Race: 0 for white; 1 otherwise
- Sex: 0 for male; 1 for female
- Weight: weight in kilograms at the follow-up interview
- YearsSmoke: number of years an individual has smoked

For this example, it is assumed that the variables Activity, Age, BaseWeight, Education, Exercise, PerDay, Race, Sex, and YearsSmoke make up a valid adjustment set that you can use to obtain unbiased estimates of the potential outcome means.

The following statements use the LOGSELECT procedure to fit a logistic model for the treatment variable Quit:

```
proc logselect data=mylib.SmokingWeight;
   class Sex Race Education Exercise Activity;
   model Quit = Sex Race Education Exercise
                Activity Age YearsSmoke PerDay;
   store out=mylib.logTrtModel;
run;
```

The model for the treatment variable is specified in the MODEL statement. The STORE statement creates an analytic store that contains the fitted model in the logTrtModel data table.

The following statements use the SCORE statement in the ASTORE procedure to obtain the predicted probabilities of a subject being assigned to each treatment level:

```
proc astore;
   score data=mylib.SmokingWeight out=mylib.logScored
         rstore=mylib.logTrtModel copyvars=(Quit Change);
run;
```

You use the DATA= option to specify the data table to score, and you use the RSTORE= to specify the stored model to use for scoring. To avoid data duplication for large data tables, the variables in the input data table are not included in the output data table unless you specify them in the COPYVARS= option. To estimate the potential outcome means by inverse probability weighting, you need the treatment variable and observed outcome values, so the variables Quit and Change are specified in the COPYVARS= option.

The following statements use the CAEFFECT procedure to estimate potential outcome means by using inverse probability weighting (IPW):

```
proc caeffect data=mylib.logScored inference;
   treatvar Quit;
   outcomevar Change;
   pom treatlev = 1 treatProb = P_Quit1;
   pom treatlev = 0 treatProb = P_Quit0;
   difference reflev=0;
run;
```

When you use the CAEFFECT procedure, you must specify the TREATVAR statement and at least one POM statement. You specify the input data table to use for the potential outcome means estimation in the DATA=

option. The TREATVAR statement specifies information about the treatment variable. Each POM statement specifies a potential outcome of interest for your analysis.

For each potential outcome specification, you must use the TREATLEV= option to identify the level of treatment that defines the potential outcome. In this example, two potential outcomes are specified. One outcome is defined by the level 1, which corresponds to a subject quitting smoking, and the second outcome is defined by the level 0, which corresponds to a subject not quitting. For each potential outcome specification, you must also identify any variables in the input data table that are needed for the estimation. For IPW estimation, you must specify variables in the input data table that contain the previously computed treatment assignment probabilities; you do this by using the TREATPROB= option. In this example, the names of those variables are created by PROC ASTORE. If you do not know the names of the variables created by PROC ASTORE that contain the predicted values, you can use the DESCRIBE statement in PROC ASTORE and check the "Output Variables" table that it creates to find the variable names.

For IPW estimation, the observed outcome values are required, and therefore you must also specify the OUTCOMEVAR statement. Finally, in this example, you specify a DIFFERENCE statement to request a difference between the potential outcome mean estimates. A single difference is requested, and the REFLEV= option specifies that the treatment level 0 is used as the reference level. In general, you can specify multiple DIFFERENCE statements, and in each DIFFERENCE statement you can specify reference levels, event levels, or both.

The output from this analysis is displayed by default. It is presented in Output 6.1.1 through Output 6.1.4.

The "Model Information" table in Output 6.1.1 summarizes important information, including the data source, treatment variable name, estimation method, and type of outcome variable.

Output 6.1.1 Model Information

The CAEFFECT Procedure

Model Information	
Data Source	LOGSCORED
Treatment Variable	Quit
Outcome Variable	Change
Outcome Type	Continuous
Estimation Method	Inverse Probability Weighting

Output 6.1.2 displays the "Number of Observations" table. In this analysis, a number of observations are not used, because there are missing predicted treatment assignment probabilities. These probabilities are missing because there are missing values in the predictor variables.

Output 6.1.2 Number of Observations

Number of Observations Read	1746
Number of Observations Used	1566

Output 6.1.3 displays the "Treatment Profile" table. This table includes information about the levels of the treatment variable that appear in the input data table and the number of observations in each level that are used to estimate the potential outcome means.

Output 6.1.3 Treatment Profile

Treatment Profile	
Quit	Total Frequency
0	1163
1	403

Output 6.1.4 displays the potential outcome mean estimates and their difference. The "POM Estimates" table indicates the level of treatment that defines the potential outcome and the estimate of its mean. The "POM Differences" table shows the requested difference in potential outcome mean estimates. The table also includes nonprinting columns that contain the potential outcome mean estimates for the event and reference levels; these columns are accessible if you save the table as an output data set by using the ODS OUTPUT statement or if you modify the table's template. If the relevant causal assumptions hold true for the data-generating process that produced these data, then the difference in potential outcome mean estimates corresponds to an estimate of the average treatment effect (ATE) of a weight change that is approximately 3.25 kilograms greater because the subjects quit smoking.

Output 6.1.4 Potential Outcome Mean and Difference Estimates

POM Estimates

Treatment Level	Estimate	Standard Error	95% Confidence Limits		Z	Pr > |Z|
1	5.05146	0.47254	4.12530	5.97762	10.69	<.0001
0	1.80099	0.21938	1.37100	2.23097	8.21	<.0001

POM Differences

Treatment Levels							
Event	Reference	Difference	Standard Error	95% Confidence Limits		Z	Pr > |Z|
1	0	3.2505	0.52098	2.22936	4.27158	6.24	<.0001

By default, PROC CAEFFECT does not produce standard errors or confidence limits, because the validity of the covariance matrix estimate for the potential outcome mean estimates depends on factors outside the procedure invocation. In this example, where maximum likelihood estimation is used to estimate the treatment model and inverse probability weighting is used to estimate the potential outcome means, the variance estimation method that the procedure uses is known to produce valid but conservative statistical inference even without the use of data splitting. For this reason, standard errors and confidence limits are produced by using the INFERENCE option. For more information about the factors that determine the validity of the inferential statistics that the procedure produces, see the section "Standard Errors and Confidence Limits" on page 227.

You can also use the IPW estimation method in PROC CAEFFECT to produce estimates of the potential outcome means that are conditional on subjects receiving a particular level of treatment. To specify the level of treatment to condition on, you can use the CONDEVENT= option in the TREATVAR statement. The following statements request an analysis that uses the same data, but in this case for potential outcome mean estimates conditional on subjects quitting smoking:

```
proc caeffect data=mylib.logScored;
   treatvar Quit / condevent=1;
   outcomevar Change;
   pom treatlev = 1 treatProb = P_Quit1;
   pom treatlev = 0 treatProb = P_Quit0;
   difference reflev=0;
run;
```

The potential outcome mean estimates conditional on subjects quitting smoking and their difference are displayed in Output 6.1.5. The "POM Estimates" and "POM Differences" tables now include a column for the level of treatment that defines the conditioning event; this column is not displayed but is accessible if you save the table as an output data set by specifying the ODS OUTPUT statement. In this case, because the potential outcome that defines the event level in the difference comparison is the same as the level that defines the conditioning event, the estimated difference would correspond to an estimate of the average treatment effect for the treated (ATT).

Output 6.1.5 Conditional Potential Outcome Mean and Difference Estimates

The CAEFFECT Procedure

POM Estimates

Treatment Level	Estimate
1	4.52491
0	1.28412

POM Differences

Treatment Levels		
Event	Reference	Difference
1	0	3.2408

Example 6.2: Estimation by Regression Adjustment

(View the complete code for this example (caeffectRegAdjEx1.sas) in the example repository.)

This example uses the birthwgt data set available in the Sashelp library. The data are information about infant mortality in 2003 and were obtained from the US National Center for Health Statistics. A random sample of 100,000 observations is used in this example. The analysis of these data is purely illustrative, and definitive conclusions should not be drawn from it.

The following DATA step creates the data table mylib.birthwgt in your CAS session. For this example, it is assumed that your CAS engine libref is named mylib, but you can substitute any appropriately defined CAS engine libref.

```
data mylib.birthwgt;
   set sashelp.birthwgt;
run;
```

For this example, the treatment variable Smoking is an indicator of maternal smoking behavior, with values of 'Yes' and 'No'. The outcome variable of interest, Death, is an indicator of infant death within one year of

birth, with values of 'Yes' and 'No'. The following variables are assumed to be confounding variables that you can use in the model for the outcome variable, Death:

- AgeGroup represents maternal ages of less than 20, between 20 and 35, and greater than 35, with values 1, 2, and 3, respectively.
- Drinking is an indicator of maternal drinking during pregnancy, with values 'Yes' and 'No'.
- Married is an indicator of marital status, with values 'Yes' and 'No'.
- SomeCollege is an indicator of whether the mother has 12 or more years of education, with values 'Yes' and 'No'.

The following statements use the GRADBOOST procedure, which is part of SAS Viya machine learning software, to fit a model for the outcome variable, Death:

```
proc gradboost data=mylib.birthwgt ntrees=100 seed=8959;
   target Death / level=nominal;
   input  Smoking AgeGroup Married Drinking
          SomeCollege /level=nominal;
   savestate rstore=mylib.gbOutMod;
run;
```

The fitted gradient boosting model is saved in the data table gbOutMod, which you can provide as input to PROC CAEFFECT. The following statements use the saved outcome model to estimate two potential outcome means by regression adjustment:

```
proc caeffect data=mylib.birthwgt;
   treatvar Smoking;
   outcomevar Death( event='Yes') / type=Categorical;
   outcomemodel restore=mylib.gbOutMod predname=P_DeathYes;
   pom treatlev='Yes';
   pom treatlev='No';
run;
```

You use the POM statement to specify the potential outcomes of interest for your analysis. For each potential outcome specification, you must use the TREATLEV= option to identify the level of treatment that defines the potential outcome.

In this example, you use the TYPE= option in the OUTVAR statement to indicate that the outcome variable of interest is categorical. For categorical outcomes, PROC CAEFFECT estimates the probability of a designated outcome event level. The potential outcome means that you estimate therefore correspond to the counterfactual probability of that outcome occurring, and all predicted outcome values must be between 0 and 1, inclusive. You specify the designated event level for the outcome variable by using the EVENT= option.

When you provide a previously fit outcome model as input to PROC CAEFFECT, you use the OUTCOMEMODEL statement. You specify the name of the analytic store that contains the saved model by using the RESTORE= option. You must also use the PREDNAME= option to provide the name of the variable that contains the outcome of interest when you score data by using the analytic store. If you do not know the names of the variables that the analytic store creates, you can use the DESCRIBE statement in PROC ASTORE and check the "Output Variables" table that it produces to find the variable names.

The output from this analysis is displayed by default and is presented in Output 6.2.1 through Output 6.2.3.

The "Model Information" table in Output 6.2.1 summarizes important information, including the data source, treatment variable name, estimation method, and type of outcome variable.

Output 6.2.1 Model Information

The CAEFFECT Procedure

Model Information	
Data Source	BIRTHWGT
Treatment Variable	Smoking
Outcome Type	Categorical
Estimation Method	Regression Adjustment
Outcome Model Store	GBOUTMOD

Output 6.2.2 displays the "Number of Observations" table. In this analysis, a number of observations are not used, because the treatment variable Smoking has missing values. To indicate that a missing value is a valid value of the treatment variable, you can specify the COUNTMISSING option in the TREATVAR statement.

Output 6.2.2 Number of Observations

Number of Observations Read	100000
Number of Observations Used	94388

Output 6.2.3 displays the potential outcome mean estimates that are obtained by using the gradient boosting model for the outcome variable.

Output 6.2.3 Estimates Using a Stored Model

POM Estimates	
Treatment Level	**Estimate**
Yes	0.00829
No	0.00640

As an alternative to providing a previously fit outcome model as input to PROC CAEFFECT, you can also specify variables in the input data table that contain predicted counterfactual outcome values for each treatment level of interest. You specify the names of these variables by using the PREDOUT= option for each potential outcome specification in a POM statement. To obtain these predicted values, you can use a fitted model for the outcome and score the data with the treatment set to the counterfactual level of interest.

The following statements demonstrate one way to obtain predicted counterfactual outcome values that you can provide as input to PROC CAEFFECT. This example uses PROC ASTORE to score the data by using the previously fit model that is saved in the gbOutMod data table.

The first step is to create a table that renames the observed treatment variable tempSmoking and sets the value of the treatment variable Smoking to the counterfactual level 'Yes' for scoring the data. The following DATA step performs these tasks and creates the gbPredData data table:

```
data mylib.gbPredData;
   set mylib.birthwgt;
   tempSmoking = Smoking;
   Smoking = 'Yes';
run;
```

After creating the gbPredData data table, you can score the counterfactual outcomes by using PROC ASTORE, as in the following statements. To avoid data duplication for large data tables, the variables in the input data table are not included in the output data table that PROC ASTORE creates unless you specify them in the COPYVARS= option. To score the counterfactual outcomes for the treatment level 'No', you must include all the predictors that are used in the outcome model in the COPYVARS= option along with the observed treatment values that are stored in the variable tempSmoking.

```
proc astore;
   score data=mylib.gbPredData out=mylib.gbPredData
         rstore=mylib.gbOutMod
         copyvars=(tempSmoking AgeGroup Married
                   Drinking SomeCollege Death);
run;
```

Because PROC ASTORE uses the same variable names for the predicted values to score the next counterfactual values, you must first rename the variable that contains the predicted outcome probability of interest. You must also set the value of the treatment variable Smoking to the level 'No' to score the remaining counterfactual outcomes. These steps are performed using the following statements:

```
data mylib.gbPredData;
   set mylib.gbPredData;
   rename P_DeathYes = SmokingPred;
   Smoking = 'No';
run;
```

Next, the data are scored again for the counterfactual treatment level 'No':

```
proc astore;
   score data=mylib.gbPredData out=mylib.gbPredData
         rstore=mylib.gbOutMod
         copyvars=(tempSmoking SmokingPred Death);
run;
```

Because there are no additional counterfactual values to score, you can limit the list of variables in the COPYVARS= option to only those that you need for the effect estimation. Although it is not necessary to do this before using PROC CAEFFECT, you can use the following statements to rename the variable that contains the newly predicted counterfactual outcome values, and you can also replace the temporary name that was applied to the observed treatment variable with the original variable name, Smoking:

```
data mylib.gbPredData;
   set mylib.gbPredData;
   rename P_DeathYes = NoSmokingPred tempSmoking=Smoking;
run;
```

The following statements use the previously computed predicted counterfactual outcome values as input to estimate the potential outcome means:

```
proc caeffect data=mylib.gbPredData;
   treatvar Smoking;
   outcomevar Death( event='Yes') / type=Categorical;
   pom treatlev='Yes' predOut=SmokingPred;
   pom treatlev='No'  predOut=NoSmokingPred;
run;
```

Output 6.2.4 displays the potential outcome mean estimates that are obtained by using the previously estimated counterfactual outcome values. These estimates of the potential outcome means are the same as the estimates shown in Output 6.2.3, because the same model for the outcome variable is used to obtain the predictions.

Output 6.2.4 Estimates Using Previously Predicted Values

The CAEFFECT Procedure

POM Estimates

Treatment Level	Estimate
Yes	0.00829
No	0.00640

Example 6.3: Estimation by Doubly Robust Methods

(View the complete code for this example (caeffectDREx1.sas) in the example repository.)

This example uses the mylib.SmokingWeight data table from "Example 6.1: Estimation by Inverse Probability Weighting" on page 231. The same causal question is examined, but this example uses the augmented inverse probability weighting (AIPW) and targeted maximum likelihood estimation (TMLE) methods.

The AIPW and TMLE methods are doubly robust methods, and they combine modeling of both the treatment and outcome variables. The following statements use the LOGSELECT procedure to fit a logistic model for the treatment variable Quit and create an output data table that contains the predictions from the fitted model:

```
proc logselect data=mylib.SmokingWeight;
   class Sex Race Education Exercise Activity;
   model Quit(Event='1') = Sex Race Education
         Exercise Activity Age YearsSmoke PerDay;
   output out=mylib.swDREstData pred=pTrt copyvars=(_ALL_);
run;
```

The EVENT= response variable option in the MODEL statement ensures that the LOGSELECT procedure models the probability that the variable Quit takes the value 1. In the OUTPUT statement, the PRED= option names the variable in the output data table that contains the predicted event probability, and the COPYVARS= option copies all variables from the input data table to the output data table.

The following DATA step creates the variable pCnt, which contains the predicted probability of being assigned to the control condition, Quit=0:

```
data mylib.swDREstData;
   set mylib.swDREstData;
   pCnt = 1 - pTrt;
run;
```

An alternative way to obtain the predicted treatment probabilities is to save the fitted treatment model as an analytic store and use the ASTORE procedure. This approach is demonstrated in "Example 6.1: Estimation by Inverse Probability Weighting" on page 231.

The following statements use the BART procedure to fit a model for the outcome variable by using Bayesian additive regression trees (BART). The model for this example is fit by using an ensemble of 100 trees; 200 samples of the ensemble are saved for prediction, and the fitted model is saved in the analytic store bartOutMod.

```
proc bart data=mylib.swDREstData nTree=100 nMC=200 seed=2156;
   class Sex Race Education Exercise Activity Quit;
   model Change = Sex Race Education Exercise Quit Activity
         Age YearsSmoke PerDay;
   store out=mylib.bartOutMod;
run;
```

The following statements use the CAEFFECT procedure to estimate potential outcome means by using the AIPW estimation method and to compute the difference between the two potential outcomes:

```
proc caeffect data=mylib.swDREstData method=aipw;
   treatvar Quit;
   outcomevar Change;
   outcomemodel restore=mylib.bartOutMod predName=P_Change;
   pom treatLev=1 treatProb=pTrt;
   pom treatLev=0 treatProb=pCnt;
   difference evtLev=1;
run;
```

When you use PROC CAEFFECT, at least one POM statement is required in order to specify the potential outcomes of interest, and the TREATVAR statement is required in order to specify the treatment variable of interest. For the AIPW estimation method, you must also specify the outcome variable by using the OUTCOMEVAR statement. For each potential outcome specification, you use the TREATLEV= option to identify the level of treatment that defines the potential outcome, and you use the TREATPROB= option to specify the variable in the input data table that contains the predicted probability of being assigned to that level.

The AIPW estimation method also requires predicted counterfactual values for each observation. For each potential outcome of interest, the counterfactual predictions are obtained by replacing the observed treatment level with the level that defines the potential outcome. In this example, the previously fit BART model is specified by using the OUTCOMEMODEL statement, and the procedure uses the previously fit model to predict counterfactual outcome values. When you provide as input to PROC CAEFFECT an analytic store that contains a model for the outcome variable, the input data table must contain the predictor variables that the model requires, and you must specify the names of the variables created by the analytic store that contains the predicted values. If you do not know these variable names, you can use the DESCRIBE statement in PROC ASTORE to describe the store.

As an alternative to providing a previously fit outcome model as input to PROC CAEFFECT, you can specify variables in the input data table that contain predicted counterfactual outcome values for each treatment level

of interest. You specify the names of these variables by using the PREDOUT= option for each potential outcome specification in a POM statement. An example of how you can use this alternative syntax and how you can obtain the necessary counterfactual predictions is provided by "Example 6.2: Estimation by Regression Adjustment" on page 235.

The output from this analysis is displayed by default and is presented in Output 6.3.1 through Output 6.3.3.

The "Model Information" table in Output 6.3.1 summarizes important information, including the data source, treatment variable name, estimation method, type of outcome variable, and name of the analytic store used for outcome modeling.

Output 6.3.1 Model Information

The CAEFFECT Procedure

Model Information	
Data Source	SWDRESTDATA
Treatment Variable	Quit
Outcome Variable	Change
Outcome Type	Continuous
Estimation Method	AIPW
Outcome Model Store	BARTOUTMOD

Output 6.3.2 displays the "Number of Observations" table. In this analysis, a number of observations are not used, because there are missing predicted treatment assignment probabilities. These probabilities are missing because there are missing values in the predictor variables and because a generalized linear model, in particular logistic regression, is used to model the treatment variable. The BART model that is used for the outcome variable can predict counterfactual outcome values even for observations that have missing predictor values.

Output 6.3.2 Number of Observations

Number of Observations Read	1746
Number of Observations Used	1566

Output 6.3.3 displays the potential outcome mean estimates and their difference. The "POM Estimates" table indicates the level of treatment that defines the potential outcome and the estimate of its mean. The "POM Differences" table shows the requested difference in potential outcome mean estimates. With the proper causal assumptions, the difference in potential outcome mean estimates corresponds to an estimate of the average treatment effect (ATE) of a weight change that is approximately 3.27 kilograms greater because the subjects quit smoking. This value is comparable to the estimate that is obtained in "Example 6.1: Estimation by Inverse Probability Weighting" on page 231.

Output 6.3.3 Potential Outcome Mean and Difference Estimates

POM Estimates	
Treatment Level	**Estimate**
1	5.12356
0	1.78772

Output 6.3.3 *continued*

POM Differences		
Treatment Levels		
Event	Reference	Difference
1	0	3.3358

The TMLE method is a doubly robust method that you can use as an alternative to the AIPW method. The required inputs are the same for both methods. The following statements use PROC CAEFFECT to estimate potential outcome means by using the TMLE method and to compute the difference between the two potential outcomes:

```
proc caeffect data=mylib.swDREstData method=tmle;
   treatvar Quit;
   outcomevar Change;
   outcomemodel restore=mylib.bartOutMod predName=P_Change;
   pom treatLev=1 treatProb=pTrt;
   pom treatLev=0 treatProb=pCnt;
   difference evtLev=1;
run;
```

The syntax is identical to the syntax that is used to invoke the AIPW method, except that the METHOD= option now specifies the TMLE method.

As in the analysis that uses the AIPW method, the output from the analysis that uses the TMLE method is displayed by default.

The "Model Information" table in Output 6.3.4 summarizes important information, including the data source, treatment variable name, estimation method, type of outcome variable, and name of the analytic store used for outcome modeling.

Output 6.3.4 Model Information

The CAEFFECT Procedure

Model Information	
Data Source	SWDRESTDATA
Treatment Variable	Quit
Outcome Variable	Change
Outcome Type	Continuous
Estimation Method	TMLE
Outcome Model Store	BARTOUTMOD

Output 6.3.5 displays the "Number of Observations" table. It is identical to the table that is produced by the analysis that uses the AIPW method.

Output 6.3.5 Number of Observations

Number of Observations Read	1746
Number of Observations Used	1566

Output 6.3.6 displays the potential outcome mean estimates and their difference. The values in these tables are almost the same as the values in the corresponding tables that are produced by the AIPW method.

Output 6.3.6 Potential Outcome Mean and Difference Estimates

POM Estimates

Treatment Level	Estimate
1	5.12564
0	1.78811

POM Differences

Treatment Levels		
Event	Reference	Difference
1	0	3.3375

References

Cole, S. R., and Frangakis, C. E. (2009). "The Consistency Statement in Causal Inference: A Definition or an Assumption?" *Epidemiology* 20:3–5.

Elwert, F. (2013). "Graphical Causal Models." In *Handbook of Causal Analysis for Social Research*, edited by S. L. Morgan, 245–273. Dordrecht: Springer.

Gruber, S., and Van der Laan, M. J. (2009). "Targeted Maximum Likelihood Estimation: A Gentle Introduction." Working paper no. 252, University of California, Berkeley.

Hernán, M. A., and Robins, J. M. (2020). *Causal Inference: What If*. Boca Raton, FL: Chapman & Hall/CRC.

Imbens, G. W., and Rubin, D. B. (2015). *Causal Inference for Statistics, Social, and Biomedical Sciences: An Introduction*. New York: Cambridge University Press.

Lunceford, J. K., and Davidian, M. (2004). "Stratification and Weighting via the Propensity Score in Estimation of Causal Treatment Effects: A Comparative Study." *Statistics in Medicine* 23:2937–2960.

Murnane, R. J., and Willett, J. B. (2011). *Methods Matter: Improving Causal Inference in Educational and Social Science Research*. New York: Oxford University Press.

Neyman, J., Dabrowska, D. M., and Speed, T. P. (1990). "On the Application of Probability Theory to Agricultural Experiments: Essay on Principles, Section 9." *Statistical Science* 5:465–472. Translated and edited by Dabrowska and Speed from the Polish original by Neyman (1923).

Pearl, J. (2009a). "Causal Inference in Statistics: An Overview." *Statistics Surveys* 3:96–146.

Pearl, J. (2009b). *Causality: Models, Reasoning, and Inference*. 2nd ed. Cambridge: Cambridge University Press.

Rubin, D. B. (1980). "Comment on D. Basu, 'Randomization Analysis of Experimental Data: The Fisher Randomization Test'." *Journal of the American Statistical Association* 75:591–593.

Rubin, D. B. (1990). "Comment: Neyman (1923) and Causal Inference in Experiments and Observational Studies." *Statistical Science* 5:472–480.

Shpitser, I., VanderWeele, T., and Robins, J. M. (2010). "On the Validity of Covariate Adjustment for Estimating Causal Effects." In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, edited by P. Grünwald and P. Spirtes, 527–536. Corvallis, OR: AUAI Press.

Spirtes, P., Glymour, C., and Scheines, R. (2001). *Causation, Prediction, and Search*. 2nd ed. Cambridge, MA: MIT Press.

Stefanski, L. A., and Boos, D. D. (2002). "The Calculus of M-Estimation." *American Statistician* 56:29–38.

Tsiatis, A. A. (2006). *Semiparametric Theory and Missing Data*. New York: Springer.

Van der Laan, M. J. (2010a). "Targeted Maximum Likelihood Based Causal Inference: Part I." *International Journal of Biostatistics* 6:2.

Van der Laan, M. J. (2010b). "Targeted Maximum Likelihood Based Causal Inference: Part II." *International Journal of Biostatistics* 6:3.

Van der Laan, M. J., and Rose, S. (2011). *Targeted Learning: Causal Inference for Observational and Experimental Data*. New York: Springer.

VanderWeele, T. J. (2009). "Concerning the Consistency Assumption in Causal Inference." *Epidemiology* 20:880–883.

Chapter 7
CARDINALITY Procedure

Contents

Overview: CARDINALITY Procedure **246**
PROC CARDINALITY Features 246
PROC CARDINALITY Compared to Other SAS Procedures 247
Using CAS Sessions and CAS Engine Librefs 247
Getting Started: CARDINALITY Procedure **248**
Syntax: CARDINALITY Procedure **250**
PROC CARDINALITY Statement 251
FREQ Statement 252
VAR Statement 252
WEIGHT Statement 253
Details: CARDINALITY Procedure **253**
Displayed Output 254
Examples: CARDINALITY Procedure **255**
Example 7.1: Limited Cardinality of the Species Variable 255
Example 7.2: Limited Cardinality of the Sepal Length Variable 257
Example 7.3: More Levels of the Sepal Length Variable 259
Example 7.4: A Variable with a User-Defined Format 261
Example 7.5: Forcing Another Order on the engineSize Variable 262
References **264**

Overview: CARDINALITY Procedure

The CARDINALITY procedure determines a variable's cardinality or limited cardinality in SAS Viya. The cardinality of a variable is the number of its distinct values, and the limited cardinality of a variable is the number of its distinct values that do not exceed a specified threshold.

In order to decide whether to include a variable in a subsequent SAS analysis as a classification or interval variable, it is sufficient to compute only the limited cardinality (and not the full cardinality) because both tend to yield the same decision. The limited cardinality is computationally less expensive than the full cardinality, especially for big data.

The CARDINALITY procedure creates the following:

- a cardinality data table, which contains summary information for each variable and some additional statistics about numeric variables
- a details data table, which contains the levels of each variable

PROC CARDINALITY Features

The CARDINALITY procedure has the following features:

- treats all variables as classification variables and attempts to determine the highest levels of each variable not to exceed a specified limit
- performs a single pass to determine the limited cardinality of each variable
- runs with all the input variables in the data table or with a specified list of variables
- can be run multiple times with a WHERE clause to reveal more levels. The results are not affected by the number of units doing the work (distributed or threaded or both) because PROC CARDINALITY produces the highest levels of each variable without exceeding the specified limit.
- recommends a level (CLASS, INTERVAL, or ID) for each variable. You can override these recommended levels in subsequent steps.
- can be run multiple times to extract subsequent segments of the full histogram of any variable
- builds the levelization in an ascending or descending order with or without formats

PROC CARDINALITY Compared to Other SAS Procedures

Other SAS procedures, such as the SUMMARY procedure, can calculate the cardinality of variables, but some are expensive to compute and result in a huge data set, and others yield nondeterministic results (that is, the results depend on the number of working units).

Using CAS Sessions and CAS Engine Librefs

SAS Cloud Analytic Services (CAS) is the analytic server and associated cloud services in SAS Viya. This section describes how to create a CAS session and set up a CAS engine libref that you can use to connect to the CAS session. It assumes that you have a CAS server already available; contact your system administrator if you need help starting and terminating a server. This CAS server is identified by specifying the host on which it runs and the port on which it listens for communications. To simplify your interactions with this CAS server, the host information and port information for the server are stored as SAS option values that are retrieved automatically whenever this CAS server needs to be accessed. You can examine the host and port values for the server at your site by using the following statements:

```
proc options option=(CASHOST CASPORT);
run;
```

In addition to starting a CAS server, your system administrator might also have created a CAS session and a CAS engine libref for your use. You can define your own sessions and CAS engine librefs that connect to the CAS server as shown in the following statements:

```
cas mysess;
libname mylib cas sessref=mysess;
```

The CAS statement creates the CAS session named mysess, and the LIBNAME statement creates the mylib CAS engine libref that you use to connect to this session. It is not necessary to explicitly name the CASHOST and CASPORT of the CAS server in the CAS statement, because these values are retrieved from the corresponding SAS option values.

If you have created the mysess session, you can terminate it by using the TERMINATE option in the CAS statement as follows:

```
cas mysess terminate;
```

For more information about the CAS and LIBNAME statements, see the section “Introduction to Shared Concepts” on page 6 in Chapter 2, “Shared Concepts.”

Getting Started: CARDINALITY Procedure

(View the complete code for this example at https://github.com/sascommunities/doc-supplement-casstat/tree/main/Examples/cardinality_gs.sas.)

NOTE: Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections "Using CAS Sessions and CAS Engine Librefs" on page 7 and "Loading a SAS Data Set onto a CAS Server" on page 7 in Chapter 2, "Shared Concepts."

This example uses the Iris data set as input to demonstrate how to use PROC CARDINALITY. The Iris data published by Fisher (1936) include the species of iris and the sepal length, sepal width, petal length, and petal width (which are measured in millimeters) on 50 iris specimens from each of three species: *Iris setosa, I. versicolor, and I. virginica*.

You can load the sashelp.iris data set into your CAS session by naming your CAS engine libref in the first statement of the following DATA step:

```
data mylib.iris;
   set sashelp.iris;
run;
```

These statements assume that your CAS engine libref is named mylib, but you can substitute any appropriately defined CAS engine libref.

The following statements show the basic usage:

```
proc cardinality data=mylib.iris outcard=mylib.card
                outdetails=mylib.details maxlevels=10;
run;
```

The DATA= option names mylib.iris as the input data table. Two output data tables are requested: the required OUTCARD= option requests that cardinality and summary information be stored in the mylib.card data table, and the OUTDETAILS= option requests that the levels found for each variable be stored in the mylib.details data table. The MAXLEVELS= option specifies 10 as the maximum number of levels to report.

The following statements produce the "Cardinality Output Data" table, shown in Figure 7.1. This table contains one row for each variable in the input data table. If you do not want to display all the variables in the input data table, you can use the VAR statement to specify which variables you want to display.

```
proc print data=mylib.card;
   var _varname_ _type_ _cardinality_ _more_;
run;
```

Figure 7.1 Cardinality Output Data Table

Obs	_VARNAME_	_TYPE_	_CARDINALITY_	_MORE_
1	Species	C	3	N
2	SepalLength	N	10	Y
3	SepalWidth	N	10	Y
4	PetalLength	N	10	Y
5	PetalWidth	N	10	Y

Table 7.1 explains the columns in the "Cardinality Output Data Table." This example works with a simple subset of the variables in both the cardinality and details data tables.

Table 7.1 Variables from the Cardinality Data

Name	Description
VARNAME	Variable name
TYPE	Variable type (N for numeric or C for character)
CARDINALITY	Number of levels extracted (less than or equal to the value of the MAXLEVELS= option)
MORE	Indication of more unreported levels (Y to indicate more levels or N to indicate no more levels)

Figure 7.1 shows that the Species variable has _TYPE_ = C, indicating that it is a character variable; its _CARDINALITY_ value is 3, indicating that three levels (values) are reported in the details data; and _MORE_ = N, indicating that there are no unreported levels.

Figure 7.1 shows that the SepalLength variable is a numeric variable (_TYPE_ = N); its _CARDINALITY_ value is 10, indicating that 10 different values are reported; and _MORE_ = Y, indicating that there are still more levels that can be explored.

The following statements produce the details output data table, as shown in Figure 7.2:

```
data details;
   set mylib.details;
   where _varname_ in ('Species', 'SepalLength');
run;

proc print data=details;
   var _VARNAME_ _INDEX_ _FREQ_ _RAWNUM_ _RAWCHAR_;
run;
```

Figure 7.2 Details Output Data Table

Obs	_VARNAME_	_INDEX_	_FREQ_	_RAWNUM_	_RAWCHAR_
1	Species	1	50	.	Setosa
2	Species	2	50	.	Versicolor
3	Species	3	50	.	Virginica
4	SepalLength	1	1	43	
5	SepalLength	2	3	44	
6	SepalLength	3	1	45	
7	SepalLength	4	4	46	
8	SepalLength	5	2	47	
9	SepalLength	6	5	48	
10	SepalLength	7	6	49	
11	SepalLength	8	10	50	
12	SepalLength	9	9	51	
13	SepalLength	10	4	52	
14	SepalLength	.	105	52	

Table 7.2 explains the columns in the details output data table.

Table 7.2 Variables from the Details Output Data Table

Name	Description
VARNAME	Variable name
INDEX	Index of the level
FREQ	Frequency of the level
RAWNUM	Raw level of the variable if numeric
RAWCHAR	Raw level of the variable if character

The Species variable has three rows in the details data table, one row for each of the reported levels (the _CARDINALITY_ variable in the "Cardinality Output Data Table" has the value 3). Each of these rows has _FREQ_ = 50, and the _RAWCHAR_ values are reported for each level. There are no more unreported levels for the variable Species (_MORE_ = N); thus, no row has a missing value (.) in the _INDEX_ column for the Species variable.

The "Details Output Data Table" reports the top 10 levels of the SepalLength variable and shows that there are still as many as 105 unreported levels that are greater than 52. (For the SepalLength variable, _CARDINALITY_ = 10 and _MORE_ = Y in the "Cardinality Output Data Table.") Therefore, SepalLength occupies 11 rows in the details data table (10 reported levels and one additional level that describes the group of all unreported levels.) The _INDEX_ of the unreported group of levels for the SepalLength variable is missing, reminding you that many levels make up this group. Essentially, the last row lumps all the other SepalLength rows into one level that has a missing _INDEX_ value. So you see 10 levels in addition to the levels greater than 52, a total of 11 rows for the SepalLength variable. The last row, which includes a missing _INDEX_ value, contains enough information to run the procedure again with a WHERE clause to obtain the next set of details.

Syntax: CARDINALITY Procedure

The following statements are available in the CARDINALITY procedure:

```
PROC CARDINALITY OUTCARD=libref.data-table < options > ;
    FREQ variable ;
    VAR variables < / options > ;
    WEIGHT variable ;
```

The PROC CARDINALITY statement is required.

PROC CARDINALITY Statement

PROC CARDINALITY OUTCARD=*libref.data-table* < *options* > **;**

The PROC CARDINALITY statement invokes the procedure. Table 7.3 summarizes the options available in the PROC CARDINALITY statement.

Table 7.3 PROC CARDINALITY Statement Options

Option	Description
Input and Output Data Sets	
DATA=	Specifies the input data table
OUTCARD=	Names the cardinality data table to contain the summary information for each variable
OUTDETAILS=	Names an optional data table to contain the detailed levels of all the variables
Cardinality Options	
MAXLEVELS=	Specifies the maximum number of levels to consider
ORDER=	Specifies the order to be applied to all variables that have an unspecified order

You must specify the following option:

OUTCARD=*libref.data-table*

specifies the output data table to contain the cardinality and summary information for each variable. *libref.data-table* is a two-level name, where *libref* refers to the library, and *data-table* specifies the name of the output data table. For more information about this two-level name, see the DATA= option and the section "Using CAS Sessions and CAS Engine Librefs" on page 247.

You can also specify the following *options*:

DATA=*libref.data-table*

names the input data table for PROC CARDINALITY to use. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to the data, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 247.

data-table specifies the name of the input data table.

MAXLEVELS=*number*

specifies the maximum *number* of levels of the variables in the input data to consider. The *number* should be between 5 and 254.

By default, MAXLEVELS=20.

ORDER=ASC | DESC | ASCFMT | DESFMT
specifies the order to be applied to all variables whose order is not specified in a VAR statement.

You can specify the following values:

ASC	orders variables in unformatted ascending order.
DESC	orders variables in unformatted descending order.
ASCFMT	orders variables in formatted ascending order.
DESFMT	orders variables in formatted descending order.

By default, ORDER=ASC.

OUTDETAILS=*libref.data-table*
specifies the output data table to contain the levels that are found for each variable. *libref.data-table* is a two-level name, where *libref* refers to the library, and *data-table* specifies the name of the input data table. For more information about this two-level name, see the DATA= option and the section "Using CAS Sessions and CAS Engine Librefs" on page 247.

All the levels that are higher than the value of the MAXLEVELS= option are grouped together into one level that has a missing value (.) in the _INDEX_ column in the output data table. For this level, the corresponding raw and formatted levels contain the last values visible in the histogram. This is useful when you need the next segment of the histogram past the visible levels of a variable.

FREQ Statement

FREQ *variable* **;**

The *variable* in the FREQ statement identifies a numeric variable in the input data table that contains the frequency of occurrence of each observation. PROC CARDINALITY treats each observation as if it appears f times, where f is the value of the FREQ *variable* for the observation. If f is not an integer, it is truncated to an integer. If f is less than 1 or missing, the observation is not used in the analysis. When you omit the FREQ statement, each observation is assigned a frequency of 1.

Missing and nonpositive values do not contribute to the other statistics.

VAR Statement

VAR *variables* < / *options* > **;**

The VAR statement enables you to specify a subset of *variables* in the input data table (which is specified in the DATA= option in the PROC CARDINALITY statement) and specify how to order them.

You can specify one or more VAR statements for each type of ordering that you want. You cannot specify a variable more than once in all the VAR statements.

ORDER=ASC | DESC | ASCFMT | DESFMT
requests a specific method of levelization. You can specify the following values:

ASC	orders the specified *variables* in unformatted ascending order.
DESC	orders the specified *variables* in unformatted descending order.
ASCFMT	orders the specified *variables* in formatted ascending order.
DESFMT	orders the specified *variables* in formatted descending order.

By default, the value of the ORDER= option in the PROC CARDINALITY statement is used. If the ORDER= option is not specified in the PROC CARDINALITY statement, ORDER=ASC by default if the variable is unformatted or ORDER=ASCFMT by default if the variable is formatted.

WEIGHT Statement

WEIGHT *variable* **;**

The *variable* in the WEIGHT statement is used as a weight to perform a weighted analysis of the data. Observations that have nonpositive or missing weights are not included in the analysis. When you omit the WEIGHT statement, all observations that are used in the analysis are assigned a weight of 1.

Details: CARDINALITY Procedure

The CARDINALITY procedure extracts from a specified data table the distinct values of its variables up to a specified maximum number of levels. The maximum number of levels that you specify cannot exceed 254. The number of distinct levels of a variable is called its cardinality.

You need this information in a data exploration phase in order to understand the contents of big data and the possible role that a variable can play in subsequent analyses.

A limited cardinality study is sufficient to determine the variable roles—class, interval, or ID (a record identifier)—on the basis of the distinct values that are observed from the data. A full cardinality study might be computationally prohibitive, because it requires a lot of memory and resources. You can arrive at the same conclusion by running a limited cardinality analysis on the data.

The limited cardinality study reads and processes data until the number of levels reaches the value that you specify in the MAXLEVELS= option. All the levels that are higher than the value of this option are grouped together into one level that has a missing value (.) in the _INDEX_ column in the output details data table. All variables whose visibility is 100 are identified as having a class role. Any nonnumeric variable whose number of levels is greater than the MAXLEVELS= option value is identified as having an ID role. A numeric variable whose number of levels is greater than the MAXLEVELS= option value is identified as having either an interval or ID role, depending on whether all the processed data of the variable are unique or not. If all the processed data of the variable are unique, then the variable is identified as having an ID role; otherwise it is identified as having an interval role.

Displayed Output

The output data tables have fixed variable names and labels. Each variable occupies one row in the cardinality data table; the number of rows that it can occupy in the details data table is less than or equal to the value of the MAXLEVELS= option plus 1. The following statements run PROC CARDINALITY and then PROC CONTENTS to examine contents of the CARD table:

```
proc cardinality data=mylib.iris outcard=mylib.card
                 outdetails=mylib.details maxlevels=10;
run;

data card;
   set mylib.card;
run;

proc contents data=card order=varnum;
run;
```

Figure 7.3 Contents of the Cardinality Output Data Table

The CONTENTS Procedure

Variables in Creation Order

#	Variable	Type	Len	Format	Label
1	_VARNAME_	Char	32	$.	Variable name
2	_FMTWIDTH_	Num	8	BEST.	Width of the variable formatted value
3	_TYPE_	Char	1	$.	Type of the raw values
4	_RLEVEL_	Char	10	$.	Recommended level for analytics
5	_ORDER_	Char	8	$.	Variable sort order
6	_MORE_	Char	1	$.	Have more unreported levels
7	_CARDINALITY_	Num	8	BEST.	Number of levels
8	_NOBS_	Num	8	BEST.	Number of observations
9	_SUMFREQS_	Num	8	BEST.	Total summation of frequencies
10	_NMISS_	Num	8	BEST.	Number of missing values
11	_MISSFMT_	Char	32	$.	Format of the missing value
12	_VISIBLE_	Num	8	BEST.	Percentage of the visible part of the report
13	_MIN_	Num	8	BEST.	Minimum numeric value
14	_MAX_	Num	8	BEST.	Maximum numeric value
15	_MEAN_	Num	8	BEST.	Mean
16	_STDDEV_	Num	8	BEST.	Standard deviation
17	_SKEWNESS_	Num	8	BEST.	Skewness
18	_KURTOSIS_	Num	8	BEST.	Kurtosis
19	_MFREQ_	Num	8	BEST.	Maximum frequency
20	_MFREQFOUNDLEVEL_	Char	1	$.	Found maximum frequency in the visible part of the report
21	_MFREQNUM_	Num	8	BEST.	Numeric level with the maximum frequency
22	_MFREQCHR_	Char	32	$.	Character level with the maximum frequency
23	_MFREQCFMT_	Char	32	$.	Formatted level with the maximum frequency
24	_LASTNUM_	Num	8	BEST.	Last raw numeric value (cutoff)
25	_LASTCHR_	Char	32	$.	Last raw character value (cutoff)
26	_LASTCFMT_	Char	32	$.	Last formatted value (cutoff)

Note that even though the numbers of columns is fixed, the width of some of the columns in both data tables (for example, _RAWCHAR_ and _CFMT_) is related to the raw and formatted data values of each input variable. The width of these two variables is set to the maximum width over all the variables in the reports.

Figure 7.4 Contents of the Details Output Data Table

The CONTENTS Procedure

Variables in Creation Order

#	Variable	Type	Len	Format	Label
1	_VARNAME_	Char	32	$.	Variable name
2	_INDEX_	Num	8	BEST.	Level index
3	_FREQ_	Num	8	BEST.	Level frequency
4	FREQPERCENT	Num	8	BEST.	Percentage of the frequency of current level
5	NMISSPERCENT	Num	8	BEST.	Percentage of non missing values
6	_RAWNUM_	Num	8	BEST.	Raw numeric value
7	_RAWCHAR_	Char	32	$.	Raw character value
8	_FMTWIDTH_	Num	8	BEST.	Width of the variable formatted value
9	_CFMT_	Char	32	$.	Formatted value of a variable

Examples: CARDINALITY Procedure

The examples in this chapter are available in the GitHub repository located at https://github.com/sascommunities/doc-supplement-casstat.

NOTE: Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections "Using CAS Sessions and CAS Engine Librefs" on page 7 and "Loading a SAS Data Set onto a CAS Server" on page 7 in Chapter 2, "Shared Concepts."

Example 7.1: Limited Cardinality of the Species Variable

(View the complete code for this example (cardinality_ex1.sas) in the example repository.)

This example illustrates how you can use PROC CARDINALITY to determine the limited cardinality of a variable. It uses the same input data table as is used in the section "Getting Started: CARDINALITY Procedure" on page 248.

You can load the sashelp.iris data set into your CAS session by naming your CAS engine libref in the first statement of the following DATA step:

```
data mylib.iris;
   set sashelp.iris;
run;
```

The following statements run PROC CARDINALITY and plot the levels from the details data set for the variable Species:

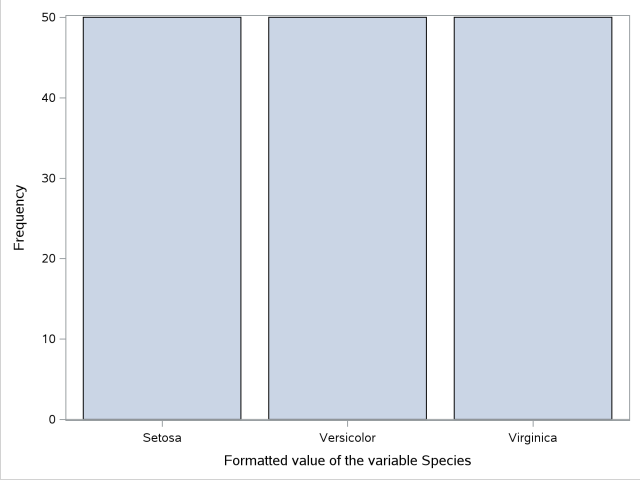
```
proc cardinality data=mylib.iris outcard=mylib.card
                outdetails=mylib.details maxlevels=5;
run;

data sp;
   set mylib.details( where=(_varname_='Species'));
   label _cfmt_='Formatted value of the variable Species';
   if _index_ = . then do;
      _cfmt_=cats(">",left(_cfmt_));
   end;
  _cfmt_=left(_cfmt_);
run;

proc sgplot data=sp;
  vbar _cfmt_ / freq=_freq_;
run;
```

Output 7.1.1 shows that the variable Species has only three levels in the details data.

Output 7.1.1 Histogram of Species



Whether MAXLEVELS=100 or 5, the Species variable remains fully visible with a _CARDINALITY_ value of 3.

Example 7.2: Limited Cardinality of the Sepal Length Variable

(View the complete code for this example (cardinality_ex2.sas) in the example repository.)

This example demonstrates how to use PROC CARDINALITY with more levels; it uses the same data table as is used in the section "Getting Started: CARDINALITY Procedure" on page 248.

You can load the sashelp.iris data set into your CAS session by naming your CAS engine libref in the first statement of the following DATA step:

```
data mylib.iris;
   set sashelp.iris;
run;
```

The following statements run PROC CARDINALITY and display some of the contents of the cardinality data set:

```
proc cardinality data=mylib.iris outcard=mylib.card
                 outdetails=mylib.details maxlevels=100;
run;


proc print data=mylib.card;
   var _varname_ _type_ _cardinality_ _more_ _visible_ _min_ _max_;
run;
```

Output 7.2.1 shows that all the variables have a _VISIBILITY_ value of 100%.

Output 7.2.1 Visibility of the Variables

Obs	_VARNAME_	_TYPE_	_CARDINALITY_	_MORE_	_VISIBLE_	_MIN_	_MAX_
1	Species	C	3	N	100	.	.
2	SepalLength	N	35	N	100	43	79
3	SepalWidth	N	23	N	100	20	44
4	PetalLength	N	43	N	100	10	69
5	PetalWidth	N	22	N	100	1	25

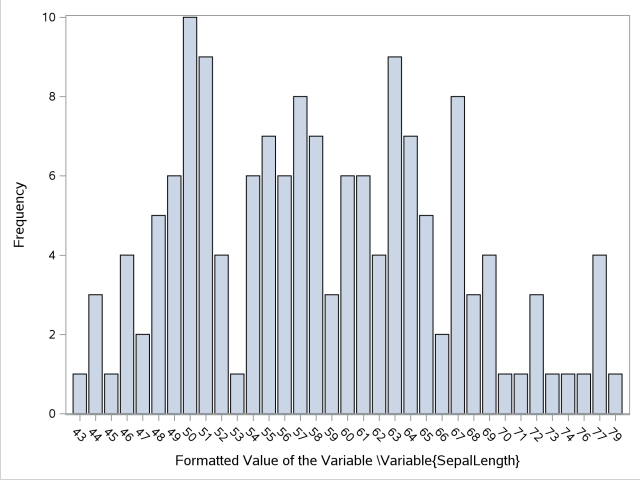
The following statements extract and display the levels of the variable SepalLength:

```
data sp;
   label _cfmt_='Formatted Value of the Variable \Variable{SepalLength}';
   set mylib.details( where=(_varname_='SepalLength'));
   if _index_ = . then do;
      _cfmt_=cats(">",left(_cfmt_));
   end;
   _cfmt_=left(_cfmt_);
run;

proc sgplot data=sp;
   vbar _cfmt_ / freq=_freq_;
run;
```

Output 7.2.2 shows the histogram.

Output 7.2.2 Full Histogram of the SepalLength Variable



The following statements specify MAXLEVELS=5, so only the leftmost five levels of the full histogram are displayed. All the other levels are combined into one level, which is reported in the details data table with a missing value in the _INDEX_ column.

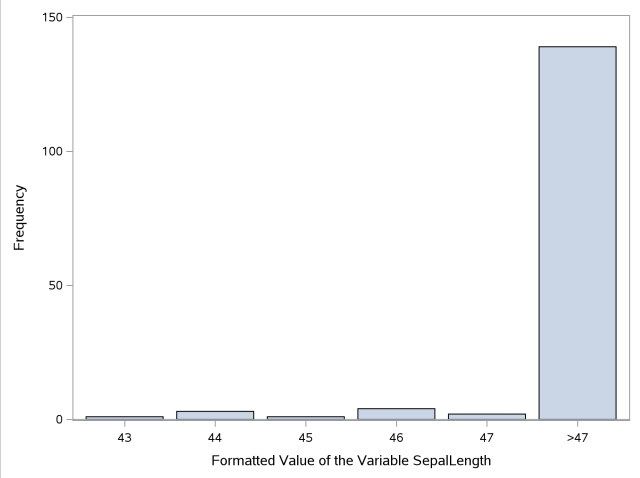
```
proc cardinality data=mylib.iris outcard=mylib.card
                 outdetails=mylib.details maxlevels=5;
run;

data sp;
   label _cfmt_='Formatted Value of the Variable SepalLength';
   set mylib.details( where=(_varname_='SepalLength'));
   if _index_ = . then do;
      _cfmt_=cats(">",left(_cfmt_));
   end;
   _cfmt_=left(_cfmt_);
run;

proc sgplot data=sp;
   vbar _cfmt_ / freq=_freq_;
run;
```

Output 7.2.3 shows the full histogram of the variable SepalLength. It shows all 35 levels of the variable.

Output 7.2.3 Limited Histogram of the SepalLength Variable



Example 7.3: More Levels of the Sepal Length Variable

(View the complete code for this example (cardinality_ex3.sas) in the example repository.)

You can load the sashelp.iris data set into your CAS session by naming your CAS engine libref in the first statement of the following DATA step:

```
data mylib.iris;
   set sashelp.iris;
run;
```

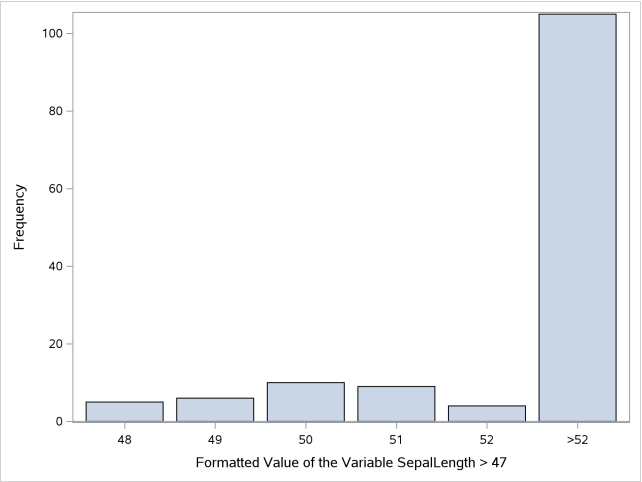
You can dig deeper by requesting a subsequent segment of the histogram of any variable. This example further examines the SepalLength variable. The cutoff reported from Example 7.2 was 47, which means that all levels less than or equal to 47 have been reported. So you might want to run PROC CARDINALITY again in order to report levels greater than 47, as shown in the following statements:

```
proc cardinality data=mylib.iris (where=(SepalLength > 47))
                 outcard=mylib.card outdetails=mylib.details maxlevels=5;
   var SepalLength;
run;

data sp;
   set mylib.details;
   label _cfmt_='Formatted Value of the Variable SepalLength > 47';
   if _index_ = . then do;
      _cfmt_=cats(">",left(_cfmt_));
   end;
  _cfmt_=left(_cfmt_);
run;

proc sgplot data=sp;
   vbar _cfmt_ / freq=_freq_;
run;
```

Output 7.3.1 Histogram for SepalLength Levels Greater Than 47



The union of the detailed levels from Example 7.2 and this example cover 10 levels of the SepalLength variable.

Table 7.4 lists the variables that are related to the maximum frequency in the histogram.

Table 7.4 More Variables from the Cardinality Data

Name	Description
MFREQ	Maximum frequency
MFREQFOUNDLEVEL	Maximum frequency found in the visible part of the report
MFREQNUM	Numeric level that has the maximum frequency
MFREQCHR	Character level that has the maximum frequency
MFREQFMT	Formatted level that has the maximum frequency

The maximum frequency can be attained in the visible part of the histogram, or it can be hidden with the others. If the level that has the maximum frequency is in the visible part, the _MFREQFOUNDLEVEL_ variable is set to Y (Yes); otherwise it is set to N (No). This information is reported here because it is useful for subsequent analysis.

The following statements illustrate the variables that pertain to the maximum frequency of each variable.

```
proc cardinality data=mylib.iris outcard=mylib.card
                outdetails=mylib.details maxlevels=5;
run;

proc print data=mylib.card;
   var _varname_ _mf:;
run;
```

Output 7.3.2 Maximum Frequency Information

Obs	_VARNAME_	_MFREQ_	_MFREQFOUNDLEVEL_	_MFREQNUM_	_MFREQCHR_	_MFREQCFMT_
1	Species	50	Y	.	Setosa	Setosa
2	SepalLength	139	N	.		
3	SepalWidth	131	N	.		
4	PetalLength	126	N	.		
5	PetalWidth	101	N	.		

For the Species variable, the value of _MFREQ_ is 50. Because the Species variable has a limited cardinality of 3, _MFREQFOUNDLEVEL_ = Y. Also note that both _MFREQCHR_ and its formatted value in _MFREQCFMT_ are Setosa.

For the SepalLength variable, _MFREQ_ is 139 (which occurs for the value _MFREQNUM_= .), and _MFREQFOUNDLEVEL_ = N (which means that the visible part of the histogram does not contain the level that has the maximum frequency).

Example 7.4: A Variable with a User-Defined Format

(View the complete code for this example (cardinality_ex4.sas) in the example repository.)

This example demonstrates how PROC CARDINALITY determines how to order variables that have user-defined formats. User-defined formats are considered to be prior knowledge about the variable and are used by default. (You can override this behavior by requesting a specific order in the ORDER= option.)

You can load the sashelp.cars data set into your CAS session by naming your CAS engine libref in the first statement of the following DATA step:

```
data mylib.cars;
   set sashelp.cars;
run;
```

The following statements create the format on the variable engineSize, run the procedure, and examine some of the variables in the cardinality data set:

```
proc format casfmtlib='myfmtlib';
   value engsize
       low - <3 ='toys'
       3   - <6 ='granny'
       6   -high='usable';
run;

data mylib.cars;
   format engineSize engsize.;
   set mylib.cars;
run;

proc cardinality data=mylib.cars outcard=mylib.card
                 outdetails=mylib.details maxlevels=5;
   var engineSize;
run;
```

```
title 'Cars data with a user-defined format';
proc print data=mylib.card;
   var _varname_ _order_ _more_ _cardinality_;
run;
```

Output 7.4.1 Summary of the Variable engineSize

Cars data with a user-defined format

Obs	_VARNAME_	_ORDER_	_MORE_	_CARDINALITY_
1	engineSize	ASCFMT	N	3

The order that is used for the engineSize variable is ascending formatted (ASCFMT), because engineSize has a user-defined format (engsize7). Three levels were found (_CARDINALITY_ = 3). There are no more levels to report (_MORE_ = N). The full histogram is displayed. The following statements produce Output 7.4.2:

```
proc print data=mylib.details;
   var _varname_ _index_ _freq_ _cfmt_;
run;
```

Output 7.4.2 Details Data with a User-Defined Format for the Variable engineSize

Cars data with a user-defined format

Obs	_VARNAME_	_INDEX_	_FREQ_	_CFMT_
1	engineSize	1	242	granny
2	engineSize	2	178	toys
3	engineSize	3	8	usable

The order that is used for the engineSize variable is ascending formatted (ASCFMT), because engineSize has a user-defined format engsize7. All the levels are reported. There is no missing values in the _INDEX_ column for the engineSize variable.

Example 7.5: Forcing Another Order on the engineSize Variable

(View the complete code for this example (cardinality_ex5.sas) in the example repository.)

You can load the sashelp.cars data set into your CAS session by naming your CAS engine libref in the first statement of the following DATA step:

```
data mylib.cars;
   set sashelp.cars;
run;
```

The following statements specify the ORDER= option in the VAR statement in order to use a different levelization order:

```
title 'Cars data with a user-defined format with an ASC order';
proc cardinality data=mylib.cars outcard=mylib.card
                outdetails=mylib.details maxlevels=5;
   var engineSize /order=asc;
run;

proc print data=mylib.card ;
   var _varname_ _order_ _more_ _cardinality_;
run;
```

Output 7.5.1 Summary of the engineSize Variable without a Format

Cars data with a user-defined format with an ASC order

Obs	_VARNAME_	_ORDER_	_MORE_	_CARDINALITY_
1	EngineSize	ASC	Y	5

The order that is used in the engineSize variable is ascending unformatted, as requested by the ORDER=ASC option in the VAR statement. The reported levels are based on the numeric raw values of the engineSize variable. The format engsize7. is ignored. The raw values are more than 5 (the value of the MAXLEVEL= option). The details data table contains six rows: five for reported known levels and one for all the remaining unreported levels). The cardinality data table reports _MORE_ = Y (Yes) and _CARDINALITY_ = 5 for the engineSize variable. The details of the variable engineSize are discussed next, starting with the details data set produced in the previous step:

```
data details(drop=_rawchar_);
   set mylib.details;
   if _index_ = . then do;
      _cfmt_ = cats(">",  put(_rawnum_,best12.));
      _rawnum_ = .;
   end;
     _cfmt_=left(_cfmt_);
run;

proc print data=details;
run;
```

Output 7.5.2 Details of the engineSize Variable without a Format

Cars data with a user-defined format with an ASC order

Obs	_VARNAME_	_INDEX_	_FREQ_	FREQPERCENT	NMISSPERCENT	_RAWNUM_	_FMTWIDTH_	_CFMT_
1	EngineSize	1	2	0.4672897196	0.4672897196	1.3	12	1.3
2	EngineSize	2	1	0.2336448598	0.2336448598	1.4	12	1.4
3	EngineSize	3	6	1.4018691589	1.4018691589	1.5	12	1.5
4	EngineSize	4	10	2.3364485981	2.3364485981	1.6	12	1.6
5	EngineSize	5	4	0.9345794393	0.9345794393	1.7	12	1.7
6	EngineSize	.	405	94.626168224	94.626168224	.	12	>1.7

The order that is used in the engineSize variable is ascending unformatted, as requested by the ORDER= ASC option in the VAR statement. The reported levels are based on the numeric raw value of the engineSize

variable in ascending order. The format engsize7. is ignored. Five raw values are displayed with one additional level, indicating that the histogram of the variable is partial and not full. The observation for which _INDEX_ = . indicates that there are still unreported levels in the raw values. A second pass with a WHERE clause can reveal the details of the levels.

References

Fisher, R. A. (1936). "The Use of Multiple Measurements in Taxonomic Problems." *Annals of Eugenics* 7:179–188.

Chapter 8
CATTRANSFORM Procedure

Contents

Overview: CATTRANSFORM Procedure **266**
PROC CATTRANSFORM Features 266
Using CAS Sessions and CAS Engine Librefs 266
Getting Started: CATTRANSFORM Procedure **267**
Syntax: CATTRANSFORM Procedure **269**
PROC CATTRANSFORM Statement 270
CODE Statement 271
FREQ Statement 271
INPUT Statement 272
METHOD Statement 272
OUTPUT Statement 274
SAVESTATE Statement 274
TARGET Statement 275
WEIGHT Statement 275
Details: CATTRANSFORM Procedure **275**
Unsupervised Grouping 276
Supervised Grouping 276
One-Hot Encoding 276
Scoring 276
Displayed Output 277
ODS Table Names 278
Examples: CATTRANSFORM Procedure **278**
Example 8.1: Binning by Grouping Rare Levels 279
Example 8.2: Classification Tree Binning 281
Example 8.3: Regression Tree Binning 283

Overview: CATTRANSFORM Procedure

The CATTRANSFORM procedure transforms categorical variables in SAS Viya. It supports binning and encoding methods that are commonly performed as part of the data preprocessing step in creating machine learning models.

Categorical variables that have high cardinality can slow the computing performance of machine learning algorithms and increase the risk of model overfitting. PROC CATTRANSFORM supports several binning methods, both supervised and unsupervised, that enable you to reduce cardinality. These methods include grouping rare levels, classification tree binning, regression tree binning, and weight-of-evidence binning. The procedure also supports one-hot encoding by transforming the values of categorical variables to numeric values that start with the number 1.

PROC CATTRANSFORM facilitates transformation scoring of new data by using either an analytic store or scoring code in a SAS DATA step.

PROC CATTRANSFORM Features

The CATTRANSFORM procedure has the following features:

- supports supervised and unsupervised binning methods
- labels data by using one-hot encoding
- places missing values into a separate bin
- computes evaluation statistics for comparing the predictive performance of transformed variables
- scores transformations by using an analytic store
- generates scoring code for a SAS DATA step

Using CAS Sessions and CAS Engine Librefs

SAS Cloud Analytic Services (CAS) is the analytic server and associated cloud services in SAS Viya. This section describes how to create a CAS session and set up a CAS engine libref that you can use to connect to the CAS session. It assumes that you have a CAS server already available; contact your system administrator if you need help starting and terminating a server. This CAS server is identified by specifying the host on which it runs and the port on which it listens for communications. To simplify your interactions with this CAS server, the host information and port information for the server are stored as SAS option values that are retrieved automatically whenever this CAS server needs to be accessed. You can examine the host and port values for the server at your site by using the following statements:

```
proc options option=(CASHOST CASPORT);
run;
```

In addition to starting a CAS server, your system administrator might also have created a CAS session and a CAS engine libref for your use. You can define your own sessions and CAS engine librefs that connect to the CAS server as shown in the following statements:

```
cas mysess;
libname mylib cas sessref=mysess;
```

The CAS statement creates the CAS session named mysess, and the LIBNAME statement creates the mylib CAS engine libref that you use to connect to this session. It is not necessary to explicitly name the CASHOST and CASPORT of the CAS server in the CAS statement, because these values are retrieved from the corresponding SAS option values.

If you have created the mysess session, you can terminate it by using the TERMINATE option in the CAS statement as follows:

```
cas mysess terminate;
```

For more information about the CAS and LIBNAME statements, see the section "Introduction to Shared Concepts" on page 6 in Chapter 2, "Shared Concepts."

Getting Started: CATTRANSFORM Procedure

NOTE: Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections "Using CAS Sessions and CAS Engine Librefs" on page 7 and "Loading a SAS Data Set onto a CAS Server" on page 7 in Chapter 2, "Shared Concepts."

This example demonstrates the basic features of the CATTRANSFORM procedure.

The following DATA step creates the data table mylib.catdata, which consists of 1,000 observations and includes two nominal variables:

```
data mylib.catdata;
    call streaminit(12345);
    do myid=1 to 1000;
        y=rand('table',.1,.9);
        c1=rand('table',.35,.25,.2,.1,.04,.03,.02,.01);
        c2=rand('table',.01,.02,.03,.04,.1,.2,.25,.35);
        output;
    end;
run;
```

These statements assume that your CAS engine libref is named mylib, as in the section "Using CAS Sessions and CAS Engine Librefs" on page 266, but you can substitute any appropriately defined CAS engine libref.

The following statements execute PROC CATTRANSFORM and use the default transformation method to group rare levels:

```
proc cattransform data=mylib.catdata;
    input c1 c2;
    output out=mylib.out;
run;
```

The INPUT statement defines the categorical input variables c1 and c2, and the OUTPUT statement puts the transformed variables into the output data table mylib.out.

The results are shown in Figure 8.1 through Figure 8.4, which display the following tables: "Transformation Requests," "Variable Transformation Information," "Nominal Variable Information," and "Bin Details." The "Bin Details" table shows that four levels of both variables were combined into a bin to meet the default rare threshold percentage of 5%.

Figure 8.1 Transformation Requests

The CATTRANSFORM Procedure

Transformation Requests

Transformation Name	Transformation Variables	N Nominal Transform Method
CAT	2	Group Rare

Figure 8.2 Variable Transformation Information

Variable Transformation Information

Variable	Result Variable	N	N Miss	N Bins
c1	CAT_c1	1000	0	5
c2	CAT_c2	1000	0	5

Figure 8.3 Nominal Variable Information

Nominal Variable Information

Variable	N	N Miss	N Levels
c1	1000	0	8
c2	1000	0	8

Figure 8.4 Bin Details

Bin Details

Variable	Bin ID	N Levels	N
c1	1	1	363
c1	2	1	212
c1	3	1	211
c1	4	1	110
c1	5	4	104
c2	1	1	369
c2	2	1	248
c2	3	1	189
c2	4	1	96
c2	5	4	98

Syntax: CATTRANSFORM Procedure

The following statements are available in the CATTRANSFORM procedure:

PROC CATTRANSFORM < *options* > **;**
CODE FILE=*filename* **;**
FREQ *variable* **;**
INPUT *variables* **;**
METHOD *method*(< *options* >) **;**
OUTPUT OUT=*libref.data-table* < *option* > **;**
SAVESTATE RSTORE=*libref.data-table* **;**
TARGET *variable* / **LEVEL=INTERVAL** | **NOMINAL** < *option* > **;**
WEIGHT *variable* **;**

The PROC CATTRANSFORM and INPUT statements are required. The TARGET statement is required for the supervised binning methods and optional for the unsupervised binning methods.

The following sections describe the PROC CATTRANSFORM statement and then describe the other statements in alphabetical order.

PROC CATTRANSFORM Statement

PROC CATTRANSFORM < *options* > **;**

The PROC CATTRANSFORM statement invokes the procedure.

You can specify the following *options*:

BINMISSING
places missing values into a separate bin whose bin ID value is 0.

DATA=*libref.data-table*
names the input data table for PROC CATTRANSFORM to use. The default is the most recently created data table. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to the data, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 266.

data-table specifies the name of the input data table.

DISTINCTCOUNTLIMIT=*integer*
specifies the maximum level limit.

By default, DISTINCTCOUNTLIMIT=5000.

EVALUATIONSTATS(*options***)**
requests evaluation statistics about the transformed variables. You can specify the following *options*:

WOEADJUST=*number*
specifies the weight-of-evidence adjustment value. The *number* can range from 0 to 1, inclusive.

By default, WOEADJUST=0.5.

WOEDEFINITION=EVENT | NONEVENT
specifies the weight-of-evidence definition. If WOEDEFINITION=EVENT, the weight of evidence is defined as event/nonevent. If WOEDEFINITION=NONEVENT, the weight of evidence is defined as nonevent/event.

By default, WOEDEFINITION=NONEVENT.

MISSINGEVALNONEVENT
treats missing targets as nonevents.

NBINS=*integer*
specifies the number of bins for all input variables.

OUTLEVELBINMAP=*libref.data-table*
saves the bin mapping information to a data table. The bin mapping table contains three columns: Variable, Level, and BinID. *libref.data-table* is a two-level name, where *libref* refers to the library, and *data-table* specifies the name of the output data table. For more information about this two-level name, see the DATA= option and the section "Using CAS Sessions and CAS Engine Librefs" on page 266.

PREFIX="*string***"**
specifies the prefix for the transformed variables.

By default, PREFIX="CAT".

PREPROCESSRARE
groups levels whose frequency percentages are 5% or less with the next least frequent level until all percentages are greater than 5% before implementing the specified method.

CODE Statement

CODE FILE=*filename* **;**

The CODE statement generates scoring code and saves it in a file that you can include in a SAS DATA step.

You must specify the following option:

FILE=*filename*
specifies the name of the file to write the SAS scoring code to.

FREQ Statement

FREQ *variable* **;**

The *variable* in the FREQ statement identifies a numeric variable in the input data table that contains the frequency of occurrence of each observation. PROC CATTRANSFORM treats each observation as if it appears f times, where f is the value of the FREQ *variable* for the observation. If f is not an integer, it is truncated to an integer. If f is less than 1 or missing, the observation is not used in the analysis. When you omit the FREQ statement, each observation is assigned a frequency of 1.

INPUT Statement

INPUT *variables* **;**

The INPUT statement specifies one or more *variables* as input for binning or encoding. All input variables are assumed to be nominal.

METHOD Statement

METHOD *method(< options >)* **;**

The METHOD statement specifies the binning or encoding method and any associated options.

You must specify one of the following *methods*:

GROUPRARE(*options***)**
groups levels that have a rare frequency or percentage with the next least frequent level until a threshold has been satisfied for all levels. By default, the *method* is GROUPRARE.

You can specify the following *options*:

RARETHRESHOLD=*integer*
specifies the rare frequency threshold where levels whose frequencies are less than or equal to the threshold are grouped with the next least-frequent level. The *integer* must be greater than 0.

RARETHRESHOLDPERCENT=*number*
specifies the rare threshold percentage where levels whose percentages are less than or equal to the threshold are grouped with the next least-frequent level. The *number* can range between 0 and 100.

By default, RARETHRESHOLDPERCENT=5.0.

ONEHOT
uses one-hot encoding of the input variables.

TREE(*options***)**
uses the supervised tree-based binning method. You can specify the following *options*:

CRITERIA=GAIN | GAINRATIO | GINI | SSE
specifies the tree criteria. You can specify the following criteria:

GAIN	uses the information gain as the tree-growing criterion for a nominal target.
GAINRATIO	uses the gain ratio as the tree-growing criterion for a nominal target.
GINI	uses the Gini statistic as the tree-growing criterion for a nominal target.
SSE	uses the residual sum of squares as the tree-growing criterion for an interval target.

MAXNBINS=*integer*
specifies the maximum number of bins. The *integer* must be greater than or equal to 2. The default for each *integer* is computed as the binary logarithm (log base 2) of the number of distinct values.

MINNBINS=*integer*
specifies the minimum number of bins. The *integer* must be greater than or equal to 1.

MINNOBSINBIN=*integer*
specifies the minimum number of observations to include in a bin.

MINPERNOBSINBIN=*number*
specifies the minimum percentage of observations to include in a bin. The *number* can be between 0 and 50.

By default, MINPERNOBSINBIN=5.0.

NOTE: For the tree-based method, a valid target is required.

WOE(*options***)**
uses the supervised weight-of-evidence (WOE) binning method.

You can specify the following *options*:

MAXNBINS=*integer*
specifies the maximum number of bins. The *integer* must be greater than or equal to 2. The default for each *integer* is computed as the binary logarithm (log base 2) of the number of distinct values.

MINNBINS=*integer*
specifies the minimum number of bins. The *integer* can be greater than or equal to 1.

MINNOBSINBIN=*integer*
specifies the minimum number of observations to include in a bin.

MINPERNOBSINBIN=*number*
specifies the minimum percentage of observations to include in a bin. The *number* can be between 0 and 50.

By default, MINPERNOBSINBIN=5.0.

NOTE: For the WOE method, a nominal target and a target event are required.

OUTPUT Statement

OUTPUT OUT=*libref.data-table* < *option*> **;**

The OUTPUT statement creates an output data table that contains the transformed variables.

You must specify the following option:

OUT=*libref.data-table*
names the output data table for PROC CATTRANSFORM to use. You must specify this option before any other options. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to where the data table is to be stored, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 266.

data-table specifies the name of the output data table.

You can also specify the following *option*:

COPYVAR=*variable*
COPYVARS=(*variables***)**
copies one or more *variables* from the input data table to the output data table.

SAVESTATE Statement

SAVESTATE RSTORE=*libref.data-table* **;**

The SAVESTATE statement creates an analytic store for the model and saves it as a binary object in a data table. You can use the analytic store in the ASTORE procedure to score new data. For more information, see Chapter 4, "ASTORE Procedure" (*SAS Viya: Machine Learning Procedures*).

You must specify the following option:

RSTORE=*libref.data-table*
specifies a data table in which to save the analytic store for the model. *libref.data-table* is a two-level name, where *libref* refers to the library, and *data-table* specifies the name of the output data table. For more information about this two-level name, see the DATA= option and the section "Using CAS Sessions and CAS Engine Librefs" on page 266.

TARGET Statement

TARGET *variable* / **LEVEL=INTERVAL | NOMINAL** < *option* > **;**

The TARGET statement names the target *variable* to use in the supervised tree-based and weight-of-evidence binning methods. You can also use the target *variable* to compute evaluation statistics. The target *variable* can be nominal or interval, and it cannot be listed in the INPUT statement.

If you specify LEVEL=NOMINAL, you can also specify the following *option*:

EVENT="*string***"**
specifies the event of a nominal target for calculating binary target statistics, which are Gini, weight of evidence, and information value. This option is required for the weight-of-evidence binning method and is optional for other methods. If you specify an event along with a multinomial target, PROC CATTRANSFORM computes the binary target statistics by combining the nonevent levels.

WEIGHT Statement

WEIGHT *variable* **;**

The *variable* in the WEIGHT statement is used as a weight to perform a weighted analysis of the data. Observations that have nonpositive or missing weights are not included in the analysis. The values of the WEIGHT variable can be nonintegral and are not truncated. When you omit the WEIGHT statement, all observations that are used in the analysis are assigned a weight of 1.

Details: CATTRANSFORM Procedure

PROC CATTRANSFORM transforms categorical variables in SAS Viya. It supports supervised and unsupervised binning methods for categorical input variables. The supervised methods include tree-based and weight-of-evidence-based binning. For the tree-based method, the target can be nominal or interval. For the weight-of-evidence-based method, the target must be nominal. For unsupervised binning, PROC CATTRANSFORM supports the grouping of rare levels. It can also encode categorical input variables with one-hot encoding.

For each input variable, if the target is specified, the evaluation statistics of transformed variable can be computed. The statistics include chi-square, G2, Gini, weight of evidence, information value, *F* test, residual sum of squares, and relative variable importance.

Unsupervised Grouping

PROC CATTRANSFORM provides one unsupervised binning method that you specify by using the GROUPRARE option in the METHOD statement. It groups levels that have a rare frequency or percentage with the next least-frequent level until a threshold has been satisfied for all levels. You can specify the threshold as a frequency or a percentage. By default, the threshold is 5.0%.

Supervised Grouping

The supervised binning methods in PROC CATTRANSFORM, the tree-based and weight-of-evidence-based methods, are specified by the TREE and WOE options, respectively, in the METHOD statement. For the tree-based method, the target can be nominal or interval. If the target is nominal, the classification tree algorithm is used. If the target is interval, the regression tree algorithm is used. For the weight-of-evidence method, the target must be nominal, and the target event must be specified.

One-Hot Encoding

The one-hot encoding method in PROC CATTRANSFORM, which is specified by the ONEHOT option in the METHOD statement, transforms the values of a categorical variables to numeric values that start with the number 1. The order of the values in the transformed variable corresponds to the sorted order of the original values. The transformation is useful for machine learning because it converts nominal variables to variables that have only numeric values. The encoded variables are also sequentially numbered like binned variables.

One-hot encoding is known for transforming each value of a categorical variable to a zero-one variable, and the resulting set of zero-one variables is used as input to machine learning algorithms. Although the transformed variable from the one-hot method could be further processed into zero-one variables, the zero-one variables are not necessary in SAS modeling procedures. Because the CLASS statement internally creates the zero-one variables from any nominal variable, you can choose to specify either the original variable or the transformed variable when you use SAS procedures for modeling.

Scoring

PROC CATTRANSFORM facilitates transformation scoring of new data by using either an analytic store or scoring code in a SAS DATA step.

When you specify the SAVESTATE statement with the RSTORE= option, an analytic store is created for the scoring model. You can then score new data from the model by using the ASTORE procedure. For more information, see Chapter 4, “ASTORE Procedure” (*SAS Viya: Machine Learning Procedures*).

Alternatively, you can specify the CODE statement with the FILE= option to save scoring code in a file. In this case, you score new data by using the SAS DATA step.

Displayed Output

PROC CATTRANSFORM produces output that is organized into various tables, which the following sections describe in the order of their appearance.

Transformation Requests

The "Transformation Requests" table displays the transformation name, number of transformation variables, and transformation method.

If you specify the EVALUATIONSTATS option in the PROC CATTRANSFORM statement and the EVENT= option in the TARGET statement, the table also displays the weight-of-evidence definition, weight-of-evidence adjustment value, and information value factor.

Variable Transformation Information

The "Variable Transformation Information" table displays the variable name, target variable, result variable, number of observations, number of missing observations, and number of bins.

If you specify the EVALUATIONSTATS option in the PROC CATTRANSFORM statement with an interval target, the table also displays the *F*-test statistic.

If you specify the EVALUATIONSTATS option with a nominal target, the table also displays the chi-square and G2 statistics.

If you specify the EVALUATIONSTATS option together with the TREE option in the METHOD statement, the table also displays the residual sum of squares and relative variable importance.

If you specify the EVALUATIONSTATS option and the EVENT= option in the TARGET statement with a nominal target, the table also displays the information value and weight-of-evidence statistics.

Nominal Variable Information

The "Nominal Variable Information" table displays the variable name, number of observations, number of missing observations, and number of levels for the inputs and nominal target.

Bin Details

For each bin, the "Bin Details" table displays the variable name, bin ID, number of levels, and number of observations.

If you specify the EVALUATIONSTATS option in the PROC CATTRANSFORM statement with an interval target, the table also displays the mean, standard deviation, and number of missing values of the target for each bin.

If you specify the EVALUATIONSTATS option and the EVENT= option in the TARGET statement with a nominal target, the table also displays the event count, nonevent count, weight of evidence, and information value for each bin.

OutputCasTables Table

The OutputCasTables table is a special table that has information about each CAS table that is created during a CAS action execution. The information for each CAS table consists of the CAS table name, the caslib in which the table resides, and the number of columns and rows in the CAS table.

ODS Table Names

Every table that PROC CATTRANSFORM creates has a name associated with it. You must use this name to refer to the table when you use ODS statements. The name of each table and a short description of the contents are listed in Table 8.1.

Table 8.1 ODS Tables Produced by PROC CATTRANSFORM

Table Name	Description	Statement	Option
BinDetails	Bin details	PROC CATTRANSFORM	Default
NomVarInfo	Nominal variable information	PROC CATTRANSFORM	Default
OutputCasTables	See the section "OutputCasTables Table" on page 278	PROC CATTRANSFORM, OUTPUT, or SAVESTATE	OUTLEVELBINMAP=
TransInfo	Transformation requests	PROC CATTRANSFORM	Default
VarTransInfo	Variable transformation information	PROC CATTRANSFORM	Default

Examples: CATTRANSFORM Procedure

NOTE: Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections "Using CAS Sessions and CAS Engine Librefs" on page 7 and "Loading a SAS Data Set onto a CAS Server" on page 7 in Chapter 2, "Shared Concepts."

Example 8.1: Binning by Grouping Rare Levels

This example shows how to use the CATTRANSFORM procedure to reduce the number of levels of nominal variables by using the binning method of grouping rare levels. The input data come from the Hmeq data set, which is in the Sampsio library that SAS provides. It contains 5,960 mortgage applicants. A target variable named Bad indicates whether the customer has paid on the loan or defaulted on it. Table 8.2 describes the variables in Hmeq.

Table 8.2 Variables in the Home Equity (Hmeq) Data Set

Variable	Role	Level	Description
Bad	Response	Binary	1 = customer defaulted on the loan or is seriously delinquent 0 = customer is current on loan payments
CLAge	Predictor	Num	Age of oldest credit line in months
CLNo	Predictor	Num	Number of credit lines
DebtInc	Predictor	Num	Debt-to-income ratio
Delinq	Predictor	Num	Number of delinquent credit lines
Derog	Predictor	Num	Number of major derogatory reports
Job	Predictor	Char	Occupational category
Loan	Predictor	Num	Requested loan amount
MortDue	Predictor	Num	Amount due on existing mortgage
nInq	Predictor	Num	Number of recent credit inquiries
Reason	Predictor	Binary	`'DebtCon'` = debt consolidation `'HomeImp'` = home improvement
Value	Predictor	Num	Value of current property
YoJ	Predictor	Num	Years at present job

You can load the Hmeq data set into your CAS session by specifying your CAS engine libref in the first statement in the following DATA step. These statements assume that your CAS engine libref is named mylib, but you can substitute any appropriately defined CAS engine libref.

```
data mylib.hmeq;
   set sampsio.hmeq;
run;
```

The following statements call PROC CATTRANSFORM to bin the selected variables by using the default GROUPRARE option in the METHOD statement. Also, the BINMISSING option places the missing values into a separate bin.

```
proc cattransform data=mylib.hmeq binmissing;
    input reason job derog delinq ninq;
    output out=mylib.score;
run;
```

PROC CATTRANSFORM generates the results tables shown in Output 8.1.1 through Output 8.1.4.

The "Transformation Requests" table in Output 8.1.1 shows that the number of transformation variables is five and the transformation method is "Group Rare."

Output 8.1.1 Transformation Requests

The CATTRANSFORM Procedure

Transformation Requests

Transformation Name	N Transformation Variables	Nominal Transform Method
CAT	5	Group Rare

The "Variable Transformation Information" table in Output 8.1.2 shows the variable name, result variable, number of observations, number of missing observations, and number of bins for each variable.

Output 8.1.2 Variable Transformation Information

Variable Transformation Information

Variable	Result Variable	N	N Miss	N Bins
DELINQ	CAT_DELINQ	5380	580	4
DEROG	CAT_DEROG	5252	708	3
NINQ	CAT_NINQ	5450	510	6
JOB	CAT_JOB	5681	279	6
REASON	CAT_REASON	5708	252	3

The "Nominal Variable Information" table in Output 8.1.3 shows the number of levels of each input variable.

Output 8.1.3 Nominal Variable Information

Nominal Variable Information

Variable	N	N Miss	N Levels
DELINQ	5380	580	14
DEROG	5252	708	11
JOB	5681	279	6
NINQ	5450	510	16
REASON	5708	252	2

The "Bin Details" table in Output 8.1.4 displays the binning details for each input variable. The first step of the method of grouping rare levels encodes the values to ordered numerical values from most frequent to least frequent. The second step groups levels that have a rare frequency with the next least-frequent level until the threshold has been met. In this case, the default threshold is 5%. In addition, the BINMISSING option places the missing values into a bin and assigns a value of 0 to them.

Output 8.1.4 Bin Details

Bin Details

Variable	Bin ID	N Levels	N
DELINQ	0	1	580
DELINQ	1	1	4179
DELINQ	2	1	654
DELINQ	3	12	547
DEROG	0	1	708
DEROG	1	1	4527
DEROG	2	10	725
NINQ	0	1	510
NINQ	1	1	2531
NINQ	2	1	1339
NINQ	3	1	780
NINQ	4	1	392
NINQ	5	12	408
JOB	0	1	279
JOB	1	1	2388
JOB	2	1	1276
JOB	3	1	948
JOB	4	1	767
JOB	5	2	302
REASON	0	1	252
REASON	1	1	3928
REASON	2	1	1780

The OUTPUT statement generates the "OutputCasTables" table, which is not shown. It displays the scored table that contains the transformed variables.

Example 8.2: Classification Tree Binning

This example shows how to use the CATTRANSFORM procedure to reduce the number of levels of nominal variables by using the classification tree binning method.

The following DATA step creates the data table mylib.catdata, which consists of 1,000 observations that contain two nominal predictor variables and a nominal target:

```
data mylib.catdata;
    call streaminit(12345);
    do myid=1 to 1000;
        y=rand('table',.1,.9);
        c1=rand('table',.35,.25,.2,.1,.04,.03,.02,.01);
        c2=rand('table',.01,.02,.03,.04,.1,.2,.25,.35);
        output;
    end;
run;
```

These statements assume that your CAS engine libref is named mylib, but you can substitute any defined CAS engine libref.

The following statements use PROC CATTRANSFORM to perform supervised binning by using the classification tree algorithm:

```
proc cattransform data=mylib.catdata evaluationstats;
    input c1 c2;
    target y / level=nominal event='1';
    method tree;
    output out=mylib.score;
    savestate rstore=mylib.classmodel;
run;
```

The statements generate the results shown in Output 8.2.1 through Output 8.2.4, which display the following tables: "Transformation Requests," "Variable Transformation Information," "Nominal Variable Information," and "Bin Details."

Specifying the TREE binning method option with the EVENT and EVALUATIONSTATS options adds columns to the tables. The "Transformation Requests" table displays the WOE definition, WOE adjustment, and information value (IV) factor. The "Variable Transformation Information" table displays evaluation statistics for comparing the predictive performance of the transformed variables. The "Nominal Variable Information" table displays the number of observations, number of missing observations, and number of nominal variable levels. The "Bin Details" table has event counts, weight-of-evidence, and information value for each bin.

The OUTPUT and SAVESTATE statements also generate the "OutputCasTables" results table, which is not shown. It displays the scored table from the OUTPUT statement and the scoring model from the SAVESTATE statement.

Output 8.2.1 Transformation Requests

The CATTRANSFORM Procedure

Transformation Requests

Transformation Name	Transformation Variables	Nominal Transform Method	WOE Definition	WOE Adjustment	IV Factor
CAT	2	DTree	Non-event by event	0.5	1

Output 8.2.2 Variable Transformation Information

Variable Transformation Information

Variable	Target Variable	Result Variable	N	N Miss	N Bins	Information Value	Weight of Evidence	Gini Index	Chi-Square DF	Chi-Square	Pr > ChiSq	G2 DF
c1	y	CAT_c1	1000	0	3	0.085429	0.213135	0.155564	2	7.258582	0.0265	2
c2	y	CAT_c2	1000	0	3	0.01236	0.294732	0.053339	2	1.026574	0.5985	2

Variable Transformation Information

Variable	G2	Pr > G2	RSS (DeltaD)	Relative Variable Importance
c1	7.36112	0.0252	1.271573	1
c2	1.061916	0.5880	0.179837	0.141429

Output 8.2.3 Nominal Variable Information

Nominal Variable Information

Variable	N	N Miss	N Levels
c1	1000	0	8
c2	1000	0	8
y	1000	0	2

Output 8.2.4 Bin Details

Bin Details

Variable	Bin ID	N Levels	N	Event Count	Non-Event Count	Weight of Evidence	Information Value
c1	1	3	343	44	299	-0.31476	0.038555
c1	2	2	404	37	367	0.063432	0.001585
c1	3	3	253	16	237	0.46446	0.045289
c2	1	3	575	60	515	-0.08119	0.003916
c2	2	2	344	31	313	0.081204	0.002195
c2	3	3	81	6	75	0.294717	0.006248

Example 8.3: Regression Tree Binning

This example shows how to use the CATTRANSFORM procedure to reduce the number of levels of nominal variables by using the regression tree binning method.

The following DATA step creates the data table mylib.regdata, which consists of 1,000 observations that contain two nominal predictor variables and an interval target:

```
data mylib.regdata;
    call streaminit(12345);
    do myid=1 to 1000;
        y=round(rand('normal',50,10),.1);
        c1=rand('table',.35,.25,.2,.1,.04,.03,.02,.01);
        c2=rand('table',.01,.02,.03,.04,.1,.2,.25,.35);
        output;
    end;
run;
```

These statements assume that your CAS engine libref is named mylib, but you can substitute any defined CAS engine libref.

The following statements use PROC CATTRANSFORM to perform supervised binning by using the regression tree algorithm:

```
proc cattransform data=mylib.regdata evaluationstats;
    input c1 c2;
    target y / level=interval;
    method tree;
    output out=mylib.score;
```

```
    savestate rstore=mylib.regmodel;
run;
```

The statements generate the results shown in Output 8.3.1 through Output 8.3.4, which display the following tables: "Transformation Requests," "Variable Transformation Information," "Nominal Variable Information," and "Bin Details."

Specifying the TREE method option with an interval target and the EVALUATIONSTATS option adds columns to the tables. The "Transformation Requests" table shows that the target is interval by denoting the regression tree binning method. The "Variable Transformation Information" table displays evaluation statistics for comparing the predictive performance of the transformed variables. The "Nominal Variable Information" table displays the number of observations, number of missing observations, and number of input variable levels. The "Bin Details" table displays descriptive statistics of the target variable for each bin.

The OUTPUT and SAVESTATE statements also generate the "OutputCasTables" results table, which is not shown. It displays the scored table from the OUTPUT statement and the scoring model from the SAVESTATE statement.

Output 8.3.1 Transformation Requests

The CATTRANSFORM Procedure

Transformation Requests

Transformation Name	N Transformation Variables	Nominal Transform Method
CAT	2	RTree

Output 8.3.2 Variable Transformation Information

Variable Transformation Information

Variable	Target Variable	Result Variable	N	N Miss	N Bins	Num DF	Den DF	F Value	Pr > F	RSS (DeltaD)	Relative Variable Importance
c1	y	CAT_c1	1000	0	3	2	997	2.10	0.1228	416.8862	1
c2	y	CAT_c2	1000	0	3	2	997	0.71	0.4935	140.6059	0.337276

Output 8.3.3 Nominal Variable Information

Nominal Variable Information

Variable	N	N Miss	N Levels
c1	1000	0	8
c2	1000	0	8

Output 8.3.4 Bin Details

Bin Details

Variable	Bin ID	N Levels	N	Mean Target Variable	Std Target Variable	N Miss Target Variable
c1	1	3	252	50.71905	9.201859	0
c1	2	3	514	49.86498	10.23368	0
c1	3	2	234	48.86624	10.13163	0
c2	1	2	302	50.31325	10.43932	0
c2	2	1	362	49.89282	9.884384	0
c2	3	5	336	49.37708	9.635069	0

Chapter 9
CORRELATION Procedure

Contents

Overview: CORRELATION Procedure **288**
PROC CORRELATION Features 288
PROC CORRELATION Compared with Other SAS Procedures 289
Using CAS Sessions and CAS Engine Librefs 290
Getting Started: CORRELATION Procedure **291**
Syntax: CORRELATION Procedure **293**
PROC CORRELATION Statement 293
BY Statement 296
DISPLAY Statement 296
DISPLAYOUT Statement 297
FREQ Statement 298
VAR Statement 298
WEIGHT Statement 298
WITH Statement 299
Details: CORRELATION Procedure **299**
Pearson Product-Moment Correlation 299
Probability Values 300
Cronbach's Coefficient Alpha 300
Missing Values 302
OUTP= Data Table 302
ODS Table Names 303
Examples: CORRELATION Procedure **304**
Example 9.1: Computing Correlations between Two Sets of Variables 304
Example 9.2: Computing Cronbach's Coefficient Alpha 307
Example 9.3: Saving Correlations in an Output Data Table 309
References **310**

Overview: CORRELATION Procedure

The CORRELATION procedure computes Pearson correlation coefficients and the probabilities that are associated with these statistics in SAS Viya.

The Pearson product-moment correlation is a parametric measure of a linear relationship between two variables. When only one set of analysis variables is specified, the default correlation analysis includes descriptive statistics for each analysis variable and pairwise Pearson correlation statistics for these variables. When two sets of analysis variables are specified, the default correlation analysis includes descriptive statistics for each analysis variable and pairwise Pearson correlation statistics between the two sets of variables. When the relationship between two variables is nonlinear or when outliers are present, the correlation coefficient might incorrectly estimate the strength of the relationship.

PROC CORRELATION Features

The CORRELATION procedure offers the following functionality:

- provides a WITH statement for analysis of pairwise Pearson correlation statistics between two sets of variables
- provides a FREQ statement for grouped analysis
- provides a WEIGHT statement for weighted analysis
- produces output data tables that contain descriptive statistics for analysis variables, Pearson correlation statistics, covariances, sums of squares, and corrected sums of squares
- computes Cronbach's coefficient alpha for estimating reliability

Because the CORRELATION procedure runs on CAS, it also does the following:

- enables you to run on a cluster of machines that distribute the data and the computations
- enables you to run in single-machine mode
- exploits all the available cores and concurrent threads (for information about how PROC CORRELATION uses threads, see the section "Multithreading" on page 102 in Chapter 2, "Shared Concepts")

PROC CORRELATION Compared with Other SAS Procedures

The CORRELATION procedure provides functionality that is comparable to that of the HPCORR and CORR procedures in Base SAS software.

The functionality of the CORRELATION procedure closely resembles that of the HPCORR procedure, which is a high-performance procedure. The CORRELATION procedure is the next generation of the HPCORR procedure, and it was developed specifically for SAS Viya. Both procedures are designed to run on a cluster of machines that distribute the data and the computations.

The CORRELATION procedure, the HPCORR procedure, and the CORR procedure have the similarities and differences shown in Table 9.1.

Table 9.1 Comparison of PROC CORRELATION, PROC HPCORR, and PROC CORR

Feature	PROC CORRELATION	PROC HPCORR	PROC CORR
Supports the computation of Pearson correlation coefficients and the probabilities associated with these statistics	Yes	Yes	Yes
Supports BY statement	Yes	No	Yes
Supports DISPLAY and DISPLAYOUT statements	Yes	No	No
Supports the computation of nonparametric measure of association	No	No	Yes
Supports OUTP statements	Yes	Yes	Yes
Supports Cronbach's alpha coefficient alpha for estimating reliability	Yes	No	Yes
Threading	Executes on multiple threads	Executes on multiple threads	Executes on a single thread

Using CAS Sessions and CAS Engine Librefs

SAS Cloud Analytic Services (CAS) is the analytic server and associated cloud services in SAS Viya. This section describes how to create a CAS session and set up a CAS engine libref that you can use to connect to the CAS session. It assumes that you have a CAS server already available; contact your system administrator if you need help starting and terminating a server. This CAS server is identified by specifying the host on which it runs and the port on which it listens for communications. To simplify your interactions with this CAS server, the host information and port information for the server are stored as SAS option values that are retrieved automatically whenever this CAS server needs to be accessed. You can examine the host and port values for the server at your site by using the following statements:

```
proc options option=(CASHOST CASPORT);
run;
```

In addition to starting a CAS server, your system administrator might also have created a CAS session and a CAS engine libref for your use. You can define your own sessions and CAS engine librefs that connect to the CAS server as shown in the following statements:

```
cas mysess;
libname mylib cas sessref=mysess;
```

The CAS statement creates the CAS session named mysess, and the LIBNAME statement creates the mylib CAS engine libref that you use to connect to this session. It is not necessary to explicitly name the CASHOST and CASPORT of the CAS server in the CAS statement, because these values are retrieved from the corresponding SAS option values.

If you have created the mysess session, you can terminate it by using the TERMINATE option in the CAS statement as follows:

```
cas mysess terminate;
```

For more information about the CAS and LIBNAME statements, see the section “Introduction to Shared Concepts” on page 6 in Chapter 2, “Shared Concepts.”

Getting Started: CORRELATION Procedure

(View the complete code for this example at https://github.com/sascommunities/doc-supplement-casstat/tree/main/Examples/cascorrgs.sas.)

This example creates a data table and then uses PROC CORRELATION to produce Pearson correlations.

The following statements create the mylib.Fitness data table. These data have been altered to contain some missing values:

```
*----------------- Data on Physical Fitness -----------------*
| These measurements were made on men involved in a physical |
| fitness course at N.C. State University.                   |
| The variables are Age (years), Weight (kg),                |
| Runtime (time to run 1.5 miles in minutes), and            |
| Oxygen (oxygen intake, ml per kg body weight per minute)   |
| Certain values were changed to missing for the analysis.   |
*------------------------------------------------------------*;
data mylib.Fitness;
   input Age Weight Oxygen RunTime @@;
   datalines;
44 89.47 44.609 11.37    40 75.07 45.313 10.07
44 85.84 54.297  8.65    42 68.15 59.571  8.17
38 89.02 49.874   .      47 77.45 44.811 11.63
40 75.98 45.681 11.95    43 81.19 49.091 10.85
44 81.42 39.442 13.08    38 81.87 60.055  8.63
44 73.03 50.541 10.13    45 87.66 37.388 14.03
45 66.45 44.754 11.12    47 79.15 47.273 10.60
54 83.12 51.855 10.33    49 81.42 49.156  8.95
51 69.63 40.836 10.95    51 77.91 46.672 10.00
48 91.63 46.774 10.25    49 73.37   .    10.08
57 73.37 39.407 12.63    54 79.38 46.080 11.17
52 76.32 45.441  9.63    50 70.87 54.625  8.92
51 67.25 45.118 11.08    54 91.63 39.203 12.88
51 73.71 45.790 10.47    57 59.08 50.545  9.93
49 76.32   .      .      48 61.24 47.920 11.50
52 82.78 47.467 10.50
;
```

These statements assume that your CAS engine libref is named mylib, but you can substitute any appropriately defined CAS engine libref.

The following statements invoke the CORRELATION procedure and request a correlation analysis:

```
proc correlation data=mylib.Fitness;
run;
```

The "Simple Statistics" table in Figure 9.1 displays univariate statistics for the analysis variables.

Figure 9.1 Univariate Statistics

The CORRELATION Procedure

4 Variables	Age	Weight	Oxygen	RunTime

Figure 9.1 *continued*

Simple Statistics						
Variable	N	Mean	Std Dev	Sum	Minimum	Maximum
Age	31	47.67742	5.21144	1478	38.00000	57.00000
Weight	31	77.44452	8.32857	2401	59.08000	91.63000
Oxygen	29	47.22721	5.47718	1370	37.38800	60.05500
RunTime	29	10.67414	1.39194	309.55000	8.17000	14.03000

By default, all numeric variables not listed in other statements are used in the analysis. Observations that have nonmissing values for each variable are used to derive the univariate statistics for that variable.

The "Pearson Correlation Coefficients" table in Figure 9.2 displays the Pearson correlation, the *p*-value under the null hypothesis of zero correlation, and the number of nonmissing observations for each pair of variables.

Figure 9.2 Pearson Correlation Coefficients

Pearson Correlation Coefficients Prob > |r| under H0: Rho=0 Number of Observations				
	Age	Weight	Oxygen	RunTime
Age	1.0000	-0.2335	-0.3147	0.1448
		0.2061	0.0963	0.4536
	31	31	29	29
Weight	-0.2335	1.0000	-0.1536	0.2007
	0.2061		0.4264	0.2965
	31	31	29	29
Oxygen	-0.3147	-0.1536	1.0000	-0.8684
	0.0963	0.4264		<.0001
	29	29	29	28
RunTime	0.1448	0.2007	-0.8684	1.0000
	0.4536	0.2965	<.0001	
	29	29	28	29

By default, Pearson correlation statistics are computed from observations that have nonmissing values for each pair of analysis variables. Figure 9.2 displays a correlation of –0.86843 between Runtime and Oxygen, which is significant with a *p*-value less than 0.0001. That is, an inverse linear relationship exists between these two variables. As Runtime (time in minutes to run 1.5 miles) increases, Oxygen (oxygen intake in milliliters per kilogram body weight per minute) decreases.

Syntax: CORRELATION Procedure

The following statements are available in PROC CORRELATION:

PROC CORRELATION < *options* > ;
  BY *variables* ;
  DISPLAY < *table-list* > < / *options* > ;
  DISPLAYOUT *table-spec-list* < / *options* > ;
  FREQ *variable* ;
  VAR *variables* ;
  WEIGHT *variable* ;
  WITH *variables* ;

The PROC CORRELATION statement is the only required statement for the CORRELATION procedure. The following sections describe the PROC CORRELATION statement and then describe the other statements in alphabetical order.

PROC CORRELATION Statement

PROC CORRELATION < *options* > ;

The PROC CORRELATION statement invokes the procedure. Table 9.2 summarizes the options available in the PROC CORRELATION statement.

Table 9.2 PROC CORRELATION Statement Options

Option	Description
Data Tables	
DATA=	Specifies the input data table
OUTP=	Specifies the output data table to contain the Pearson correlation statistics
Statistical Analysis	
EXCLNPWGT	Excludes observations that have nonpositive weight values from the analysis
NOMISS	Excludes observations that have missing analysis values from the analysis
Statistics	
ALPHA	Computes Cronbach's coefficient alpha
COV	Computes covariances
CSSCP	Computes corrected sums of squares and crossproducts
SSCP	Computes sums of squares and crossproducts
VARDEF=	Specifies the divisor for variance calculations
Printed Output	
BEST=	Displays the specified number of ordered correlation coefficients
NOCORR	Suppresses Pearson correlations
NOPROB	Suppresses *p*-values
NOSIMPLE	Suppresses descriptive statistics
RANK	Displays ordered correlation coefficients

ALPHA

calculates and prints Cronbach's coefficient alpha. PROC CORRELATION computes separate coefficients by using raw and standardized values (scaling the variables to a unit variance of 1). For each analysis variable, PROC CORRELATION computes the correlation between that variable and all the remaining variables. It also computes Cronbach's coefficient alpha by using only the remaining variables.

When you specify this option, the Pearson correlations are also displayed. If you also specify the OUTP= option, the output data table also contains observations that have Cronbach's coefficient alpha. For more information about Cronbach's coefficient alpha, see the section "Cronbach's Coefficient Alpha" on page 300.

This option is not valid if a WITH statement is specified.

BEST=*n*

prints the *n* highest correlation coefficients for each variable, where $n \geq 1$. Correlations are ordered from highest to lowest in absolute value. (If you do not specify this option, PROC CORRELATION prints correlations in a rectangular table, using the variable names as row and column labels.)

COV

displays the variance and covariance matrix. When you specify this option, the Pearson correlations are also displayed. If you also specify the OUTP= option, the output data table also contains the covariance matrix with the corresponding _TYPE_ variable value 'COV.'

CSSCP

displays a table of the corrected sums of squares and crossproducts. When you specify this option, the Pearson correlations are also displayed. If you also specify the OUTP= option, the output data table also contains a CSSCP matrix with the corresponding _TYPE_ variable value 'CSSCP.'

DATA=*libref.data-table*

names the input data table for PROC CORRELATION to use. The default is the most recently created data table. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to the data, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 290.

data-table specifies the name of the input data table.

EXCLNPWGT

EXCLNPWGTS

excludes observations that have nonpositive weight values from the analysis. (By default, PROC CORRELATION treats observations that have negative weights like those observations that have zero weights and counts them in the total number of observations.)

NOCORR
suppresses display of Pearson correlations.

NOMISS
excludes observations that have missing values from the analysis. (If you do not specify this option, PROC CORRELATION computes correlation statistics by using all the nonmissing pairs of variables.)

NOPROB
suppresses display of the probabilities that are associated with each correlation coefficient.

NOSIMPLE
suppresses printing of simple descriptive statistics for each variable. However, if you request an output data table, the output data table still contains simple descriptive statistics for the variables.

OUTP=*libref.data-table*
creates an output data table that contains Pearson correlation statistics.

libref.data-table is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to the data, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 290.

data-table specifies the name of the output data table.

This data table also includes means, standard deviations, and the number of observations. If you also specify the ALPHA option, the output data table also contains six observations that have Cronbach's coefficient alpha.

RANK
displays the ordered correlation coefficients for each variable. Correlations are ordered from highest to lowest in absolute value.

SSCP
displays a table of the sums of squares and crossproducts. When you specify this option, the Pearson correlations are also displayed. If you also specify the OUTP= option, the output data table contains an SSCP matrix and the corresponding _TYPE_ variable value is 'SSCP.'

VARDEF=DF | N | WDF | WEIGHT | WGT
specifies the variance divisor in the calculation of variances and covariances. The default is VARDEF=DF.

Table 9.3 displays available values and associated divisors for the VARDEF= option, where n is the number of nonmissing observations and w_j is the weight associated with the jth nonmissing observation.

Table 9.3 Possible Values for the VARDEF= Option

Value	Description	Divisor
DF	Degrees of freedom	$n - 1$
N	Number of observations	n
WDF	Sum of weights minus one	$\sum_j^n w_j - 1$
WEIGHT | WGT	Sum of weights	$\sum_j^n w_j$

BY Statement

BY *variables* **;**

You can specify a BY statement in PROC CORRELATION to obtain separate analyses of observations in groups that are defined by the values of the BY variables. If you specify more than one BY statement, only the last one specified is used. For more information, see the discussion of BY-group processing in the "Grouping Data" section of *SAS Programmers Guide: Essentials*.

DISPLAY Statement

DISPLAY < *table-list* > < / *options* > **;**

The DISPLAY statement enables you to specify a list of display tables to display or exclude. This statement is similar to the ODS SELECT, ODS EXCLUDE, and ODS TRACE statements. However, the DISPLAY statement can improve performance when a large number of tables could be generated (such as in BY-group processing). The procedure processes the DISPLAY statement on a CAS server and thus sends only a subset of ODS tables to the SAS client. Because ODS statements are processed on a SAS client, first all the generated display tables are sent to the client, and then the client creates a subset.

If you use both DISPLAY and ODS statements together, the DISPLAY statement takes precedence over the ODS statements. Note that the ODS EXCLUDE statement processes tables that are sent to the client after they have been filtered by the DISPLAY statement. In some cases, it might appear that the ODS EXCLUDE statement is taking precedence because it can further filter the tables. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

You can specify the *table-list* as a list of table names, paths, partial pathnames, and regular expressions.

The table names that you can specify are listed in the section "ODS Table Names" on page 303. A path is a table name that is prefixed with dot-separated grouping information. For example, a SelectionSummary table that a procedure produces during a selection routine might have the path *Bygroup1.Summary.SelectionSummary*. A partial pathname does not include all groups; for example, *SelectionSummary* and *Summary.SelectionSummary* are partial pathnames for *Bygroup1.Summary.SelectionSummary*.

When you specify a table name or partial pathname, all display tables whose paths end in the specified name are selected for display or exclusion. For example, both *SelectionSummary* and *Summary.SelectionSummary* select *Bygroup1.Summary.SelectionSummary*.

A regular expression is enclosed in forward slashes (/). For example, specifying "/tions/" selects all pathnames that contain the substring "tions"; in particular, the *Bygroup1.Summary.SelectionSummary* table is selected. Specifying "!/tions/" selects all pathnames that do not contain the substring "tions"; in particular, the *Bygroup1.Summary.SelectionSummary* table is not selected.

You can specify the following *options* after a slash (/):

CASESENSITIVE
performs a case-sensitive comparison of table names in the *table-list* to display table names when tables are subsetted for display. To preserve case, you must enclose table names in the *table-list* in quotation marks.

EXCLUDE
displays all display tables except those that you specify in the *table-list*.

EXCLUDEALL
suppresses display of all tables. This option takes precedence over the other options.

TRACE
displays the display table names, labels, and paths.

DISPLAYOUT Statement

DISPLAYOUT *table-spec-list* < / *options* > **;**

The DISPLAYOUT statement enables you to create output tables from your displayed output. This statement is similar to the ODS OUTPUT statement. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

The *table-spec-list* specifies a list of output tables to create. Each entry in the list has either a *key*=*value* format or a *key* format:

key=*value*	specifies *key* as the ODS table name, path, or partial pathname, and specifies *value* as the output table name.
key	specifies *key* as the ODS table name and also as the output table name.

The ODS table names that you can specify are listed in the section "ODS Table Names" on page 303. You cannot specify the ODS table named OutputCasTables in the *table-spec-list*.

Table names and partial pathnames are discussed under the DISPLAY statement. The DISPLAYOUT statement does not support regular expressions.

You can specify the following *options* after a slash (/):

INCLUDEALL
creates output tables for all display tables. The name of the created output table is the same as the name of the corresponding display table. If you specify this option, the *table-spec-list* specification is ignored.

NOREPLACE
does not replace any existing output table of the same name.

REPEATED
replicates all output tables on all nodes.

FREQ Statement

FREQ *variable* **;**

The *variable* in the FREQ statement identifies a numeric variable in the input data table that contains the frequency of occurrence of each observation. PROC CORRELATION treats each observation as if it appears f times, where f is the value of the FREQ *variable* for the observation. If f is not an integer, it is truncated to an integer. If f is less than 1 or missing, the observation is not used in the analysis. When you omit the FREQ statement, each observation is assigned a frequency of 1.

The effects of the FREQ and WEIGHT statements are similar except when n is not an integer and when degrees of freedom are calculated.

VAR Statement

VAR *variables* **;**

The VAR statement lists *variables* for which to compute correlation coefficients. If the VAR statement is not specified, PROC CORRELATION computes correlations for all numeric variables that are not listed in other statements.

WEIGHT Statement

WEIGHT *variable* **;**

The *variable* in the WEIGHT statement is used in the calculation of Pearson weighted product-moment correlations.

Observations that have missing weights are excluded from the analysis. By default, for observations that have nonpositive weights, weights are set to 0 and the observations are included in the analysis. You can use the EXCLNPWGT option to exclude observations that have negative or zero weights from the analysis. If a WEIGHT statement is not included, all observations that are used in the analysis are assigned a weight of 1.

WITH Statement

WITH *variables* **;**

The WITH statement lists *variables* for which to compute correlation coefficients. The WITH statement requests correlations of the form $r(X_i, Y_j)$, where $Y_1, \ldots, Y_n$ are the row analysis variables that are specified in the WITH statement and $X_1, \ldots, X_m$ are the column analysis variables that are specified in the VAR statement; or, if there is no VAR statement, all numeric variables that are not specified in any other statement. The correlation matrix has a rectangular structure of the form

$$\begin{bmatrix} r(Y_1, X_1) & \cdots & r(Y_1, X_m) \\ \vdots & \ddots & \vdots \\ r(Y_n, X_1) & \cdots & r(Y_n, X_m) \end{bmatrix}$$

For example, the statements

```
proc correlation data=mylib.data;
   var x1 x2;
   with y1 y2 y3;
run;
```

produce correlations for the following combinations:

$$\begin{bmatrix} r(Y1, X1) & r(Y1, X2) \\ r(Y2, X1) & r(Y2, X2) \\ r(Y3, X1) & r(Y3, X2) \end{bmatrix}$$

Details: CORRELATION Procedure

Pearson Product-Moment Correlation

The Pearson product-moment correlation is a parametric measure of association for two variables. It measures both the strength and the direction of a linear relationship. If one variable X is an exact linear function of another variable Y, a positive relationship exists if the correlation is 1, and a negative relationship exists if the correlation is –1. If there is no linear predictability between the two variables, the correlation is 0. If the two variables are normal with correlation 0, the two variables are independent. Correlation does not imply causality because an underlying causal relationship might not exist in some cases.

The formula for the population Pearson product-moment correlation, denoted ρ_{xy}, is

$$\rho_{xy} = \frac{\mathrm{Cov}(x, y)}{\sqrt{\mathrm{V}(x)\mathrm{V}(y)}} = \frac{\mathrm{E}(\,(x - \mathrm{E}(x))(y - \mathrm{E}(y))\,)}{\sqrt{\mathrm{E}(x - \mathrm{E}(x))^2\,\mathrm{E}(y - \mathrm{E}(y))^2}}$$

The sample correlation, such as a Pearson product-moment correlation or weighted product-moment correlation, estimates the population correlation. The formula for the sample Pearson product-moment correlation is

as follows, where $\bar{x}$ is the sample mean of x and $\bar{y}$ is the sample mean of y:

$$r_{xy} = \frac{\sum_i ((x_i - \bar{x})(y_i - \bar{y}))}{\sqrt{\sum_i (x_i - \bar{x})^2 \sum_i (y_i - \bar{y})^2}}$$

The formula for a weighted Pearson product-moment correlation is as follows, where w_i is the weight, $\bar{x}_w$ is the weighted mean of x, and $\bar{y}_w$ is the weighted mean of y:

$$r_{xy} = \frac{\sum_i w_i (x_i - \bar{x}_w)(y_i - \bar{y}_w)}{\sqrt{\sum_i w_i (x_i - \bar{x}_w)^2 \sum_i w_i (y_i - \bar{y}_w)^2}}$$

Probability Values

Probability values for the Pearson correlation are computed by treating the following equation as if it came from a t distribution with $(n - 2)$ degrees of freedom, where r is the sample correlation:

$$t = (n-2)^{1/2} \left(\frac{r^2}{1 - r^2} \right)^{1/2}$$

Cronbach's Coefficient Alpha

Analyzing latent constructs such as job satisfaction, motor ability, sensory recognition, or customer satisfaction requires instruments to accurately measure the constructs. Interrelated items can be summed to obtain an overall score for each participant. Cronbach's coefficient alpha estimates the reliability of this type of scale by determining the internal consistency of the test or the average correlation of items within the test (Cronbach 1951).

When a value is recorded, the observed value contains some degree of measurement error. Two sets of measurements on the same variable for the same individual might not have identical values. However, repeated measurements for a series of individuals do show some consistency. Reliability measures internal consistency from one set of measurements to another. The observed value Y is divided into two components, a true value T and a measurement error E. The measurement error is assumed to be independent of the true value:

$$Y = T + E \quad \mathrm{Cov}(T, E) = 0$$

The reliability coefficient of a measurement test is defined as the squared correlation between the observed value Y and the true value T:

$$r^2(Y, T) = \frac{\mathrm{Cov}(Y, T)^2}{\mathrm{V}(Y)\mathrm{V}(T)} = \frac{\mathrm{V}(T)^2}{\mathrm{V}(Y)\mathrm{V}(T)} = \frac{\mathrm{V}(T)}{\mathrm{V}(Y)}$$

This coefficient is the proportion of the observed variance due to true differences among individuals in the sample. If Y is the sum of several observed variables that measure the same feature, you can estimate $V(T)$. Cronbach's coefficient alpha, which is based on a lower bound for $V(T)$, is an estimate of the reliability coefficient.

Suppose p variables are used with $Y_j = T_j + E_j$ for $j = 1, 2, \ldots, p$, where Y_j is the observed value, T_j is the true value, and E_j is the measurement error. The measurement errors (E_j) are independent of the true

values (T_j) and are also independent of each other. Let $Y_0 = \sum_j Y_j$ be the total observed score, and let $T_0 = \sum_j T_j$ be the total true score. Because

$$(p-1)\sum_j V(T_j) \geq \sum_{i \neq j} \mathrm{Cov}(T_i, T_j)$$

a lower bound for $V(T_0)$ is given by

$$\frac{p}{p-1}\sum_{i \neq j} \mathrm{Cov}(T_i, T_j)$$

With $\mathrm{Cov}(Y_i, Y_j) = \mathrm{Cov}(T_i, T_j)$ for $i \neq j$, a lower bound for the reliability coefficient, $V(T_0)/V(Y_0)$, is then given by Cronbach's coefficient alpha:

$$\alpha = \left(\frac{p}{p-1}\right)\frac{\sum_{i \neq j} \mathrm{Cov}(Y_i, Y_j)}{V(Y_0)} = \left(\frac{p}{p-1}\right)\left(1 - \frac{\sum_j V(Y_j)}{V(Y_0)}\right)$$

If the variances of the items vary widely, you can standardize the items to a standard deviation of 1 before computing the coefficient alpha. If the variables are dichotomous (0,1), the coefficient alpha is equivalent to the Kuder-Richardson 20 (KR-20) reliability measure.

When the correlation between each pair of variables is 1, the coefficient alpha has a maximum value of 1. With negative correlations between some variables, the coefficient alpha can have a value less than 0. The larger the overall alpha coefficient, the more likely that items contribute to a reliable scale. Nunnally and Bernstein (1994) suggest 0.70 as an acceptable reliability coefficient; smaller reliability coefficients are seen as inadequate. However, acceptability varies by discipline.

To determine how each item reflects the reliability of the scale, you calculate a coefficient alpha after deleting each variable independently from the scale. Cronbach's coefficient alpha from all variables except the *k*th variable is given by

$$\alpha_k = \left(\frac{p-1}{p-2}\right)\left(1 - \frac{\sum_{i \neq k} V(Y_i)}{V(\sum_{i \neq k} Y_i)}\right)$$

If the reliability coefficient increases after an item is deleted from the scale, you can assume that the item is not highly correlated with other items in the scale. Conversely, if the reliability coefficient decreases, you can assume that the item is highly correlated with other items in the scale. For more information about how to interpret Cronbach's coefficient alpha, see Yu (2001).

Listwise deletion of observations that have missing values is necessary to correctly calculate Cronbach's coefficient alpha. PROC CORRELATION does not automatically use listwise deletion if you specify the ALPHA option. Therefore, you should use the NOMISS option if the data table contains missing values. Otherwise, PROC CORRELATION prints a warning message that indicates that you need to use the NOMISS option with the ALPHA option.

Missing Values

Observations that have missing values for the variables in the FREQ or WEIGHT statement are omitted from the analysis.

By default, PROC CORRELATION uses *pairwise deletion* when observations contain missing values in the analysis variables for which correlation coefficients are computed. PROC CORRELATION includes all nonmissing pairs of values for each pair of variables in the statistical computations. Therefore, the correlation statistics might be based on different numbers of observations.

If you specify the NOMISS option, PROC CORRELATION uses *listwise deletion* and excludes all observations that have missing values from the analysis. Therefore, the number of observations for each pair of variables is identical.

OUTP= Data Table

If you specify the OUTP= option, PROC CORRELATION creates an output data table that contains statistics for Pearson correlation.

The new data table contains the following variables:

- BY variables, if any
- _TYPE_ variable, which identifies the type of observation
- _NAME_ variable, which identifies the variable that corresponds to a particular row of the correlation matrix
- INTERCEPT variable, which identifies variable sums when the SSCP option is specified
- the column analysis variables that are specified in the VAR statement or, if there is no VAR statement, all numeric variables that are not specified in any other statement

You can use a combination of the _TYPE_ and _NAME_ variables to identify the contents of an observation. The _NAME_ variable indicates which row of the correlation matrix the observation corresponds to. The values of the _TYPE_ variable are as follows:

- SSCP, uncorrected sums of squares and crossproducts
- CSSCP, corrected sums of squares and crossproducts
- COV, covariances
- MEAN, mean of each variable
- STD, standard deviation of each variable
- N, number of nonmissing observations for each variable
- SUMWGT, sum of the weights for each variable when a WEIGHT statement is specified

- CORR, correlation statistics for each variable

If you specify the SSCP option, the OUTP= data table includes an additional observation that contains intercept values. If you specify the ALPHA option, the OUTP= data table also includes observations that have the following _TYPE_ values:

- RAWALPHA, Cronbach's coefficient alpha for raw variables
- STDALPHA, Cronbach's coefficient alpha for standardized variables
- RAWALDEL, Cronbach's coefficient alpha for raw variables after deleting one variable
- STDALDEL, Cronbach's coefficient alpha for standardized variables after deleting one variable
- RAWCTDEL, the correlation between a raw variable and the total of the remaining raw variables
- STDCTDEL, the correlation between a standardized variable and the total of the remaining standardized variables

The output data table does not have a special type like the OUTP= data table that PROC CORR creates. Therefore, if the OUTP= output data table that PROC CORRELATION creates is provided as input to a SAS/STAT procedure that recognizes special data table types (such as the REG and FACTOR procedures), the input type must also be specified. For example, the following statements produce an output data table mylib.outdata:

```
proc correlation data=mylib.data outp=mylib.outdata;
run;

proc factor data=mylib.outdata(Type=CORR);
run;
```

The mylib.outdata data table is then provided to PROC FACTOR as input with the **(Type=CORR)** option to indicate that the input data table contains a correlation matrix. This might save computational time when the initial data table mylib.data is large and the computation of the correlation matrix can be done using multiple threads.

ODS Table Names

PROC CORRELATION assigns a name to each table it creates. You must use these names to refer to tables when you use the DISPLAY statement, the DISPLAYOUT, or the Output Delivery System (ODS). These names are listed in Table 9.4.

Table 9.4 ODS Tables Produced by PROC CORRELATION

ODS Table Name	Description	Option
Cov	Covariances	COV
CronbachAlpha	Coefficient alpha	ALPHA
CronbachAlphaDel	Coefficient alpha with deleted variable	ALPHA
Csscp	Corrected sums of squares and crossproducts	CSSCP
PearsonCorr	Pearson correlations	Default output
SimpleStats	Simple descriptive statistics	Default output
Sscp	Sums of squares and crossproducts	SSCP
VarInformation	Variable information	Default output

Examples: CORRELATION Procedure

The examples in this chapter are available in the GitHub repository located at `https://github.com/sascommunities/doc-supplement-casstat`.

NOTE: Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections "Using CAS Sessions and CAS Engine Librefs" on page 7 and "Loading a SAS Data Set onto a CAS Server" on page 7 in Chapter 2, "Shared Concepts."

Example 9.1: Computing Correlations between Two Sets of Variables

(View the complete code for this example (cascorrex1.sas) in the example repository.)

The following statements create the data table Setosa, which contains measurements for four iris parts from Fisher's iris data (1936): sepal length, sepal width, petal length, and petal width. The data table has been altered to contain some missing values.

```
*------------------- Data on Iris Setosa --------------------*
| The data table contains 50 iris specimens from the species |
| Iris Setosa with the following four measurements:          |
| SepalLength (sepal length)                                 |
| SepalWidth  (sepal width)                                  |
| PetalLength (petal length)                                 |
| PetalWidth  (petal width)                                  |
| Certain values were changed to missing for the analysis.   |
*------------------------------------------------------------*;
data mylib.Setosa;
  input SepalLength SepalWidth PetalLength PetalWidth @@;
  label sepallength='Sepal Length in mm.'
        sepalwidth='Sepal Width in mm.'
        petallength='Petal Length in mm.'
        petalwidth='Petal Width in mm.';
```

```
  datalines;
50 33 14 02  46 34 14 03  46 36 .  02
51 33 17 05  55 35 13 02  48 31 16 02
52 34 14 02  49 36 14 01  44 32 13 02
50 35 16 06  44 30 13 02  47 32 16 02
48 30 14 03  51 38 16 02  48 34 19 02
50 30 16 02  50 32 12 02  43 30 11 .
58 40 12 02  51 38 19 04  49 30 14 02
51 35 14 02  50 34 16 04  46 32 14 02
57 44 15 04  50 36 14 02  54 34 15 04
52 41 15 .   55 42 14 02  49 31 15 02
54 39 17 04  50 34 15 02  44 29 14 02
47 32 13 02  46 31 15 02  51 34 15 02
50 35 13 03  49 31 15 01  54 37 15 02
54 39 13 04  51 35 14 03  48 34 16 02
48 30 14 01  45 23 13 03  57 38 17 03
51 38 15 03  54 34 17 02  51 37 15 04
52 35 15 02  53 37 15 02
;
```

The following statements request a correlation analysis between two sets of variables, the sepal measurements (length and width) and the petal measurements (length and width):

```
title 'Fisher (1936) Iris Setosa Data';
proc correlation data=mylib.Setosa sscp cov;
   var  sepallength sepalwidth;
   with petallength petalwidth;
run;
```

The "Simple Statistics" table in Output 9.1.1 displays univariate statistics for the analysis variables that are specified in the VAR and WITH statements.

Output 9.1.1 Simple Statistics

Fisher (1936) Iris Setosa Data

The CORRELATION Procedure

Simple Statistics

Variable	N	Mean	Std Dev	Sum	Minimum	Maximum	Label
PetalLength	49	14.71429	1.62019	721.00000	11.00000	19.00000	Petal Length in mm.
PetalWidth	48	2.52083	1.03121	121.00000	1.00000	6.00000	Petal Width in mm.
SepalLength	50	50.06000	3.52490	2503	43.00000	58.00000	Sepal Length in mm.
SepalWidth	50	34.28000	3.79064	1714	23.00000	44.00000	Sepal Width in mm.

When the WITH statement is specified together with the VAR statement, the CORRELATION procedure produces rectangular matrices for statistics such as covariances and correlations. The matrix rows correspond to the variables specified in the WITH statement (PetalLength and PetalWidth), and the matrix columns correspond to the variables specified in the VAR statement (SepalLength and SepalWidth). The CORRELATION procedure uses the WITH variable labels to label the matrix rows.

The SSCP option requests a table of the uncorrected sum of squares and crossproducts matrix, and the COV option requests a table of the covariance matrix.

The sum of squares and crossproducts statistics for each pair of variables are computed by using observations that have nonmissing row and column variable values. The "Sums of Squares and Crossproducts" table in Output 9.1.2 displays the crossproduct, the sum of squares for the row variable, and the sum of squares for the column variable for each pair of variables.

Output 9.1.2 Sums of Squares and Crossproducts

Sums of Squares and Crossproducts
SSCP / Row Var SS / Col Var SS

	SepalLength	SepalWidth
PetalLength	36214.0000	24756.0000
Petal Length in mm.	10735.0000	10735.0000
	123793.0000	58164.0000
PetalWidth	6113.0000	4191.0000
Petal Width in mm.	355.0000	355.0000
	121356.0000	56879.0000

The variances are computed by using observations that have nonmissing values for the analysis variables. The "Variances and Covariances" table in Output 9.1.3 displays the covariance, variance for the row variable, variance for the column variable, and associated degrees of freedom for each pair of variables.

Output 9.1.3 Variances and Covariances

Variances and Covariances
Covariance / Row Var Variance / Col Var Variance / DF

	SepalLength	SepalWidth
PetalLength	1.270833	1.363095
Petal Length in mm.	2.625000	2.625000
	12.333333	14.605442
	48	48
PetalWidth	0.911348	1.048316
Petal Width in mm.	1.063387	1.063387
	11.801418	13.627216
	47	47

When there are missing values in the analysis variables, the "Pearson Correlation Coefficients" table in Output 9.1.4 displays the correlation, the p-value under the null hypothesis of zero correlation, and the number of observations for each pair of variables.

Output 9.1.4 Pearson Correlation Coefficients

Pearson Correlation Coefficients
Prob > |r| under H0: Rho=0
Number of Observations

	Sepal Length in mm.	Sepal Width in mm.
PetalLength	0.2233	0.2201
Petal Length in mm.	0.1229	0.1285
	49	49
PetalWidth	0.2573	0.2754
Petal Width in mm.	0.0775	0.0582
	48	48

Example 9.2: Computing Cronbach's Coefficient Alpha

(View the complete code for this example (correx2.sas) in the example repository.)

The following statements create the data table Fish1. The cubic root of the weight (Weight3) is computed as a one-dimensional measure of the size of a fish.

```
*------------------- Fish Measurement Data -----------------------*
| The data table contains 35 fish from the species Bream caught   |
| in Finland's Lake Laengelmavesi with the following measurements:|
| Weight   (in grams)                                             |
| Length3  (length from the nose to the end of the tail, in cm)   |
| HtPct    (max height, as percentage of Length3)                 |
| WidthPct (max width,  as percentage of Length3)                 |
*-----------------------------------------------------------------*;
data mylib.Fish1 (drop=HtPct WidthPct);
   title 'Fish Measurement Data';
   input Weight Length3 HtPct WidthPct @@;
   Weight3= Weight**(1/3);
   Height=HtPct*Length3/100;
   Width=WidthPct*Length3/100;
   datalines;
242.0 30.0 38.4 13.4     290.0 31.2 40.0 13.8
340.0 31.1 39.8 15.1     363.0 33.5 38.0 13.3
430.0 34.0 36.6 15.1     450.0 34.7 39.2 14.2
500.0 34.5 41.1 15.3     390.0 35.0 36.2 13.4
450.0 35.1 39.9 13.8     500.0 36.2 39.3 13.7
475.0 36.2 39.4 14.1     500.0 36.2 39.7 13.3
500.0 36.4 37.8 12.0        .  37.3 37.3 13.6
600.0 37.2 40.2 13.9     600.0 37.2 41.5 15.0
700.0 38.3 38.8 13.8     700.0 38.5 38.8 13.5
610.0 38.6 40.5 13.3     650.0 38.7 37.4 14.8
575.0 39.5 38.3 14.1     685.0 39.2 40.8 13.7
620.0 39.7 39.1 13.3     680.0 40.6 38.1 15.1
700.0 40.5 40.1 13.8     725.0 40.9 40.0 14.8
720.0 40.6 40.3 15.0     714.0 41.5 39.8 14.1
850.0 41.6 40.6 14.9    1000.0 42.6 44.5 15.5
920.0 44.1 40.9 14.3     955.0 44.0 41.1 14.3
925.0 45.3 41.4 14.9     975.0 45.9 40.6 14.7
950.0 46.5 37.9 13.7
;
```

The following statements request a correlation analysis and compute Cronbach's coefficient alpha for the variables Weight3, Length3, Height, and Width:

```
title 'Fish Measurement Data';
proc correlation data=mylib.fish1 nomiss alpha;
   var Weight3 Length3 Height Width;
run;
```

The ALPHA option computes Cronbach's coefficient alpha for the analysis variables.

The "Simple Statistics" table in Output 9.2.1 displays univariate descriptive statistics for each analysis variable.

Output 9.2.1 Simple Statistics

Fish Measurement Data

The CORRELATION Procedure

Simple Statistics

Variable	N	Mean	Std Dev	Sum	Minimum	Maximum
Weight3	34	8.44751	0.97574	287.21524	6.23168	10.00000
Length3	34	38.38529	4.21628	1305	30.00000	46.50000
Height	34	15.22057	1.98159	517.49950	11.52000	18.95700
Width	34	5.43805	0.72967	184.89370	4.02000	6.74970

The "Pearson Correlation Coefficients" table in Output 9.2.2 displays Pearson correlation statistics for pairs of analysis variables.

Output 9.2.2 Pearson Correlation Coefficients

Pearson Correlation Coefficients N = 34
Prob > |r| under H0: Rho=0

	Weight3	Length3	Height	Width
Weight3	1.0000	0.9652 <.0001	0.9626 <.0001	0.9279 <.0001
Length3	0.9652 <.0001	1.0000	0.9549 <.0001	0.9217 <.0001
Height	0.9626 <.0001	0.9549 <.0001	1.0000	0.9263 <.0001
Width	0.9279 <.0001	0.9217 <.0001	0.9263 <.0001	1.0000

Because the data table contains only one species of fish, all the variables are highly correlated. Using the ALPHA option, the CORRELATION procedure computes Cronbach's coefficient alpha in Output 9.2.3. The Cronbach's coefficient alpha is a lower bound for the reliability coefficient for the raw variables and the standardized variables. Positive correlation is needed for the alpha coefficient because variables measure a common entity.

Output 9.2.3 Cronbach's Coefficient Alpha

Cronbach Coefficient Alpha

Variables	Alpha
Raw	0.822134
Standardized	0.985145

Because the variances of some variables vary widely, you should use the standardized score to estimate reliability. The overall standardized Cronbach's coefficient alpha of 0.985145 provides an acceptable lower bound for the reliability coefficient. This is much greater than the suggested value of 0.70 given by Nunnally and Bernstein (1994).

The standardized alpha coefficient provides information about how each variable reflects the reliability of the scale with standardized variables. If the standardized alpha decreases after removing a variable from the construct, then this variable is strongly correlated with other variables in the scale. On the other hand, if the

standardized alpha increases after removing a variable from the construct, then removing this variable from the scale makes the construct more reliable. The "Cronbach Coefficient Alpha with Deleted Variables" table in Output 9.2.4 does not show a significant increase or decrease in the standardized alpha coefficients. For more information about Cronbach's alpha, see the section "Cronbach's Coefficient Alpha" on page 300.

Output 9.2.4 Cronbach's Coefficient Alpha with Deleted Variables

Cronbach Coefficient Alpha with Deleted Variable			
Raw Variables		Standardized Variables	
Correlation with Total	Alpha	Correlation with Total	Alpha
0.975379	0.783365	0.973464	0.977103
0.967602	0.881987	0.967177	0.978783
0.964715	0.655098	0.968079	0.978542
0.934635	0.824069	0.937599	0.986626

Example 9.3: Saving Correlations in an Output Data Table

(View the complete code for this example (correx3.sas) in the example repository.)

The following statements compute Pearson correlations for the Fitness data table that is created in the section "Getting Started: CORRELATION Procedure" on page 291:

```
title 'Correlations for a Fitness and Exercise Study';
proc correlation data=mylib.Fitness nomiss outp=mylib.CorrOutp;
   var weight oxygen runtime;
run;
```

The NOMISS option excludes from the analysis observations that have missing values of the variables that are specified in the VAR statement—that is, the same set of 28 observations is used to compute the correlation for each pair of variables. The OUTP= option creates an output data table named CorrOutp to contain the Pearson correlation statistics.

The "Pearson Correlation Coefficients" table in Output 9.3.1 displays the correlation and the *p*-value under the null hypothesis of zero correlation.

Output 9.3.1 Pearson Correlation Coefficients

Correlations for a Fitness and Exercise Study

The CORRELATION Procedure

Pearson Correlation Coefficients N = 28 Prob > |r| under H0: Rho=0			
	Weight	Oxygen	RunTime
Weight	1.0000	-0.1842 0.3481	0.1950 0.3199
Oxygen	-0.1842 0.3481	1.0000	-0.8684 <.0001
RunTime	0.1950 0.3199	-0.8684 <.0001	1.0000

The following statements display the output data table that is shown in Output 9.3.2:

```
title 'Output Data Table from PROC CORRELATION';
proc print data=mylib.CorrOutp noobs;
run;
```

Output 9.3.2 OUTP= Data Table with Pearson Correlations

Output Data Table from PROC CORRELATION

TYPE	_NAME_	Weight	Oxygen	RunTime
MEAN		77.2168	47.1327	10.6954
STD		8.4495	5.5535	1.4127
N		28.0000	28.0000	28.0000
CORR	Weight	1.0000	-0.1842	0.1950
CORR	Oxygen	-0.1842	1.0000	-0.8684
CORR	RunTime	0.1950	-0.8684	1.0000

The output data table can be used as an input data table for regression or other statistical procedures. For example, the following statements request a regression analysis that uses CorrOutp instead of reading the original data in the REG procedure:

```
title 'Input Type CORR Data Set from PROC REG';
proc reg data=mylib.CorrOutp(type=CORR);
   model runtime= weight oxygen;
run;
```

The following statements read the original data and generate the same results as the preceding statements:

```
proc reg data=mylib.Fitness;
   model runtime= weight oxygen;
run;
```

References

Cronbach, L. J. (1951). "Coefficient Alpha and the Internal Structure of Tests." *Psychometrika* 16:297–334.

Nunnally, J. C., and Bernstein, I. H. (1994). *Psychometric Theory*. 3rd ed. New York: McGraw-Hill.

Yu, C. H. (2001). "An Introduction to Computing and Interpreting Cronbach Coefficient Alpha in SAS." In *Proceedings of the Twenty-Sixth Annual SAS Users Group International Conference*. Cary, NC: SAS Institute Inc. http://www2.sas.com/proceedings/sugi26/p246-26.pdf.

Chapter 10
EFA Procedure

Contents

Overview: EFA Procedure **312**
PROC EFA Features 312
PROC EFA Compared with Other SAS Procedures 313
Using CAS Sessions and CAS Engine Librefs 314
Getting Started: EFA Procedure **315**
Syntax: EFA Procedure **324**
PROC EFA Statement 324
BY Statement 332
DISPLAY Statement 332
DISPLAYOUT Statement 333
FREQ Statement 334
NFACTORS Statement 334
PRIORS Statement 336
VAR Statement 337
WEIGHT Statement 337
Details: EFA Procedure **337**
The Common Factor Model 337
The Exploratory Nature of Common Factor Analysis 339
Factor Analysis in Practice 340
Prior Communalities 341
Determining the Number of Factors 342
Factor Extraction 343
Rotating Factors 346
Variable Weights and Variance Explained 348
Limitations 349
Missing Values 349
ODS Table Names 349
Examples: EFA Procedure **351**
Example 10.1: Estimating the Number of Factors or Components 351
Example 10.2: Principal Factor Analysis 355
Example 10.3: Iterative Factor Extraction 361
Example 10.4: Extracting and Interpreting Solutions with Maximum Likelihood Factor Analysis 366
References **377**

Overview: EFA Procedure

The EFA procedure provides analytic tools that enable you to conduct exploratory common factor analysis in SAS Viya. In the psychometric and behavioral science literature, common factor analysis is often referred to simply as factor analysis.

Common factor analysis was invented by Spearman (1904). Kim and Mueller (1978a, b) provide an elementary discussion of the common factor model. Gorsuch (1974) presents a broad survey of factor analysis, and Gorsuch (1974) and Cattell (1978) offer useful guides to practical research methodology. Harman (1976) gives a lucid discussion of many of the more technical aspects of factor analysis, especially oblique rotation. Morrison (1976) and Mardia, Kent, and Bibby (1979) provide excellent statistical treatments of common factor analysis. Mulaik (1972) provides the most thorough and authoritative general reference on factor analysis; this book is highly recommended to anyone familiar with matrix algebra. Stewart (1981) gives a nontechnical presentation of some issues to consider in deciding whether or not a factor analysis is appropriate.

For more information about the common factor model, see the section "The Common Factor Model" on page 337. For more information about the steps that are involved in exploratory factor analysis, see the section "Factor Analysis in Practice" on page 340.

PROC EFA Features

PROC EFA provides the following features:

- supports the analysis of multivariate input data
- computes and analyzes the correlation matrix for the analysis variables
- provides weighted analysis for multivariate input data
- produces output data tables that contain means, standard deviations, number of observations, and eigenvalues
- supports multiple algorithms for determining the number of factors, including Horn's parallel analysis (Horn 1965) and Velicer's minimum average partial correlation analysis (Velicer 1976)
- supports multiple algorithms for factor extraction, including principal factor extraction (iterated or noniterated), maximum likelihood estimation, and the least squares method
- supports multiple algorithms for factor rotation, including promax and the generalized Crawford-Ferguson family of rotations
- provides various methods (for example, squared multiple correlations) of specifying initial communality estimates

Because PROC EFA runs on SAS Cloud Analytic Services (CAS), it also does the following:

- enables you to run on a cluster of machines that distribute the data and the computations

- enables you to run in single-machine mode
- exploits available cores and concurrent threads

PROC EFA Compared with Other SAS Procedures

The EFA procedure provides functionality to perform exploratory common factor analysis that is comparable to that of the FACTOR procedure in SAS/STAT software.

There are two important differences between PROC EFA and PROC FACTOR. First, in PROC EFA the default prior communality estimates are always squared multiple correlations (that is, PRIORS=SMC is the default option). For PROC FACTOR, the default method of computing prior communalities depends on the factor extraction method. Second, the NFACTORS= option and the NFACTORS statement in PROC EFA enable you to compute and combine an arbitrary number of criteria to determine the number of factors to extract. In PROC FACTOR, the capabilities for exploring multiple criteria to determine the number of factors are more limited.

Additional similarities and differences are summarized in Table 10.1.

Table 10.1 Comparison of PROC EFA and PROC FACTOR

Feature	PROC EFA	PROC FACTOR
Supports multiple simultaneous methods of determining the number of factors to extract	Yes	Partially
Supports BY statement	Yes	Yes
Supports DISPLAY and DISPLAYOUT statements	Yes	No
Supports PLOTS= option	No	Yes
Input data types	Raw data only	Raw data or certain types of special SAS data sets
Threading	Yes	Mostly single-threaded
Displays reference structure for oblique rotations	REFSTRUCT option	Default
Performs factor scoring	No	Yes
Performs standard error estimation for ML	No	Yes
Performs Procrustes rotation to a fixed target	No	Yes

When you specify the PRIORS=ONES and METHOD=PRINCIPAL options in the PROC EFA statement,

the procedure performs a principal component analysis. This analysis is comparable to that performed by the PCA procedure when you specify the METHOD=EIG option.

Using CAS Sessions and CAS Engine Librefs

SAS Cloud Analytic Services (CAS) is the analytic server and associated cloud services in SAS Viya. This section describes how to create a CAS session and set up a CAS engine libref that you can use to connect to the CAS session. It assumes that you have a CAS server already available; contact your system administrator if you need help starting and terminating a server. This CAS server is identified by specifying the host on which it runs and the port on which it listens for communications. To simplify your interactions with this CAS server, the host information and port information for the server are stored as SAS option values that are retrieved automatically whenever this CAS server needs to be accessed. You can examine the host and port values for the server at your site by using the following statements:

```
proc options option=(CASHOST CASPORT);
run;
```

In addition to starting a CAS server, your system administrator might also have created a CAS session and a CAS engine libref for your use. You can define your own sessions and CAS engine librefs that connect to the CAS server as shown in the following statements:

```
cas mysess;
libname mylib cas sessref=mysess;
```

The CAS statement creates the CAS session named mysess, and the LIBNAME statement creates the mylib CAS engine libref that you use to connect to this session. It is not necessary to explicitly name the CASHOST and CASPORT of the CAS server in the CAS statement, because these values are retrieved from the corresponding SAS option values.

If you have created the mysess session, you can terminate it by using the TERMINATE option in the CAS statement as follows:

```
cas mysess terminate;
```

For more information about the CAS and LIBNAME statements, see the section “Introduction to Shared Concepts” on page 6 in Chapter 2, “Shared Concepts.”

Getting Started: EFA Procedure

(View the complete code for this example at https://github.com/sascommunities/doc-supplement-casstat/tree/main/Examples/efags.sas.)

NOTE: Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections "Using CAS Sessions and CAS Engine Librefs" on page 7 and "Loading a SAS Data Set onto a CAS Server" on page 7 in Chapter 2, "Shared Concepts."

This example demonstrates how you can use the EFA procedure to perform common factor analysis and factor rotation. The example uses the jobratings data set that is described in the examples for the FACTOR procedure in SAS/STAT software. In the original data set, 103 police officers were rated by their supervisors on 14 scales (variables). The current example excludes one of these scales—the overall rating variable.

The following DATA step creates the table jobratings:

```
options validvarname=any;
data mylib.jobratings;
   input ('Communication Skills'n
      'Problem Solving'n
      'Learning Ability'n
      'Judgment under Pressure'n
      'Observational Skills'n
      'Willingness to Confront Problems'n
      'Interest in People'n
      'Interpersonal Sensitivity'n
      'Desire for Self-Improvement'n
      'Appearance'n
      'Dependability'n
      'Physical Ability'n
      'Integrity'n) (1.);
   datalines;
2683885387986
7475887685766
5675786377587
6786977798899
9999779887888
8989789988879
8999988989979
8779479846888
3565233514311

   ... more lines ...

9989989989989
7665639956748
;
```

These statements assume that your libref is named mylib, but you can substitute any appropriately defined libref.

You conduct a common factor analysis on these variables to see what latent factors are operating behind these

ratings. As part of this analysis, you want to determine the number of latent factors that are likely to explain the data.

The following statements invoke the EFA procedure:

```
proc efa data=mylib.jobratings rotate=varimax;
   nfactors type=parallel alpha=0.05 seed=1031 nsimulations=20000;
   nfactors type=parallel alpha=0.05 seed=1128 nsimulations=20000;
   nfactors type=parallel alpha=0.10 seed=1031 nsimulations=20000;
   nfactors type=parallel alpha=0.10 seed=1128 nsimulations=20000;
   nfactors type=eigenvalue threshold=0.5  status=inactive;
   nfactors type=eigenvalue threshold=1.0  status=inactive;
   nfactors type=proportion threshold=0.9  status=inactive;
   nfactors type=proportion threshold=0.95 status=inactive;
   nfactors type=map2 status=inactive;
   nfactors type=map4 status=inactive;
run;
```

In the common factor model, you always assume that observed variables are functions of underlying latent factors. (For more information about the common factor model, see the section "The Common Factor Model" on page 337.) The goal of factor analysis is to extract and interpret a factor pattern that describes these relationships. By default, PROC EFA uses the principal factor extraction method, with prior communalities set to the squared multiple correlations (SMC) of each variable with all the other variables. You can use the METHOD= option to specify a different factor extraction method. You can use the PRIORS= option to specify different prior communality estimates. To facilitate interpretations, the ROTATE= option specifies the varimax orthogonal factor rotation.

PROC EFA supports various criteria that you can use to suggest the number of factors to extract. You use the NFACTORS statement to specify a criterion. You can use multiple NFACTORS statements to specify multiple criteria. In this example, you determine the number of factors to extract by using multiple parallel analyses, each with 20,000 simulations. You include two different thresholds for the parallel analyses and two different random number seeds. By default, the actual number of factors that are extracted is the minimum of the numbers that are suggested by the four parallel analyses. You can use the NFACTORS=*method* option to change this default behavior, where *method* is MIN, MAX, MEDIAN, or MEAN.

In this example, you specify multiple additional NFACTORS statements, all of which include the STATUS=INACTIVE option. When a criterion has an inactive status, it is computed and the results are displayed, but it is not used to determine the final number of factors that are extracted. This can be useful for investigating the robustness of the number of extracted factors to alternative criteria. The minimum eigenvalue criterion (TYPE=EIGENVALUE) computes the number of factors to extract by comparing the eigenvalues of the reduced correlation matrix to a threshold value. The proportion of common variance criterion (TYPE=PROPORTION) computes the number of factors to extract by determining the smallest number of factors such that the cumulative proportion of common variance explained exceeds a threshold value. The minimum average partial correlation analyses (TYPE=MAP2 and MAP4) compute the number of factors by determining the number of principal components that, when partialed out, results in the smallest average squared partial (or residual) correlations.

If you know the exact number of factors that you want to extract in an analysis, you can omit the NFACTORS statement and specify the NFACTORS=*n* option in the PROC EFA statement, where *n* is the number of factors that you want to extract.

The output from the factor analysis is displayed in Figure 10.1 through Figure 10.11.

The first outputs that PROC EFA produces are the "Model Information" table and a table that summarizes the number of observations in the input data table. These are shown in Figure 10.1. Together, these tables summarize important information about the input data that are analyzed as well as the primary analysis options that you specified. For this analysis, the numbers of observations that are used is the same as the number that are read. These numbers would be different if some observations were discarded from the analysis. PROC EFA uses only complete cases for analysis, so a record is discarded if any field contains a missing value.

Figure 10.1 Model Information and Number of Observations

Model Information	
Data Source	JOBRATINGS
Data Type	Raw
Prior Communalities	Squared Multiple Correlations
Factor Extraction	Principal Factor Method
Factor Rotation	Varimax

Number of Observations Read	103
Number of Observations Used	103

Next, the "Simple Statistics" table in Figure 10.2 summarizes the means and standard deviations of the analysis variables.

Figure 10.2 Simple Statistics

Simple Statistics		
Variable	**Mean**	**Standard Deviation**
Communication Skills	6.65049	1.76407
Problem Solving	6.63107	1.59035
Learning Ability	6.99029	1.33941
Judgment under Pressure	6.73786	1.73183
Observational Skills	6.93204	1.76158
Willingness to Confront Problems	7.29126	1.52516
Interest in People	6.70874	1.89235
Interpersonal Sensitivity	6.62136	1.76077
Desire for Self-Improvement	6.57282	1.72980
Appearance	7.00000	1.79869
Dependability	6.82524	1.91704
Physical Ability	7.20388	1.55525
Integrity	7.21359	1.84524

The summary output for the determination of the number of factors is shown in Figure 10.3. For most of the specified criteria, the results suggest that two latent factors should be retained to explain the data. The minimum eigenvalue criterion, with a threshold of 0.50, and the cumulative proportion criterion, with a threshold of 0.95, suggest the existence of a third latent factor. All four parallel analysis criteria suggest two latent factors, so the final number of latent factors that the analysis suggests is two.

Figure 10.3 Number of Factors

Determination of Number of Factors		
Criterion	**Description**	**Factors**
1	Parallel Analysis (Alpha = 0.05, Seed = 1031, Sims = 20000)	2
2	Parallel Analysis (Alpha = 0.05, Seed = 1128, Sims = 20000)	2
3	Parallel Analysis (Alpha = 0.10, Seed = 1031, Sims = 20000)	2
4	Parallel Analysis (Alpha = 0.10, Seed = 1128, Sims = 20000)	2
5	Minimum Eigenvalue (Eigenvalue > 0.50)	3
6	Minimum Eigenvalue (Eigenvalue > 1.00)	2
7	Total Proportion (Threshold = 0.90)	2
8	Total Proportion (Threshold = 0.95)	3
9	Min Ave Squared Partial Corr (MAP2)	2
10	Min Ave 4th-powered Partial Corr (MAP4)	2
Min = 2, Max = 2, Mean = 2, Median = 2		

Additional information about the minimum eigenvalue and cumulative proportion criteria is presented in the "Eigenvalues of the Reduced Correlation Matrix" table in Figure 10.4. After the data correlation matrix is reduced by replacing its diagonal elements with the SMC prior communality estimates, the largest eigenvalue of the resulting matrix is approximately 6.18, and the corresponding latent factor explains approximately 77% of the common variance of the input data. The second eigenvalue is approximately 1.46, and the latent factors that correspond to these first two eigenvalues explain almost 95% of the common variance. A third eigenvalue has a value of approximately 0.56, and the latent factors that correspond to these three eigenvalues explain more than 100% of the common variance. Thus, two latent factors are sufficient to explain up to 90% of the common variance in the input data, and a third factor is needed to explain more than 95% of the common variance. Similarly, a minimum eigenvalue threshold of 1.00 produces an estimate of two latent factors, whereas a third latent factor is estimated if the minimum eigenvalue threshold is 0.50. (The fourth-largest eigenvalue for the reduced correlation matrix has a value of approximately 0.28.)

Figure 10.4 Eigenvalues and Common Variance Explained

Eigenvalues of the Reduced Correlation Matrix				
	Eigenvalue	**Difference**	**Proportion**	**Cumulative**
1	6.177605	4.715319	0.7660	0.7660
2	1.462286	0.901833	0.1813	0.9473
3	0.560453	0.280939	0.0695	1.0168
4	0.279513	0.047660	0.0347	1.0515
5	0.231853	0.161134	0.0287	1.0802
6	0.070719	0.074896	0.0088	1.0890
7	-0.004177	0.033875	-0.0005	1.0885
8	-0.038053	0.047765	-0.0047	1.0838
9	-0.085818	0.024381	-0.0106	1.0731
10	-0.110199	0.014527	-0.0137	1.0595
11	-0.124726	0.023565	-0.0155	1.0440
12	-0.148291	0.058236	-0.0184	1.0256
13	-0.206527		-0.0256	1.0000

The parallel analysis results are summarized in the tables in Figure 10.5. For each of the four specified parallel analysis criteria, correlation matrices are simulated from a Wishart distribution. These simulations

represent correlation matrices that might be realized for a data-generating process in which all variables are independently and identically distributed according to the standard normal distribution. The eigenvalues of the simulated correlation matrices are computed and rank-ordered, and the eigenvalues of the input data correlation matrix are compared to the distributions of the eigenvalues at each position (largest eigenvalue, second-largest eigenvalue, and so on). When an eigenvalue of the input data correlation matrix exceeds the eigenvalue at some critical point (for example, the 95th percentile), this is understood to indicate the presence of a latent factor in the data. In all four parallel analyses in this example, the largest and second-largest eigenvalues of the input data correlation matrix exceed the simulated critical values. Thus all four parallel analyses suggest that two latent factors should be retained.

Figure 10.5 Parallel Analysis

Parallel Analysis (Criterion 1)

Component	Simulated Crit Val	Observed Eigenvalue
1	1.7988	**6.5474***
2	1.5809	**1.7727***
3	1.4326	1.0052
4	1.3112	0.7431
5	1.2084	0.6783
6	1.1153	0.4514
7	1.0308	0.3822
8	0.9510	0.2978
9	0.8732	0.2744
10	0.7988	0.2623
11	0.7234	0.2446
12	0.6482	0.1978
13	0.5652	0.1427

*** Retained Dimension (Obs > Crit)**

Parallel Analysis (Criterion 2)

Component	Simulated Crit Val	Observed Eigenvalue
1	1.7981	**6.5474***
2	1.5812	**1.7727***
3	1.4303	1.0052
4	1.3124	0.7431
5	1.2081	0.6783
6	1.1152	0.4514
7	1.0288	0.3822
8	0.9503	0.2978
9	0.8728	0.2744
10	0.7986	0.2623
11	0.7235	0.2446
12	0.6479	0.1978
13	0.5651	0.1427

*** Retained Dimension (Obs > Crit)**

Figure 10.5 *continued*

Parallel Analysis (Criterion 3)		
Component	Simulated Crit Val	Observed Eigenvalue
1	1.7599	**6.5474***
2	1.5528	**1.7727***
3	1.4106	1.0052
4	1.2929	0.7431
5	1.1922	0.6783
6	1.1016	0.4514
7	1.0168	0.3822
8	0.9372	0.2978
9	0.8601	0.2744
10	0.7842	0.2623
11	0.7095	0.2446
12	0.6328	0.1978
13	0.5499	0.1427
* Retained Dimension (Obs > Crit)		

Parallel Analysis (Criterion 4)		
Component	Simulated Crit Val	Observed Eigenvalue
1	1.7600	**6.5474***
2	1.5533	**1.7727***
3	1.4101	1.0052
4	1.2943	0.7431
5	1.1921	0.6783
6	1.1013	0.4514
7	1.0156	0.3822
8	0.9366	0.2978
9	0.8594	0.2744
10	0.7845	0.2623
11	0.7097	0.2446
12	0.6337	0.1978
13	0.5495	0.1427
* Retained Dimension (Obs > Crit)		

The minimum average partial correlation analysis (MAP) results are summarized in the "Average Partial Correlations Controlling Principal Components" table in Figure 10.6. In this example, analyses are performed that use both squared (MAP2) and fourth-powered (MAP4) partial correlations. These two analyses are summarized in separate columns in the table. The number of latent factors that are estimated by a minimum average partial correlation analysis is the number of principal components that, when partialed out of the correlation matrix, produce the smallest average squared or fourth-powered partial correlations. As shown in Figure 10.6, the minimum value is obtained for both the MAP2 and the MAP4 analysis when two principal components are partialed out.

Figure 10.6 Minimum Average Partial Correlation Analysis

Average Partial Correlations Controlling Principal Components		
Number of Components	**Squared**	**Fourth-Powered**
0	0.229076	0.069057
1	0.063655	0.011875
2	**0.036328***	**0.003287***
3	0.042970	0.004975
4	0.057102	0.009267
5	0.065326	0.014253
6	0.083345	0.022845
7	0.115706	0.037743
8	0.153423	0.050831
9	0.233063	0.120901
10	0.309145	0.167278
11	0.544163	0.416722
12	1.000000	1.000000
*** MAP = Minimum Values in Columns**		

Now that the number of factors is determined, it is time to extract and rotate a factor pattern that corresponds to a two-factor solution. For this analysis, the prior communality estimates are set to the squared multiple correlations. These are shown in Figure 10.7.

Figure 10.7 Prior Communalities

Prior Communality Estimates							
Communication Skills	**Problem Solving**	**Learning Ability**	**Judgment under Pressure**	**Observational Skills**	**Willingness to Confront Problems**	**Interest in People**	**Interpersonal Sensitivity**
0.62981394	0.58657431	0.61009871	0.63766021	0.67187583	0.64779805	0.75641519	0.75584891

Desire for Self-Improvement	**Appearance**	**Dependability**	**Physical Ability**	**Integrity**
0.57460176	0.45505304	0.63449045	0.42245324	0.68195454

Figure 10.8 displays the initial factor pattern matrix. The factor pattern matrix represents standardized regression coefficients for predicting the variables by using the extracted factors. Because the initial factors are uncorrelated, the pattern matrix is also equal to the correlations between variables and the common factors.

The pattern matrix suggests that Factor1 represents general ability. All loadings for Factor1 in the Factor Pattern are at least 0.5. Factor2 consists of high positive loadings on certain task-related skills (Willingness to Confront Problems, Observational Skills, and Learning Ability) and high negative loadings on some interpersonal skills (Interpersonal Sensitivity, Interest in People, and Integrity). This factor measures individuals' relative strength in these skills. Theoretically, individuals with high positive scores on this factor would exhibit better task-related skills than interpersonal skills. Individuals with high negative scores would exhibit better interpersonal skills than task-related skills. Individuals with scores near zero would have those skills balanced.

Figure 10.8 Initial Factor Pattern

Factor Pattern		
	Factor1	Factor2
Communication Skills	0.75441	0.07707
Problem Solving	0.68590	0.08026
Learning Ability	0.65904	0.34808
Judgment under Pressure	0.73391	-0.21405
Observational Skills	0.69039	0.45292
Willingness to Confront Problems	0.66458	0.47460
Interest in People	0.70770	-0.53427
Interpersonal Sensitivity	0.64668	-0.61284
Desire for Self-Improvement	0.73820	0.12506
Appearance	0.57188	0.20052
Dependability	0.79475	-0.04516
Physical Ability	0.51285	0.10251
Integrity	0.74906	-0.35091

Figure 10.9 displays the proportion of variance explained by each factor and the final communality estimates, including the total communality. The final communality estimates are the proportion of variance of the variables accounted for by the common factors. When the factors are orthogonal, the final communalities are calculated by taking the sum of squares of each row of the factor pattern matrix.

Figure 10.9 Initial Solution: Variance Explained and Final Communality Estimates

Variance Explained by Each Factor	
Factor1	Factor2
6.1776055	1.4622860

Final Communality Estimates: Total = 7.639892							
Communication Skills	Problem Solving	Learning Ability	Judgment under Pressure	Observational Skills	Willingness to Confront Problems	Interest in People	Interpersonal Sensitivity
0.57507775	0.47690128	0.55548854	0.58444301	0.68176986	0.66690409	0.78628588	0.79376521

Desire for Self-Improvement	Appearance	Dependability	Physical Ability	Integrity
0.56057957	0.36724994	0.63367286	0.27352712	0.68422640

Figure 10.10 displays the results of the varimax rotation of the two extracted factors and the corresponding orthogonal transformation matrix. The rotated factor pattern matrix is calculated by postmultiplying the original factor pattern matrix by the transformation matrix.

The rotated factor pattern matrix is somewhat easier to interpret. If a magnitude of at least 0.5 is required to indicate a salient variable-factor relationship, Factor1 now measures job enthusiasm and cognitive skills (Observational Skills, Willingness to Confront Problems, Learning Ability, Desire for Self-Improvement, Communication Skills, Problem Solving, Dependability, and Appearance) and Factor2 represents interpersonal skills (Interpersonal Sensitivity, Interest in People, Integrity, Judgment under Pressure, and Dependability).

Figure 10.10 Transformation Matrix and Rotated Factor Pattern

Orthogonal Transformation Matrix		
	1	**2**
1	0.75128	0.65999
2	0.65999	-0.75128

Rotated Factor Pattern		
	Factor1	**Factor2**
Communication Skills	0.61764	0.44000
Problem Solving	0.56827	0.39239
Learning Ability	0.72485	0.17346
Judgment under Pressure	0.41009	0.64519
Observational Skills	0.81759	0.11538
Willingness to Confront Problems	0.81251	0.08206
Interest in People	0.17906	0.86846
Interpersonal Sensitivity	0.08137	0.88721
Desire for Self-Improvement	0.63713	0.39325
Appearance	0.56198	0.22679
Dependability	0.56727	0.55846
Physical Ability	0.45295	0.26147
Integrity	0.33116	0.75800

To make the rotated factor pattern matrix even easier to interpret, you can use the REORDER option to rearrange the rows of the factor pattern. When you specify the REORDER option, variables whose highest absolute loading (reference structure loading for oblique rotations) is on the first factor are displayed first, from largest to smallest loading, followed by variables whose highest absolute loading is on the second factor, and so on. For an example of how to use the REORDER option, see "Example 10.2: Principal Factor Analysis" on page 355. You can also specify the FUZZ=0.5 option, which causes all the entries of the factor pattern smaller than 0.5 in absolute value to be printed as missing. For an example of how to use the FUZZ= option, see "Example 10.4: Extracting and Interpreting Solutions with Maximum Likelihood Factor Analysis" on page 366.

Figure 10.11 displays the variance explained by each factor and the final communality estimates after the orthogonal rotation. Even though the variances explained by the rotated factors are different from those of the unrotated factors (compare to Figure 10.9), the cumulative variance explained by the common factors remains the same. Note also that the final communalities for variables, as well as the total communality, remain unchanged after rotation. Although rotating a factor solution does not increase or decrease the statistical quality of the factor model, it can simplify the interpretations of the factors and redistribute the variance explained by the factors.

Figure 10.11 Rotated Solution: Variance Explained and Final Communality Estimates

Variance Explained by Each Factor	
Factor1	**Factor2**
4.1236793	3.5162122

Figure 10.11 *continued*

Final Communality Estimates: Total = 7.639892							
Communication Skills	Problem Solving	Learning Ability	Judgment under Pressure	Observational Skills	Willingness to Confront Problems	Interest in People	Interpersonal Sensitivity
0.57507775	0.47690128	0.55548854	0.58444301	0.68176986	0.66690409	0.78628588	0.79376521

Desire for Self-Improvement	Appearance	Dependability	Physical Ability	Integrity
0.56057957	0.36724994	0.63367286	0.27352712	0.68422640

Syntax: EFA Procedure

The following statements are available in the EFA procedure:

PROC EFA < *options* > ;
NFACTORS TYPE=*name* < *options* > ;
PRIORS *communalities* ;
VAR *variables* ;
FREQ *variable* ;
WEIGHT *variable* ;
BY *variables* ;
DISPLAY < *table-list* > < */options* > ;
DISPLAYOUT *table-spec-list* < */options* > ;

The PROC EFA statement is required. You can specify multiple NFACTORS statements. The following sections describe the PROC EFA statement and then describe the other statements in alphabetical order.

PROC EFA Statement

PROC EFA < *options* > ;

The PROC EFA statement invokes the procedure. Table 10.2 summarizes the available options in this statement. The options are described fully after the table.

Table 10.2 Options Available in the PROC EFA Statement

Option	Description
Data Set Option	
DATA=	Specifies the input data table
Factor Extraction and Communalities	
HEYWOOD	Sets to 1 any communality greater than 1
METHOD=	Specifies the estimation method
NFACTORS=	Specifies how to determine the number of factors

Table 10.2 *continued*

Option	Description
PRIORS=	Specifies the method of computing prior communality estimates
SEED=	Specifies the seed for pseudorandom number generation
ULTRAHEYWOOD	Allows communalities to exceed 1
Data Analysis	
NOBS=	Specifies the number of observations
VARDEF=	Specifies the divisor to use in calculating covariances or correlations
Rotation Method and Properties	
GAMMA=	Specifies the orthomax weight
NOPROMAXNORM	Turns off row normalization in promax rotation
NORM=	Specifies the row normalization method in rotation
POWER=	Specifies the power to be used in promax rotation
PREROTATE=	Specifies the prerotation method in promax rotation
RCONVERGE=	Specifies the convergence criterion for rotation cycles
RITER=	Specifies the maximum number of cycles for rotation
ROTATE=	Specifies the rotation method
TAU=	Specifies the oblimin weight
Control Display Output	
FUZZ=	Specifies the maximum absolute value to display as missing in the correlation and loading matrices
REFSTRUCT	Prints the reference structure and reference axis correlations
REORDER	Reorders the rows (variables) of various factor matrices
Numerical Properties	
CONVERGE=	Specifies the convergence criterion
MAXITER=	Specifies the maximum number of iterations
NTHREADS=	Specifies the maximum number of simultaneous computational threads to use
SINGULAR=	Specifies the singularity criterion

CONVERGE=*p*

CONV=*p*

specifies the convergence criterion for the METHOD=ALPHA, METHOD=ML, METHOD=PRINIT, or METHOD=ULS option. Iteration stops when the maximum change in the communalities is less than the value of the CONVERGE= option. By default, CONVERGE=0.001. Negative values are not allowed.

DATA=*libref.data-table*

names the input data table for PROC EFA to use. The default is the most recently created data table. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to the data, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 314.

data-table specifies the name of the input data table.

FUZZ=*p*

specifies how certain correlation and factor loading values are printed. When you specify this option, PROC EFA prints elements whose absolute values are less than *p* as missing. The exact values in any matrix can be obtained from the ODS output data sets. Negative values are not allowed.

GAMMA=*p*

specifies the orthomax weight to be used together with the option ROTATE=ORTHOMAX or PREROTATE=ORTHOMAX. Alternatively, you can specify ROTATE=ORTHOMAX(*p*), where *p* represents the orthomax weight. If you specify both GAMMA=*p* and ROTATE=ORTHOMAX(*p*), the ROTATE= option value takes precedence.

There is no restriction on valid values for the orthomax weight, although the most common values are between 0 and the number of variables. By default, GAMMA=1, which results in the varimax rotation. For more information, see the section "Simplicity Functions" on page 347.

HEYWOOD

HEY

sets to 1 any communality greater than 1, allowing iterations to proceed. See the section "Heywood Cases and Other Anomalies of Communality Estimates" on page 344 for a discussion of Heywood cases. This option has no effect if you specify METHOD=PRINCIPAL.

MAXITER=*n*

specifies the maximum number of iterations for factor extraction. You can use the MAXITER= option with the ALPHA, ML, PRINIT, or ULS method. By default, MAXITER=30.

METHOD=*name*

M=*name*

specifies the method of extracting factors. You can specify the following *names*:

ALPHA | A produces alpha factor analysis.

ML | M performs maximum likelihood factor analysis by using an algorithm due, except for minor details, to Fuller (1987). This option requires a nonsingular correlation matrix.

NONE specifies that factor extraction should not be performed. This is useful if you want to use one or more NFACTORS statements to investigate the number of latent factors that should be used to describe the data, but you do not want to extract those factors. You cannot specify this option if you use the NFACTORS=*n* option to specify that *n* factors should be extracted.

PRINCIPAL | PRIN | P performs a principal factor analysis unless you also specify PRIORS=ONE, in which case a principal component analysis is performed.

PRINIT performs an iterated principal factor analysis.

ULS | U performs an unweighted least squares factor analysis.

By default, METHOD=PRINCIPAL.

NFACTORS=*n* | *method*

NFACT=*n* | *method*

N=*n* | *method*

specifies how to determine the number of factors. If you specify a numeric value *n*, then *n* is a positive integer that represents the number of factors to be extracted. Alternatively, you can specify a *method* that is used to determine the number of factors. You can specify the following *methods*:

MAX sets the number of factors to the maximum number that is suggested by the criteria that you specify in the NFACTORS statements.

MEAN sets the number of factors to the mean of the numbers suggested by the criteria that you specify in the NFACTORS statements. If the computed mean is not an integer, the value is rounded to the nearest integer.

MEDIAN sets the number of factors to the median of the numbers suggested by the criteria that you specify in the NFACTORS statements. If the computed median is not an integer, the value is rounded to the nearest integer.

MIN sets the number of factors to the minimum number that is suggested by the criteria that you specify in the NFACTORS statements.

With PROC EFA, you can use one or more NFACTORS statements to specify criteria that are used to suggest numbers of factors to extract. You use the NFACTORS=*method* option in the PROC EFA statement to determine how to combine the numbers from these criteria to produce the final number of factors to extract. Any criteria for which you specify the STATUS=INACTIVE option in the PROC EFA statement are excluded from the calculation.

You must specify either the NFACTORS=*n* option or at least one NFACTORS statement. If you specify multiple NFACTORS statements but not the NFACTORS=*method* option, PROC EFA uses NFACTORS=MIN by default. If you specify both NFACTORS=*n* and one or more NFACTORS statements, the results for the NFACTORS statements are computed and displayed, but *n* factors are extracted.

NOBS=*n*

specifies the number of observations to use when computing significance tests. If the DATA= input data set is a raw data set, this number is defined by default to be the number of observations in the raw data set that are used for the analysis. The NOBS= option overrides this default definition. This option has no effect unless you specify METHOD=ML.

NOPROMAXNORM

NOPMAXNORM

turns off the default row normalization of the prerotated factor pattern, which is used to compute the promax target matrix.

NORM=*method*

specifies the method of normalizing the rows of the factor pattern for rotation. You can specify the following *methods*:

COV rescales the rows of the pattern matrix to represent covariances rather than correlations.

KAISER uses Kaiser's normalization. This means that the rows of the factor pattern matrix are normalized so that the sum of squares of each row is 1 ($\sum_j p_{ij}^2 = 1$).

NONE | RAW turns off row normalization.

WEIGHT weights the rows by the Cureton-Mulaik technique (Cureton and Mulaik 1975).

By default, NORM=KAISER.

NTHREADS=*n*

THREADS=*n*

specifies the number of threads to use in the computation, where *n* is an integer between 1 and 64, inclusive. The default value is the number of CPUs available in the machine.

POWER=*n*

specifies the power to use in computing the target pattern for the option ROTATE=PROMAX. Alternatively, you can specify ROTATE=PROMAX(*p*), where *p* represents the power. If you specify both POWER=*n* and ROTATE=PROMAX(*p*), the ROTATE= option value takes precedence.

Valid values must be integers greater than or equal to 1. By default, POWER=3.

PREROTATE=*name*

PRE=*name*

specifies the prerotation method for the option ROTATE=PROMAX. You can use any rotation method other than promax. See the ROTATE= option for a list of available methods.

By default, PREROTATE=VARIMAX.

PRIORS=*name*

specifies a method of computing prior communality estimates. You can specify the following *names*:

ASMC | A sets the prior communality estimates proportional to the squared multiple correlations but adjusted so their sum is equal to that of the maximum absolute correlations (Cureton 1968).

INPUT | I sets the prior communality estimates to the values that you specify in the PRIORS statement.

MAX | M sets the prior communality estimate for each variable to its maximum absolute correlation with any other variable.

ONE | O sets all prior communalities to 1.

RANDOM | R sets the prior communality estimates to pseudorandom numbers uniformly distributed between 0 and 1. You can use the SEED= option to specify the random number seed to use so that the analysis is repeatable.

SMC | S sets the prior communality estimate for each variable to its squared multiple correlation with all other variables.

By default, PRIORS=SMC.

RCONVERGE=*p*

RCONV=*p*

specifies the convergence criterion value *p* ($p > 0$) for rotation cycles. Rotation stops when the scaled change of the simplicity function value is less than *p*. Mathematically, the convergence criterion is

$$|f_{\text{new}} - f_{\text{old}}|/K < p$$

where f_{new} and f_{old} are simplicity function values of the current cycle and the previous cycle, respectively, and $K = \max(1, |f_{\text{old}}|)$ is a scaling factor. By default, RCONVERGE=1E–9.

REFSTRUCT

RS

prints the reference structure and reference axis correlations. This option has no effect unless you use the ROTATE= option to specify an oblique rotation.

REORDER

RE

reorders the rows (variables) of various factor matrices in the output. Variables whose highest absolute loading (reference structure loading for oblique rotations) is on the first factor are displayed first, from largest to smallest loading, followed by variables whose highest absolute loading is on the second factor, and so on.

RITER=*n*

specifies the maximum number of cycles for factor rotation. Except for promax and Procrustes, you can use this option with all rotation methods. The default *n* is either 10 times the number of variables or 100, whichever is greater.

ROTATE=*name*

R=*name*

specifies the rotation method. By default, ROTATE=NONE.

You can specify the following *name* to skip the factor rotation step:

NONE | N specifies that no rotation be performed, leaving the original orthogonal solution.

You can specify the following *names* for orthogonal rotations:

BIQUARTIMAX | BIQMAX specifies orthogonal biquartimax rotation. This corresponds to specifying ROTATE=ORTHOMAX(.5).

EQUAMAX | E specifies orthogonal equamax rotation. This corresponds to specifying ROTATE=ORTHOMAX with GAMMA=*number of factors*/2.

FACTORPARSIMAX | FPA specifies orthogonal factor parsimax rotation. This corresponds to specifying ROTATE=ORTHOMAX with GAMMA=*number of variables*.

ORTHCF(*p1*,*p2*) | **ORCF**(*p1*,*p2*) specifies the orthogonal Crawford-Ferguson rotation with the weights *p1* and *p2* for variable parsimony and factor parsimony, respectively. See the definitions of weights in the section "Simplicity Functions" on page 347.

ORTHGENCF(*p1*,*p2*,*p3*,*p4*) | **ORGENCF**(*p1*,*p2*,*p3*,*p4*) specifies the orthogonal generalized Crawford-Ferguson rotation with the four weights *p1*, *p2*, *p3*, and *p4*. See the definitions of weights in the section "Simplicity Functions" on page 347.

ORTHOMAX<(*p*)> | **ORMAX**<(*p*)> specifies the orthomax rotation with orthomax weight *p*. When ROTATE=ORTHOMAX, the default *p* value is 1 unless specified otherwise in the GAMMA= option. Alternatively, ROTATE=ORTHOMAX(*p*) specifies *p* as the orthomax weight or the GAMMA= value. See the definition of the orthomax weight in the section "Simplicity Functions" on page 347.

PARSIMAX | PA specifies orthogonal parsimax rotation. This corresponds to specifying ROTATE=ORTHOMAX with

$$\text{GAMMA} = \frac{nvar \times (nfact - 1)}{nvar + nfact - 2}$$

where *nvar* is the number of variables and *nfact* is the number of factors.

QUARTIMAX | QMAX | Q specifies orthogonal quartimax rotation. This corresponds to specifying ROTATE=ORTHOMAX(0).

VARIMAX | V specifies orthogonal varimax rotation. This corresponds to specifying ROTATE=ORTHOMAX with GAMMA=1.

You can specify the following *names* for oblique rotations:

BIQUARTIMIN | BIQMIN specifies biquartimin rotation. This corresponds to specifying ROTATE=OBLIMIN(.5) or ROTATE=OBLIMIN with TAU=0.5.

COVARIMIN | CVMIN specifies covarimin rotation. This corresponds to specifying ROTATE=OBLIMIN(1) or ROTATE=OBLIMIN with TAU=1.

OBBIQUARTIMAX | OBIQMAX specifies oblique biquartimax rotation.

OBEQUAMAX | OE specifies oblique equamax rotation.

OBFACTORPARSIMAX | OFPA specifies oblique factor parsimax rotation.

OBLICF(*p1*,*p2*) | **OBCF**(*p1*,*p2*) specifies the oblique Crawford-Ferguson rotation with the weights *p1* and *p2* for variable parsimony and factor parsimony, respectively. See the definitions of weights in the section "Simplicity Functions" on page 347.

OBLIGENCF(*p1*,*p2*,*p3*,*p4*) | **OBGENCF**(*p1*,*p2*,*p3*,*p4*) specifies the oblique generalized Crawford-Ferguson rotation with the four weights *p1*, *p2*, *p3*, and *p4*. See the definitions of weights in the section "Simplicity Functions" on page 347.

OBLIMIN<(*p*)> | **OBMIN**<(*p*)> specifies the oblimin rotation with oblimin weight *p*. When ROTATE=OBLIMIN, the default *p* value is 0 unless specified otherwise in the TAU= option. Alternatively, ROTATE=OBLIMIN(*p*) specifies *p* as the oblimin weight or the TAU= value. See the definition of the oblimin weight in the section "Simplicity Functions" on page 347.

OBPARSIMAX | OPA specifies oblique parsimax rotation.

OBQUARTIMAX | OQMAX specifies oblique quartimax rotation. This is the same as the quartimin method.

OBVARIMAX | OV specifies oblique varimax rotation.

PROMAX<(*n*)> | P<(*n*)> specifies oblique promax rotation. You can use the PREROTATE= option to specify the desirable prerotation method, either orthogonal or oblique. When used with ROTATE=PROMAX, the POWER= option enables you to specify the power for forming the target. You can also use ROTATE=PROMAX(*n*), where *n* is the POWER= option value.

QUARTIMIN | QMIN specifies quartimin rotation. This is the same as the oblique quartimax method. It also corresponds to specifying ROTATE=OBLIMIN(0) or ROTATE=OBLIMIN with TAU=0.

SEED=*n*

RANDOM=*n*

specifies the initial seed for the pseudorandom number generator for use with the option PRIORS=RANDOM. The value of the SEED= option must be an integer. If you omit the SEED= option or if the specified value is negative or 0, the time of day from the computer's clock is used to obtain the initial seed.

SINGULAR=*p*

SING=*p*

specifies the singularity criterion, where $0 < p < 1$. By default, SINGULAR=1E–8.

TAU=*p*

specifies the oblimin weight to use with the option ROTATE=OBLIMIN or PREROTATE=OBLIMIN. Alternatively, you can use ROTATE=OBLIMIN(*p*), where *p* represents the oblimin weight. If you specify both TAU=*p* and ROTATE=OBLIMIN(*p*), the ROTATE= option value takes precedence.

There is no restriction on valid values for the oblimin weight, although for practical purposes a negative or zero value is recommended. By default TAU=0, which results in the quartimin rotation. See the section "Simplicity Functions" on page 347 for more information.

ULTRAHEYWOOD

ULTRA

allows communalities to exceed 1. This option can cause convergence problems, because communalities can become extremely large and ill-conditioned Hessians might occur. See the section "Heywood Cases and Other Anomalies of Communality Estimates" on page 344 for a discussion of Heywood cases. This option has no effect if you specify METHOD=PRINCIPAL.

VARDEF=DF | N | WDF | WEIGHT | WGT

specifies the divisor to use for calculating variances and covariances. By default, VARDEF=DF. The values and associated divisors are displayed in the following table.

Value	Description	Divisor
DF	Degrees of freedom	$n - 1$
N	Number of observations	n
WDF	Sum of weights DF	$\sum_i w_i - 1$
WEIGHT | WGT	Sum of weights	$\sum_i w_i$

BY Statement

BY *variables* **;**

You can specify a BY statement in PROC EFA to obtain separate analyses of observations in groups that are defined by the values of the BY variables. If you specify more than one BY statement, only the last one specified is used. For more information, see the discussion of BY-group processing in the "Grouping Data" section of *SAS Programmers Guide: Essentials*.

DISPLAY Statement

DISPLAY < *table-list* > < / *options* > **;**

The DISPLAY statement enables you to specify a list of display tables to display or exclude. This statement is similar to the ODS SELECT, ODS EXCLUDE, and ODS TRACE statements. However, the DISPLAY statement can improve performance when a large number of tables could be generated (such as in BY-group processing). The procedure processes the DISPLAY statement on a CAS server and thus sends only a subset of ODS tables to the SAS client. Because ODS statements are processed on a SAS client, first all the generated display tables are sent to the client, and then the client creates a subset.

If you use both DISPLAY and ODS statements together, the DISPLAY statement takes precedence over the ODS statements. Note that the ODS EXCLUDE statement processes tables that are sent to the client after they have been filtered by the DISPLAY statement. In some cases, it might appear that the ODS EXCLUDE statement is taking precedence because it can further filter the tables. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

You can specify the *table-list* as a list of table names, paths, partial pathnames, and regular expressions.

The table names that you can specify are listed in the section "ODS Table Names" on page 349. A path is a table name that is prefixed with dot-separated grouping information. For example, a SelectionSummary table that a procedure produces during a selection routine might have the path *Bygroup1.Summary.SelectionSummary*. A partial pathname does not include all groups; for example, *SelectionSummary* and *Summary.SelectionSummary* are partial pathnames for *Bygroup1.Summary.SelectionSummary*.

When you specify a table name or partial pathname, all display tables whose paths end in the specified name are selected for display or exclusion. For example, both *SelectionSummary* and *Summary.SelectionSummary* select *Bygroup1.Summary.SelectionSummary*.

A regular expression is enclosed in forward slashes (/). For example, specifying "/tions/" selects all pathnames that contain the substring "tions"; in particular, the *Bygroup1.Summary.SelectionSummary* table is selected. Specifying "!/tions/" selects all pathnames that do not contain the substring "tions"; in particular, the *Bygroup1.Summary.SelectionSummary* table is not selected.

You can specify the following *options* after a slash (/):

CASESENSITIVE
performs a case-sensitive comparison of table names in the *table-list* to display table names when tables are subsetted for display. To preserve case, you must enclose table names in the *table-list* in quotation marks.

EXCLUDE
displays all display tables except those that you specify in the *table-list*.

EXCLUDEALL
suppresses display of all tables. This option takes precedence over the other options.

TRACE
displays the display table names, labels, and paths.

DISPLAYOUT Statement

DISPLAYOUT *table-spec-list* < / *options* > **;**

The DISPLAYOUT statement enables you to create output tables from your displayed output. This statement is similar to the ODS OUTPUT statement. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

The *table-spec-list* specifies a list of output tables to create. Each entry in the list has either a *key*=*value* format or a *key* format:

key=*value*	specifies *key* as the ODS table name, path, or partial pathname, and specifies *value* as the output table name.
key	specifies *key* as the ODS table name and also as the output table name.

The ODS table names that you can specify are listed in the section "ODS Table Names" on page 349. You cannot specify the ODS table named OutputCasTables in the *table-spec-list*.

Table names and partial pathnames are discussed under the DISPLAY statement. The DISPLAYOUT statement does not support regular expressions.

You can specify the following *options* after a slash (/):

INCLUDEALL
creates output tables for all display tables. The name of the created output table is the same as the name of the corresponding display table. If you specify this option, the *table-spec-list* specification is ignored.

NOREPLACE
does not replace any existing output table of the same name.

REPEATED
replicates all output tables on all nodes.

FREQ Statement

FREQ *variable* **;**

The *variable* in the FREQ statement identifies a numeric variable in the input data table that contains the frequency of occurrence of each observation. PROC EFA treats each observation as if it appears f times, where f is the value of the FREQ *variable* for the observation. If f is not an integer, it is truncated to an integer. If f is less than 1 or missing, the observation is not used in the analysis. When you omit the FREQ statement, each observation is assigned a frequency of 1.

NFACTORS Statement

NFACTORS TYPE=*name* < *options* > **;**

The NFACTORS statement specifies a criterion to use for suggesting the number of factors to extract in an analysis. If you use the NFACTORS=*n* option in the PROC EFA statement to specify the number of factors to extract, then the NFACTORS statement is not required. In this case, you can still use the NFACTORS statement to investigate alternative criteria for the number of factors or to assess the robustness of your analysis. If you do not specify the NFACTORS=*n* option in the PROC EFA statement, then you must specify at least one NFACTORS statement.

You must specify the following option:

TYPE=*name*
specifies the criterion to use for suggesting the number of factors to extract. You can specify the following *names*:

EIGENVALUE determines the number of factors to extract by comparing the eigenvalues of the reduced correlation matrix to a threshold value. The threshold value is specified by the THRESHOLD= option. If you omit the THRESHOLD= option, the default value is 1. Each eigenvalue greater than the threshold value indicates the presence of a common factor.

MAP | MAP2 performs a minimum average partial (MAP) correlation analysis (Velicer 1976) to determine the number of factors to extract. This criterion selects the number of factors that corresponds to the number of principal components that are partialed out to yield the smallest average squared partial (residual) correlations among variables.

MAP4 performs a minimum average partial (MAP) correlation analysis similar to that performed by the TYPE=MAP option, but using the smallest average fourth-powered partial (residual) correlations among variables (Velicer, Eaton, and Fava 2000).

PARALLEL performs a parallel analysis as described by Glorfeld (1995) and Horn (1965). This criterion selects the number of factors that corresponds to the first n consecutive eigenvalues of the sample correlation matrix that are significantly greater than the

corresponding eigenvalues of random correlation matrices that are generated from a multivariate standard normal distribution.

To compute a parallel analysis, PROC EFA simulates a large number of random correlation matrices and constructs an empirical distribution for each positional eigenvalue: largest, second-largest, and so on (Glorfeld 1995). You can use the NSIMULATIONS= option to specify the number of simulations to use. If you omit the NSIMULATIONS= option, PROC EFA uses a default value of 10,000. A factor is suggested if an observed eigenvalue is greater than the critical value at a specified one-sided alpha level, with reference to the corresponding simulated distribution of random eigenvalues. You can use the ALPHA= option to specify the alpha level. If you omit the ALPHA= option, PROC EFA uses a default value of 0.05. As soon as an observed eigenvalue is less than or equal to the corresponding critical value, no more factors are counted. In other words, only the first *n* consecutive significant eigenvalues are counted for the number of factors.

Note that the parallel analysis criterion is based on the standard correlation matrix, not the reduced correlation matrix. The choice of prior communalities does not affect the parallel analysis.

PROPORTION determines the number of factors to extract by determining the smallest number of factors such that the cumulative proportion of common variance explained exceeds a threshold value. The threshold value is specified by the THRESHOLD= option. If you omit the THRESHOLD= option, the default value is 1.

For this criterion, the eigenvalues are sorted in descending order, and the proportion of the *common* variance explained by each eigenvalue is computed. Note that the denominator (that is, the total common variance) in computing the proportion here is the sum of the initial communality estimates and not the total variances of variables (which would be the same as the number of variables with a correlation input) unless all prior communality estimates are ones.

The TYPE=EIGENVALUE and TYPE=PROPORTION criteria refer to properties of the reduced correlation matrix. The reduced correlation matrix is the correlation matrix for the analysis variables, but where the diagonal entries of the matrix are replaced by estimates of the communalities for the corresponding variables. For more information about specifying communalities, see the section "Prior Communalities" on page 341.

You can also specify the following additional *options*:

ALPHA=*p*

specifies the one-sided alpha level for computing the critical value at the upper end of the simulated distribution of eigenvalues in a parallel analysis. The value of *p* must be between 0 and 1. By default, ALPHA=0.05. This option has an effect only when you specify TYPE=PARALLEL in the NFACTORS statement.

NSIMULATIONS=*n*

NSIMS=(*n***)**

NSIM=(*n***)**

specifies the number of simulations to use in constructing an empirical distribution of eigenvalues

for a parallel analysis. The value of *n* must be greater than or equal to 200. By default, NSIMULATIONS=10000. This option has an effect only when you specify TYPE=PARALLEL in the NFACTORS statement.

SEED=*n*

specifies the initial seed for the pseudorandom number generator that is used to simulate correlation matrices for a parallel analysis. The value of the SEED= option must be an integer. If you omit the SEED= option or if the specified value is negative or 0, the time of day from the computer's clock is used to obtain the initial seed. This option has an effect only when you specify TYPE=PARALLEL in the NFACTORS statement.

STATUS=ACTIVE | INACTIVE

specifies the status of the criterion for determining the number of factors to extract. By default, STATUS=ACTIVE. A criterion that has an active status is used to determine the final number of factors to extract. A criterion that has an inactive status is computed for informational purposes only and is not used to determine the final number of factors. The final number of factors to extract is computed by combining the numbers of factors that is suggested by all the criteria whose status is active. The numbers are combined according to the value of the NFACTORS= statement in the PROC EFA statement.

THRESHOLD=*p*

specifies the threshold value for either a TYPE=EIGENVALUE or TYPE=PROPORTION criterion. For a TYPE=EIGENVALUE criterion, each eigenvalue greater than the threshold value indicates the presence of a common factor. For a TYPE=PROPORTION criterion, the number of factors that is suggested is the smallest number of eigenvalues that is required so that the proportion of common variance explained exceeds the threshold.

PRIORS Statement

PRIORS *communalities* **;**

The PRIORS statement specifies numeric values between 0.0 and 1.0 for the prior communality estimates for each variable. The first numeric value corresponds to the first variable in the VAR statement, the second value to the second variable, and so on. The number of numeric values must equal the number of variables. For example:

```
proc efa nfactors=1;
   var     x  y  z;
   priors .7 .8 .9;
run;
```

You can specify various methods of computing prior communality estimates by using the PRIORS= option in the PROC EFA statement. See the description of that option for more information about the default prior communality estimates.

VAR Statement

VAR *variables* **;**

The VAR statement specifies the numeric variables to be analyzed. If you omit this statement, all numeric variables that are not specified in other statements are analyzed.

WEIGHT Statement

WEIGHT *variable* **;**

The *variable* that you specify in the WEIGHT statement is used as a weight to perform a weighted analysis of the data. Observations that have nonpositive weights or missing weights are excluded from the analysis. The values of the WEIGHT variable can be nonintegral and are not truncated. When you omit this statement, all observations that are used in the analysis are assigned a weight of 1.

Details: EFA Procedure

The EFA procedure implements methods and algorithms that enable you first to determine the number of common factors to retain in a common factor model and then to extract common factors and rotate the corresponding factor pattern.

When you provide multivariate data as input, PROC EFA first computes the correlation matrix for the analysis variables. It is this correlation matrix that is the object of analysis.

The Common Factor Model

In the common factor model, factors are hypothesized to account for the observed correlations among variables in the data. A *common factor* is an unobserved (or latent) hypothetical variable that contributes to the variance of at least two of the observed variables. The unqualified term "factor" often refers to a common factor. A *unique factor* is an unobservable, hypothetical variable that contributes to the variance of only one of the observed variables. The model for common factor analysis posits one unique factor for each observed variable.

The equation for the common factor model is

$$y_{ij} = x_{i1}b_{1j} + x_{i2}b_{2j} + \cdots + x_{iq}b_{qj} + e_{ij}$$

where

y_{ij} is the value of the ith observation on the jth observed variable.

x_{ik} is the value of the ith observation on the kth common factor; or in the regression analysis context, a factor is an unobserved common predictor of more than one observed variable.

b_{kj} is the factor loading of the jth variable on the kth common factor; or in the regression analysis context, b_{kj} is the regression coefficient of the kth common factor for predicting the jth variable.

e_{ij} is the value of the *i*th observation on the *j*th unique factor; or in the regression analysis context, a unique factor plays the role of the residual in a model equation.

q is the number of common factors, which might not be fixed before the analysis.

Typically, the maximum number of factors that can be extracted is the same as the number of variables that are being factor-analyzed. However, it is likely that the actual number of factors that you want to retain for the extraction is much smaller than the number of variables. In this sense, factor analysis can be viewed as a data reduction technique.

For convenience, it is assumed that all variables have a mean of 0. In matrix terms, these equations are represented as

$$\mathbf{Y} = \mathbf{XB} + \mathbf{E}$$

where $\mathbf{X}$ is the matrix of unobserved factor scores and $\mathbf{B}'$ is the factor pattern (or factor loading) matrix.

The model makes two important assumptions:

- The unique factors are uncorrelated with each other.
- The unique factors are uncorrelated with the common factors.

When the factors are initially extracted, it is also assumed, for convenience, that the common factors are uncorrelated with each other and have unit variance. In this case, the common factor model implies that the covariance s_{jk} between the *j*th and *k*th variables, $j \neq k$, is given by

$$s_{jk} = b_{1j}b_{1k} + b_{2j}b_{2k} + \cdots + b_{qj}b_{qk}$$

or

$$\mathbf{S} = \mathbf{B}'\mathbf{B} + \mathbf{U}^2$$

where $\mathbf{S}$ is the covariance matrix of the observed variables and $\mathbf{U}^2$ is the diagonal covariance matrix of the unique factors.

If the original variables are standardized to unit variance, the preceding formula yields correlations instead of covariances. It is in this sense that common factors explain the correlations among the observed variables. In considering the diagonal elements of standardized $\mathbf{S}$, the variance of the *j*th variable is expressed as

$$s_{jj} = 1 = b_{1j}^2 + b_{2j}^2 + \cdots + b_{qj}^2 + [\mathbf{U}^2]_{jj}$$

where $b_{1j}^2 + b_{2j}^2 + \cdots + b_{qj}^2$ and $[\mathbf{U}^2]_{jj}$ are the *communality* and *uniqueness*, respectively, of the *j*th variable. The communality represents the extent of the overlap with the common factors. In other words, it is the proportion of variance that is accounted for by the common factors.

The difference between the correlation that is predicted by the common factor model and the actual correlation is the *residual correlation*. A good way to assess the goodness of fit of the common factor model is to examine the residual correlations.

The common factor model implies that the partial correlations among the variables, if you remove the effects of the common factors, must all be 0. When you remove the common factors, only unique factors remain; they are by definition uncorrelated.

The assumptions of common factor analysis imply that the common factors are, in general, not linear combinations of the observed variables. In fact, even if the data contain measurements on the entire population of observations, you cannot compute the scores of the observations on the common factors. Although you cannot compute the common factor scores directly, you can estimate them in a variety of ways.

The Exploratory Nature of Common Factor Analysis

In practice, you conduct a common factor analysis so that you can find the smallest number of common factors that can account for the observed correlations or covariances among variables in the data—that is, so that you can minimize the residual correlations. In addition, you are interested in understanding the meanings of the common factors.

For the sample covariance matrix or correlation matrix, a common factor analysis consists of the following four distinct but related tasks:

- determining the number of factors
- extracting factors
- rotating factors
- estimating factor scores

If raw data are provided as input, the sample covariance or correlation matrix of the observed variables is computed from the raw data for analysis.

Common factor analysis in practical applications is often described as *exploratory* (hence the alternative name "exploratory factor analysis"), because you might not start with a fixed number of factors to extract or a fixed coordinate system to interpret the factors. Instead, you might explore different *rotated* factor solutions that use different numbers of factors so that you can choose the optimal solution to interpret the extracted factors. Most of the time, such an optimal solution is also the one that obeys the simplicity principle: it uses the lowest possible number of factors to account for the correlations among observed variables while maintaining the interpretability of common factors with a *simple* rotated factor pattern (in the sense that each column of the rotated factor pattern matrix contains mostly zero loadings).

In fact, your substantive knowledge in a specific domain also plays a critical role in choosing an optimal factor solution in applications. Although PROC EFA cannot replace your domain knowledge, it provides objective analytic tools to facilitate the exploratory process in common factor analysis.

Factor Analysis in Practice

This section provides more details about how you establish or estimate a factor model through the four tasks described in the section "The Exploratory Nature of Common Factor Analysis" on page 339, which emphasizes the exploratory nature of factor analysis in applications.

Determining the Number of Factors

In exploratory factor analysis, usually the first task is to determine the number of factors that suitably account for the correlations in the multivariate data. If you have domain knowledge or a theoretical position about the underlying factors, you can fix the number of factors at this stage for the subsequent tasks of factor analysis. Otherwise, you need to determine the number of factors on the basis of the eigenvalues of the sample correlation matrix or the *reduced* correlation matrix. Whereas the sample correlation matrix contains ones as the diagonal values, a reduced correlation matrix is computed by replacing the diagonal values with *communality estimates* for the variables. A popular choice in common factor analysis is to use the squared multiple correlations (SMCs), which are less than one in most practical situations, as the initial communality estimates. Essentially, the different methods of determining the number of common factors are all based on evaluating the eigenvalues (or their cumulative sums) of the sample or reduced correlation matrix.

Extracting Factors

After you determine the number of factors or a set of plausible numbers of factors, you are ready to extract the common factors. In essence, this means that you estimate the factor pattern matrix $\mathbf{B}'$, given the data and the number of factors. Iterative and noniterative methods have been proposed for this purpose.

Factor extraction as described here is treated as an initial stage of estimation, in which the common factors are uncorrelated with each other. After you extract or estimate the factors, you want to interpret them. This entails assigning to each common factor a name that reflects the *salience* of the factor in predicting each of the observed variables—that is, the salience of the coefficients (or loadings) in the pattern matrix that corresponds to the factor. Factor interpretation is a subjective process. You can sometimes make it less subjective by *rotating* the common factors—in other words, by applying a nonsingular linear transformation to the factor pattern matrix.

Rotating Factors

Rotating a factor pattern is conceptually the same as finding a (suitable) coordinate system for locating and interpreting the extracted factors. The main goal of factor rotation is to find a rotated pattern matrix in which all the coefficients are close to 0 or ± 1. Such a pattern is easier to interpret than a pattern that has many intermediate elements (such as those commonly observed in the initial factor pattern). Therefore, most rotation methods attempt to optimize a simplicity function of the rotated pattern matrix that measures, in some sense, how close the elements are to 0 or ± 1.

If the factors are rotated by an *orthogonal transformation,* the rotated factors are still uncorrelated. If the factors are rotated by an *oblique transformation,* the rotated factors might become correlated. Oblique rotations often produce more useful or interpretable patterns than orthogonal rotations. However, a consequence of correlated factors is that there is no single unambiguous measure of the importance of a factor in explaining a variable. Thus, for oblique rotations, the pattern matrix does not provide all the necessary information for interpreting the factors; you might want to examine the *factor structure* and the *reference structure* to understand the correlations between variables and factors.

Rotating a set of factors does not change the statistical explanatory power of the factors. You cannot say that any rotation is better than any other rotation from a statistical point of view; all rotations, whether orthogonal or oblique, are equally good statistically. Therefore, the choice among different rotations must be based on nonstatistical grounds. For most applications, the preferred rotation is the one that is most easily interpretable or most compatible with substantive theories.

Estimating Factor Scores

In the factor analysis model, the factor scores x_{ik} are not observed. However, after you obtain a factor solution, you can estimate the factor scores by using the parameter estimates (for example, the estimated factor loadings) and the raw data. Estimated factor scores are useful in subsequent statistical analyses, where they can serve as predictors of other outcomes or as classification variables themselves. PROC EFA does not support the computation of factor scores.

Prior Communalities

PROC EFA supports the following methods of specifying or computing prior communality estimates:

- squared multiple correlations (SMC)
- adjusted squared multiple correlations (ASMC)
- maximum absolute correlations
- unit values
- random values
- user-specified values

For SMC priors, the prior communality of each variable is set to the squared multiple correlation of that variable with all other variables. This is the default choice of prior communalities in PROC EFA.

For ASMC priors, the prior communality of each variable is proportional to the squared multiple correlation of that variable with all other variables, but it is adjusted so that the sum of the communalities is equal to the sum of the maximum absolute correlations (Cureton 1968).

For maximum absolute correlations, the prior communality of each variable is equal to its maximum absolute correlation with any other analysis variable. If the correlation matrix for the analysis variables is singular, you should use maximum absolute correlations for prior communalities and not use SMC or ASMC priors.

If you specify unit priors, a prior communality of 1 is assigned to each variable. If you specify random priors, prior communalities are set to pseudorandom numbers that are uniformly distributed between 0 and 1. Finally, you can specify a list of numeric values between 0 and 1 to use as the prior communality estimates for each variable. In this case, the first numeric value corresponds to the first analysis variable, the second value corresponds to the second variable, and so on. The number of numeric values must equal the number of analysis variables.

Determining the Number of Factors

PROC EFA enables you to specify multiple criteria for determining the number of factors. You can also specify the same type of criterion but with different option values. PROC EFA supports the following criteria for determining the number of factors to retain:

- minimum eigenvalue
- proportion of variance explained
- parallel analysis
- minimum average partial correlation analysis

The first two criteria refer to properties of the reduced correlation matrix. The reduced correlation matrix is the correlation matrix for the analysis variables, but in which the diagonal entries of the matrix are replaced by estimates of the communalities for the corresponding variables. For more information about specifying communalities, see the section "Prior Communalities" on page 341.

The minimum eigenvalue criterion determines the number of factors to retain by comparing the eigenvalues of the reduced correlation matrix to a threshold value that you specify. Each eigenvalue that is greater than the threshold value indicates the presence of a common factor. If you do not specify a threshold value, PROC EFA uses a default value of 1; this is a very common practice.

The proportion of variance explained criterion also uses the eigenvalues of the reduced correlation matrix. For this criterion, the eigenvalues are sorted in descending order, and the proportion of the *common* variance that is explained by each eigenvalue is computed. Note that the denominator (that is, the total common variance) in computing the proportion here is the sum of the initial communality estimates and not the total variances of variables (which would be the same as the number of variables with a correlation input) unless all prior communality estimates are ones. The number of factors to retain is estimated as the smallest number of factors such that the cumulative variance explained exceeds a threshold value. If you do not specify a threshold value, PROC EFA uses a default value of 1.

A parallel analysis, first proposed by Horn (1965), selects the number of factors by comparing the eigenvalues in the sample correlation matrix to the eigenvalues of random correlation matrices that are generated from a multivariate standard normal distribution. PROC EFA simulates a large number of random correlation matrices and constructs an empirical distribution for each positional eigenvalue: largest, second-largest, and so on (Glorfeld 1995). If you do not specify the number of simulations to use, PROC EFA uses a default value of 10,000. A factor is suggested if an observed eigenvalue is greater than the critical value at a specified one-sided alpha level, with reference to the corresponding simulated distribution of random eigenvalues. If you do not specify an alpha level, PROC EFA uses a default value of 0.05. As soon as an observed eigenvalue is less than or equal to the corresponding critical value, no more factors are counted. In other words, only the first n consecutive significant eigenvalues are counted for the number of factors.

Note that the parallel analysis criterion is based on the standard correlation matrix, not the reduced correlation matrix. Therefore, the choice of prior communalities does not affect the parallel analysis.

A minimum average partial correlation analysis first performs a principal component analysis on the sample correlation matrix. Next, it computes averages of the squared or fourth-powered partial correlations between the variables after recursively controlling for one component, two components, and so on (Velicer 1976). The

selected number of factors is the number of controlled components that correspond to the smallest average squared or fourth-powered partial correlation.

Note that the minimum average partial correlation analysis criterion is based on the standard correlation matrix, not the reduced correlation matrix. Therefore, the choice of prior communality estimates does not affect the minimum average partial correlation analysis.

Each criterion that you specify is used by PROC EFA to suggest a number of factors to retain. The criteria are independent of one another, but you cannot specify two identical criteria. By default, the final number of factors that PROC EFA suggests is the minimum of the values suggested by the criteria that you specify. You can change this default behavior by specifying the NFACTORS=*method* option, where *method* is MIN, MAX, MEDIAN, or MEAN. You can also use the STATUS=INACTIVE option to specify criteria whose status is inactive. A criterion with an inactive status is computed for informational purposes only and is not used to determine the final number of factors.

Factor Extraction

After you determine the number of common factors to use for an analysis, it is time to extract those factors. That is, you must use the data and the number of factors to estimate the factor pattern matrix $\mathbf{B}'$. (For more information about the factor pattern and its relation to the original data, see the section "The Common Factor Model" on page 337.)

PROC EFA implements the following factor extraction methods:

- principal factor analysis
- iterated principal factor analysis
- unweighted least squares factor analysis
- maximum likelihood (canonical) factor analysis
- alpha factor analysis

The simplest and computationally most efficient method of common factor analysis is principal factor analysis. In a principal factor analysis, you construct the reduced correlation matrix by using the prior communalities, and the factor pattern is computed from the eigenvalues and eigenvectors of the reduced correlation matrix. Note that when the prior communalities are set to ones, a principal factor analysis is equivalent to a principal component analysis.

The remaining four methods use iterative algorithms to compute both the factor pattern and updated estimates of the communalities.

Maximum Likelihood Factor Analysis

Although principal factor analysis is perhaps the most commonly used method of common factor analysis, most statisticians prefer maximum likelihood (ML) factor analysis (Lawley and Maxwell 1971). The ML method of estimation has desirable asymptotic properties (Bickel and Doksum 1977) and produces better estimates than principal factor analysis in large samples. The unrotated ML solution is equivalent to Rao's canonical factor solution (Rao 1955) and Howe's solution that maximizes the determinant of the partial

correlation matrix (Morrison 1976). Thus, as a descriptive method, ML factor analysis does not require a multivariate normal distribution.

With the ML method, you can test hypotheses about the number of common factors. In particular, tests of the following null hypotheses are displayed by default when you specify this method:

- H_0: No common factors
- H_0: Factors retained are sufficient to explain the correlations

The test for no common factors is equivalent to Bartlett's test of sphericity. The variables should have an approximate multivariate normal distribution for the probability levels to be valid. Lawley and Maxwell (1971) suggest that the number of observations should exceed the number of variables by 50 or more, although Geweke and Singleton (1980) claim that as few as 10 observations are adequate with five variables and one common factor. Certain regularity conditions must also be satisfied for Bartlett's χ^2 test to be valid (Geweke and Singleton 1980), but in practice these conditions are usually satisfied.

Other quantities that are computed when you specify the ML method include the following:

- Akaike's information criterion (AIC) (Akaike 1973, 1974, 1987). AIC is a general criterion for estimating the best number of parameters to include in a model when you use ML estimation. The number of factors that yields the smallest value of AIC is considered best. Like the χ^2 test, AIC tends to include factors that are statistically significant but inconsequential for practical purposes.
- Schwarz's Bayesian criterion (SBC) (Schwarz 1978). SBC is another criterion, similar to AIC, for determining the best number of parameters. The number of factors that yields the smallest value of SBC is considered best. SBC seems to be less inclined to include trivial factors than either AIC or the chi-square test.
- Tucker and Lewis's reliability coefficient (Tucker and Lewis 1973)
- Squared canonical correlations. These are the same as the squared multiple correlations for predicting each factor from the variables.

You cannot use the ML method with a singular correlation matrix, and it is especially prone to Heywood cases. (See the section "Heywood Cases and Other Anomalies of Communality Estimates" for a discussion of Heywood cases.) If you have problems with ML, the best alternative is to use unweighted least squares factor analysis.

Heywood Cases and Other Anomalies of Communality Estimates

Because communalities are squared correlations, you would always expect them to lie between 0 and 1. However, it is a mathematical peculiarity of the common factor model that final communality estimates might exceed 1. If a communality equals 1, the situation is referred to as a Heywood case; if a communality exceeds 1, it is an ultra-Heywood case. An ultra-Heywood case implies that some unique factor has negative variance, which is a clear indication that something is wrong. Possible causes include the following:

- bad prior communality estimates

- too many or too few common factors
- not enough data to provide stable estimates
- the common factor model is not an appropriate model for the data

An ultra-Heywood case invalidates a factor solution. Factor analysts disagree about whether a factor solution with a Heywood case can be considered legitimate.

For the iterative extraction methods, PROC EFA stops iterating and returns an error if an estimated communality exceeds 1. To allow processing to continue when a Heywood or ultra-Heywood case is encountered, you can specify the HEYWOOD option to set the upper bound of any communality to 1, or you can specify the ULTRAHEYWOOD option to allow communalities to exceed 1.

Theoretically, the communality of a variable should not exceed its reliability. Violation of this condition is called a quasi-Heywood case and should be regarded with the same suspicion as an ultra-Heywood case.

Elements of the factor structure and reference structure matrices can exceed 1 only in the presence of an ultra-Heywood case. On the other hand, an element of the factor pattern might exceed 1 in an oblique rotation, irrespective of the occurrence of a Heywood case.

The maximum likelihood method is especially susceptible to quasi- or ultra-Heywood cases. During the iteration process, a variable that has high communality is given a high weight; this tends to increase its communality, which increases its weight, and so on.

It is often stated that the squared multiple correlation of a variable with the other variables is a lower bound to its communality. This is true if the common factor model fits the data perfectly, but it is not generally the case for real data. A final communality estimate that is less than the squared multiple correlation can, therefore, indicate poor fit, possibly because there are not enough factors. It is by no means as serious a problem as an ultra-Heywood case. Factor methods that use the Newton-Raphson method, such as the ML method, can actually produce communalities less than 0, which is a result even more disastrous than an ultra-Heywood case.

The squared multiple correlation of a factor with the variables might exceed 1, even in the absence of ultra-Heywood cases. This situation is also cause for alarm. Alpha factor analysis seems to be especially prone to this problem, but it does not occur in maximum likelihood estimation. If a squared multiple correlation is negative, too many factors were retained.

For data that do not fit the common factor model perfectly, you can expect some of the eigenvalues to be negative. If an iterative factor method converges properly, the sum of the eigenvalues that correspond to rejected factors should be 0; hence, some eigenvalues are positive and some are negative. If a principal factor analysis fails to yield any negative eigenvalues, the prior communality estimates are probably too large. Negative eigenvalues cause the cumulative proportion of variance explained to exceed 1 for a sufficiently large number of factors. The cumulative proportion of variance explained by the retained factors should be approximately 1 for principal factor analysis and should converge to 1 for iterative methods. Occasionally, a single factor can explain more than 100% of the common variance in a principal factor analysis, indicating that the prior communality estimates are too low.

If a squared canonical correlation or a coefficient alpha is negative, too many factors were retained.

Rotating Factors

The purpose of factor rotation is to facilitate the interpretation of the common factors that were computed during the extraction stage. When a common factor solution is interpreted, each factor is assigned a name that reflects its importance in predicting each of the observed variables. This process is made easier (or less subjective) if all the coefficients in the factor pattern are close to 0 or ± 1. Thus, the goal of a factor rotation is to apply a nonsingular transformation to the factor pattern matrix in order to optimize a simplicity function that measures how close the elements of the factor pattern are to 0 or ± 1.

The common factors that are initially extracted are uncorrelated with each other. (For more information about factor extraction, see the section "Factor Extraction" on page 343.) If the factors are rotated by an orthogonal transformation, the rotated factors are still uncorrelated. If the factors are rotated by an oblique transformation, the rotated factors become correlated.

Specific methods of orthogonal rotation are as follows:

- varimax
- quartimax
- biquartimax
- equamax
- parsimax
- factor parsimax

Oblique versions of these methods are also available. Other specific methods of (direct) oblique rotation are as follows:

- quartimin
- biquartimin
- covarimin

General methods of orthogonal rotation are as follows:

- orthomax with user-specified gamma
- Crawford-Ferguson family with user-specified weights on variable parsimony and factor parsimony
- generalized Crawford-Ferguson family with user-specified weights

General methods of oblique rotation are as follows:

- direct oblimin with user-specified tau
- Crawford-Ferguson family with user-specified weights on variable parsimony and factor parsimony

- generalized Crawford-Ferguson family with user-specified weights
- promax with user-specified exponent

Crawford and Ferguson (1970) provide an excellent summary of a wide class of analytic rotations. See Harman (1976) and Mulaik (1972) for further information.

Oblique rotations often produce more useful or interpretable patterns than orthogonal rotations. However, a consequence of correlated factors is that there is no single unambiguous measure of the importance of a factor in explaining a variable. Thus, for oblique rotations, the pattern matrix does not provide all the necessary information for interpreting the factors; you must also examine the factor structure and the reference structure.

Note that factor rotation is conceptually and mathematically equivalent to finding a suitable coordinate system for locating the extracted factors. For this reason, factor rotation does not change the statistical explanatory power of the factors, and there is no mathematical or statistical reason to prefer one rotation over another. Rather, the choice is subjective and based on which rotation produces a solution that is most suitable for interpretation. Although rotating a factor solution does not increase or decrease the statistical quality of the factor model, it can simplify the interpretation of the factors and redistribute the variance explained by the factors.

Simplicity Functions

To rotate a factor pattern is to apply a nonsingular linear transformation to the unrotated factor pattern matrix. An optimal transformation is usually defined as a minimum or maximum point of a simplicity function. Different rotation methods are based on different simplicity functions.

For the promax rotation, the simplicity function that is used is the sum of squared differences between the rotated factor pattern and the target matrix. The optimal transformation is obtained by minimizing this simplicity function with respect to the choices of all possible transformation.

For the generalized Crawford-Ferguson family of rotations (Jennrich 1973), the simplicity function that is optimized is

$$f = k_1 Z + k_2 H + k_3 V + k_4 Q$$

where

$$Z = (\sum_j \sum_i b_{ij}^2)^2, \quad H = \sum_i (\sum_j b_{ij}^2)^2$$

$$V = \sum_j (\sum_i b_{ij}^2)^2, \quad Q = \sum_j \sum_i b_{ij}^4$$

and where k_1, k_2, k_3, and k_4 are constants, and b_{ij} represents an element of the rotated pattern matrix. Except in specialized research, it is uncommon in practice to use this simplicity function directly for rotation. However, this simplicity function reduces to many well-known classes of rotations. One of these classes is the Crawford-Ferguson family (Crawford and Ferguson 1970), which minimizes

$$f_{\text{cf}} = c_1(H - Q) + c_2(V - Q)$$

where c_1 and c_2 are constants, $(H - Q)$ represents variable (row) parsimony, and $(V - Q)$ represents factor (column) parsimony. Therefore, the relative importance of both the variable parsimony and the factor

parsimony is adjusted using the constants c_1 and c_2. The orthomax class (see Harman 1976) maximizes the function

$$f_{\text{or}} = pQ - \gamma V$$

where γ is the orthomax weight and is usually between 0 and the number of variables p. The oblimin class minimizes the function

$$f_{\text{ob}} = p(H - Q) - \tau(Z - V)$$

where τ is the oblimin weight. For practical purposes, a negative or 0 value for τ is recommended.

All the preceding definitions are for rotations without row normalization. For rotations that use Kaiser normalization, the definition of b_{ij} is replaced by b_{ij} / h_i, where h_i is the communality estimate of variable i.

Variable Weights and Variance Explained

In PROC EFA, the principal, iterative principal, and unweighted least squares methods perform an unweighted analysis of the input variables. Alpha and ML factor analyses compute variable weights on the basis of the communalities (Harman 1976, pp. 217–218). In alpha factor analysis, the weight of a variable is the reciprocal of its communality. In ML factor analysis, the weight is the reciprocal of the uniqueness.

For uncorrelated factors, the factor pattern is equal to both the matrix of standardized regression coefficients for predicting variables from common factors and the matrix of correlations between variables and common factors. The variance explained by a factor is computed by taking the sum of squares of the corresponding column of the factor pattern, yielding an unweighted result. If the square of each loading is multiplied by the weight of the variable before the sum is taken, the result is the weighted variance explained, which (for the methods that PROC EFA implements) is equal to the corresponding eigenvalue. Whether the weighted or unweighted result is more important depends on the purpose of the analysis.

In the case of correlated factors, the variance explained by a factor can be computed with or without taking the other factors into account. If you want to ignore the other factors, the variance explained is given by the weighted or unweighted sum of squares of the appropriate column of the factor structure, because the factor structure contains simple correlations. If you want to subtract the variance explained by the other factors from the amount explained by the factor in question (the Type II variance explained), you can take the weighted or unweighted sum of squares of the appropriate column of the reference structure, because the reference structure contains semipartial correlations. There are other ways to measure the variance explained. For example, you can eliminate from each factor the variance explained by previous factors and compute a Type I variance explained. Harman (1976, pp. 268–270) provides another method, which is based on direct and joint contributions.

Final communality estimates are the squared multiple correlations for predicting the variables from the estimated factors. You can obtain them by taking the sum of squares of each row of the factor pattern, or a weighted sum of squares if variable weights are used.

Limitations

No computer program can reliably determine the optimal number of factors, because the decision is ultimately subjective. You should not blindly accept the number of factors that is obtained by default. Instead, use your own judgment.

Likewise, the interpretation of a factor solution is subjective. For this reason, there is no combination of extraction and rotation methods that is always correct for a given application. Rather, you should consider the assumptions, advantages, and disadvantages of each method and decide what is appropriate for your application. In an exploratory factor analysis, you might consider trying several different methods or extracting different numbers of factors to see which solution is easiest to interpret.

Singular correlation matrices cause problems for SMC or ASMC priors. Singularities can result from using a variable that is the sum of other variables, coding too many dummy variables from a classification variable, or having more variables than observations.

Missing Values

Observations that have missing values for any analysis variable, frequency variable, or weight variable are omitted from the analysis.

ODS Table Names

Each table that the EFA procedure creates has a name that is associated with it. You must use this name to refer to the table when you use ODS statements. Table 10.3 lists the name of each table and a short description of its contents.

Table 10.3 Tables Produced by PROC EFA

Table Name	Description	Statement	Option
AlphaCoef	Coefficient alpha for each factor	PROC EFA	METHOD=ALPHA
AveParCorrControlPC	Average partial correlations after controlling for principal components	NFACTORS	TYPE=MAP2 or MAP4
CanCorr	Squared canonical correlations	PROC EFA	METHOD=ML
Eigenvalues	Eigenvalues of the reduced correlation matrix	Default output	
FactorPattern	Factor pattern	Default output	
FactorStructure	Factor structure (matrix of correlations between variables and common factors)	PROC EFA	ROTATE= any oblique rotation
FinalCommun	Final communalities	PROC EFA	METHOD=PRINCIPAL, PRINIT, or ULS
FinalCommunWgt	Final communalities with weights	PROC EFA	METHOD=ALPHA or ML
FitMeasures	Measures of fit	PROC EFA	METHOD=ML

Table 10.3 *continued*

Table Name	Description	Statement	Option
InterFactorCorr	Interfactor correlations	PROC EFA	ROTATE= any oblique rotation
IterHistory	Iteration history	PROC EFA	METHOD=ALPHA, ML, PRINIT, or ULS
ModelInfo	Model information	Default output	
NFactors	Number of factors suggested by each criterion	NFACTORS	
NObs	Number of observations read and used	Default output	
NormObliqueTrans	Normalized oblique transformation matrix	PROC EFA	ROTATE=PROMAX
ObliqueRotFactPat	Rotated factor pattern (standardized regression coefficients for predicting the variables from the factors)	PROC EFA	ROTATE= any oblique rotation
ObliqueTrans	Oblique transformation matrix	PROC EFA	ROTATE= any oblique rotation except promax
OrthRotFactPat	Rotated factor pattern	PROC EFA	ROTATE= any orthogonal rotation
OrthTrans	Orthogonal transformation matrix	PROC EFA	ROTATE= any orthogonal rotation
OutputCasTables	A special table that has information about all the data tables that are created during execution	DISPLAYOUT	
ParallelAnalysis	Parallel analysis results	NFACTORS	TYPE=PARALLEL
PriorCommun	Prior communalities	Default output	
ProcrustesTarget	Target matrix for Procrustes transformation	PROC EFA	ROTATE=PROMAX
ProcrustesTrans	Procrustes transformation matrix	PROC EFA	ROTATE=PROMAX
ReferenceAxisCorr	Reference axis correlations (partial correlations between the primary factors when all factors other than the two being correlated are partialed out)	PROC EFA	REFSTRUCT and ROTATE= any oblique rotation
ReferenceStructure	Reference structure (matrix of semipartial correlations between variables and common factors, removing from each common factor the effects of other common factors)	PROC EFA	REFSTRUCT and ROTATE= any oblique rotation
SignifTests	Significance tests	PROC EFA	METHOD=ML
SimpleStatistics	Simple statistics	Default output	

Table 10.3 *continued*

Table Name	Description	Statement	Option
Timing	Computational time	Default output	
VarExplain	Variance explained	PROC EFA	METHOD=PRINCIPAL, PRINIT, or ULS
VarExplainWgt	Variance explained with weights	PROC EFA	METHOD=ALPHA or ML
VarWeightRotate	Variable weights for rotation	PROC EFA	NORM=WEIGHT

Examples: EFA Procedure

The examples in this chapter are available in the GitHub repository located at https://github.com/sascommunities/doc-supplement-casstat.

Example 10.1: Estimating the Number of Factors or Components

(View the complete code for this example (efaex1.sas) in the example repository.)

This example shows how to use the EFA procedure to estimate either the number of common factors or the number of principal components that are likely to explain the observed correlations in a data set. It uses the simulated testdata data set that is described in the examples for the Principal Component Analysis action set in SAS Visual Statistics software. The data table is the result of multiplying two low-rank matrices, **A** and **B**, and adding random noise.

To create the data table testdata, you can modify the DATA step that is used in the Principal Component Analysis action set documentation. For this example, it is assumed that your libref is named mylib, but you can substitute any appropriately defined libref.

In this example, you want to determine how many latent factors are needed to explain the variation in the input data. The following statements perform this analysis:

```
proc efa data=mylib.testdata method=none;
   nfactors type=eigenvalue;
   nfactors type=proportion threshold=0.9;
run;
```

When you use PROC EFA to suggest the number of latent dimensions, you specify one or more NFACTORS statements. Each NFACTORS statement describes a criterion that is used to determine the number of factors to retain.

In this example, you specify two criteria. The first criterion is the eigenvalue criterion. This criterion is based on the eigenvalues of the reduced correlation matrix (that is, the correlation matrix of the input data, but with the diagonal elements replaced by the prior communality estimates). By default, any eigenvalue greater than 1 is assumed to indicate the existence of a latent factor. You can use the THRESHOLD= option to specify a threshold value to use instead of the default value.

The second criterion that you specify is the proportion criterion. This criterion is based on the cumulative fraction of the common variance of the input data set that is explained by the latent factors that correspond to each eigenvalue. In this example, you use the THRESHOLD= option to specify a threshold value of 0.9. At this threshold, the number of factors that the proportion criterion suggests is the smallest number of factors that are necessary to explain 90% of the common variance of the input data set.

By default, PROC EFA uses squared multiple correlations (SMC) for the prior communality estimates. This means that the prior communality estimate for each variable is the squared multiple correlation of that variable with all other variables. The prior communalities are placed on the diagonal of the data correlation matrix before the eigenvalues of that matrix are computed. You can use the PRIORS= option to specify different prior communality estimates.

The "Number of Observations" table in Output 10.1.1 summarizes the number of observations that are read from the input data table and the number of observations that are used for the analysis. For this analysis, these two numbers are the same. These numbers would be different if some observations were discarded from the analysis. PROC EFA uses only complete cases for analysis, so a record is discarded if any field contains a missing value.

Output 10.1.1 Number of Observations

The EFA Procedure

Number of Observations Read	1000000
Number of Observations Used	1000000

The "Simple Statistics" table in Output 10.1.2 summarizes the means and standard deviations of the variables that are included in the analysis.

Output 10.1.2 Simple Statistics

Simple Statistics

Variable	Mean	Standard Deviation
x1	0.00051958	2.03642
x2	0.00171	3.04429
x3	-0.00036170	1.25872
x4	0.00194	1.79332
x5	0.00058028	1.47249
x6	-0.00083852	1.52032
x7	0.00230	1.83223
x8	-0.00357	2.30643
x9	-0.00022835	2.13378
x10	0.00160	3.40267

The primary output of the analysis is shown in the "Determination of Number of Factors" table in Output 10.1.3. In this analysis, the results for both specified criteria suggest that three latent factors are sufficient to describe the input data.

In general, when you specify multiple criteria, PROC EFA must combine the results from all the active criteria to form a final estimate for the number of factors. By default, PROC EFA uses the minimum number

of factors among all the active criteria. You can use the NFACTORS= option to specify a different method to obtain a final estimate. You can use the STATUS= option to change a criterion from active to inactive.

A footnote at the bottom of the table summarizes the minimum and maximum number of factors that are determined by the active criteria. It also summarizes the mean and median values of these numbers. In this analysis, the two criteria suggest the same number of latent factors, so all four summary values are the same. Thus the final result of this analysis is that three factors should be retained.

Output 10.1.3 Number of Factors

Determination of Number of Factors		
Criterion	**Description**	**Factors**
1	Minimum Eigenvalue (Eigenvalue > 1.00)	3
2	Total Proportion (Threshold = 0.90)	3
Min = 3, Max = 3, Mean = 3, Median = 3		

Additional information about the eigenvalue and proportion criteria is presented in the "Eigenvalues of the Reduced Correlation Matrix" table in Output 10.1.4. For the proportion criteria, the number of factors is the minimum number that is required in order for the cumulative common variance to meet or exceed the user-specified threshold. The table shows that three factors are sufficient to explain approximately 93% of the cumulative common variance in the original data. Thus the proportion criterion with the option value THRESHOLD=0.90 produces an estimate of three factors. For the eigenvalue criterion, the third-largest eigenvalue is larger than 1.5, whereas the fourth-largest eigenvalue is much smaller than 1.0. For this reason, the eigenvalue criterion that is specified in this analysis suggests that three factors should be retained.

Output 10.1.4 Eigenvalues and Variance Explained

Eigenvalues of the Reduced Correlation Matrix				
	Eigenvalue	**Difference**	**Proportion**	**Cumulative**
1	5.822573	3.924604	0.5850	0.5850
2	1.897969	0.333507	0.1907	0.7757
3	1.564462	0.886826	0.1572	0.9329
4	0.677636	0.678439	0.0681	1.0009
5	-0.000804	0.000258	-0.0001	1.0009
6	-0.001062	0.000542	-0.0001	1.0008
7	-0.001604	0.000082164	-0.0002	1.0006
8	-0.001686	0.000107	-0.0002	1.0004
9	-0.001794	0.000669	-0.0002	1.0002
10	-0.002462		-0.0002	1.0000

In addition to estimating the number of latent factors, you can also use PROC EFA to estimate the number of principal components that are needed to approximate an input data set. For example, you might want to determine how many components you need to retain to approximate a prespecified proportion of the total variance in the original data set. The following statements perform this analysis:

```
proc efa data=mylib.testdata method=none priors=one;
   nfactors type=proportion threshold=0.90;
   nfactors type=proportion threshold=0.95;
   nfactors type=proportion threshold=0.99;
run;
```

These statements demonstrate how you can specify three different threshold values for the proportion criterion. Each of these threshold values is applied separately to estimate the number of factors in the input data set. The specification of multiple threshold values is a useful technique to assess the degree to which the analysis results are robust to different threshold values.

In this analysis, the PRIORS=ONE option is used to specify that the prior communality estimates for all variables should be set to 1. When you specify this option value, PROC EFA estimates the number of principal components that are needed to approximate the input data set.

The "Number of Observations" and "Simple Statistics" tables for this analysis are identical to those in the previous analysis. The "Determination of Number of Factors" table is shown in Output 10.1.5. Of the three specified threshold values, the first value produces an estimate that three factors should be retained, and the remaining two criteria produce estimates that four factors should be retained.

Output 10.1.5 Number of Factors

The EFA Procedure

Determination of Number of Factors

Criterion	Description	Factors
1	Total Proportion (Threshold = 0.90)	3
2	Total Proportion (Threshold = 0.95)	4
3	Total Proportion (Threshold = 0.99)	4
Min = 3, Max = 4, Mean = 4, Median = 4		

Additional information about the number of components that are suggested by these three criteria is presented in the "Eigenvalues of the Correlation Matrix" table in Output 10.1.6. As in the previous analysis, the proportion criterion examines the cumulative variance explained by eigenvalues and corresponding principal components. For this analysis, the principal components that correspond to the three largest eigenvalues explain just under 93% of the total variance of the input data set. Thus, an estimate of three components is obtained when you set the threshold value to 0.9. When you add a fourth component, the cumulative variance explained exceeds 99%, so that four components are estimated when you set the THRESHOLD= option to 0.95 or 0.99.

Output 10.1.6 Eigenvalues and Variance Explained

Eigenvalues of the Correlation Matrix

	Eigenvalue	Difference	Proportion	Cumulative
1	5.827238	3.924651	0.5827	0.5827
2	1.902587	0.333170	0.1903	0.7730
3	1.569417	0.887490	0.1569	0.9299
4	0.681927	0.676456	0.0682	0.9981
5	0.005471	0.001057	0.0005	0.9987
6	0.004414	0.001209	0.0004	0.9991
7	0.003205	0.000800	0.0003	0.9994
8	0.002405	0.000628	0.0002	0.9997
9	0.001777	0.000218	0.0002	0.9998
10	0.001559		0.0002	1.0000

Example 10.2: Principal Factor Analysis

(View the complete code for this example (efaex2.sas) in the example repository.)

This example shows how to use the EFA procedure to perform a principal factor analysis. It uses the socioeconomics data set that is described in the examples for the FACTOR procedure in SAS/STAT software. This data set contains five variables, which represent total population (Population), median years of school (School), total employment (Employment), miscellaneous professional services (Services), and median house value (HouseValue). Each observation represents one of twelve census tracts in the Los Angeles Standard Metropolitan Statistical Area.

For this example, it is assumed that your libref is named mylib, but you can substitute any appropriately defined libref.

"Example 44.2: Principal Factor Analysis" (Chapter 44, *SAS/STAT User's Guide*) and "Example 44.4: Maximum Likelihood Factor Analysis" (Chapter 44, *SAS/STAT User's Guide*) show that the socioeconomics data set is well described by a two-factor model. The following code uses PROC EFA to perform a principal factor analysis to extract two factors:

```
proc efa data=mylib.socioeconomics nfactors=2
   method=principal rotate=promax reorder;
run;
```

When you use PROC EFA to extract a known number of factors, you use the NFACTORS= option to specify that number of factors. In this example, you specify the METHOD=PRINCIPAL option to perform principal factor extraction. PROC EFA uses squared multiple correlations by default to estimate the prior communalities for an analysis. You can use the PRIORS= option to specify alternative prior communality estimates.

The factors that are initially extracted by this method are orthogonal. To facilitate the interpretation of these factors, you specify the ROTATE=PROMAX option to perform factor rotation by the promax method. By default, the promax rotation first computes an orthogonal varimax prerotation, which is followed by an oblique Procrustes rotation. You can use the PREROTATE= option to specify a different prerotation method, which can be either orthogonal or oblique.

Finally, you specify the REORDER option so that the rows of the various factor matrices are ordered in such a way that variables whose highest absolute loading is on the first factor are displayed first, followed by variables whose highest absolute loading is on the second factor, and so on.

The first outputs that PROC EFA prints are the "Model Information" table and the "Number of Observations" table. These are shown in Output 10.2.1. Together, these tables summarize important information about the input data that are analyzed, as well as the primary analysis options that you specified. Next, the "Simple Statistics" table in Output 10.2.2 summarizes the means and standard deviations of the analysis variables.

Output 10.2.1 Model Information and Number of Observations

The EFA Procedure

Model Information	
Data Source	SOCIOECONOMICS
Data Type	Raw
Prior Communalities	Squared Multiple Correlations
Factor Extraction	Principal Factor Method
Factor Rotation	Promax

Number of Observations Read	12
Number of Observations Used	12

Output 10.2.2 Simple Statistics

Simple Statistics

Variable	Mean	Standard Deviation
Population	6241.66667	3439.99427
School	11.44167	1.78654
Employment	2333.33333	1241.21153
Services	120.83333	114.92751
HouseValue	17000	6367.53128

The prior communality estimates are shown in the table in Output 10.2.3. The prior communalities represent initial estimates of the common variance for each analysis variable. (For more information about communalities and common variance, see the section "The Common Factor Model" on page 337.) The prior communalities are used to compute the reduced correlation matrix. The reduced correlation matrix is the observed correlation matrix for the analysis variables in which the diagonal entries of that matrix are replaced by the prior communalities.

Output 10.2.3 Prior Communalities

The EFA Procedure

Prior Communality Estimates				
Population	**School**	**Employment**	**Services**	**HouseValue**
0.96859160	0.82228514	0.96918082	0.78572440	0.84701921

Next, the eigenvalues of the reduced correlation matrix are computed. These eigenvalues are summarized in the "Preliminary Eigenvalues" table in Output 10.2.4. In this example, the first two largest positive eigenvalues of the reduced correlation matrix account for 101.31% of the common variance. This is possible because the reduced correlation matrix, in general, is not necessarily positive definite, and negative eigenvalues for the matrix are possible. A pattern like this adds further support to the assumption that you might not need more than two common factors in this analysis.

Output 10.2.4 Preliminary Eigenvalues

Eigenvalues of the Reduced Correlation Matrix: Total = 4.39280116 Average = 0.87856023				
	Eigenvalue	Difference	Proportion	Cumulative
1	2.734301	1.018232	0.6225	0.6225
2	1.716069	1.676506	0.3907	1.0131
3	0.039563	0.064086	0.0090	1.0221
4	-0.024523	0.048084	-0.0056	1.0165
5	-0.072608		-0.0165	1.0000

The principal factor pattern with the two extracted factors is displayed in Output 10.2.5. The Services variable has the largest loading on the first factor, and the Population variable has the smallest. The Population and Employment variables have large positive loadings on the second factor, and the HouseValue and School variables have large negative loadings.

Output 10.2.5 Factor Pattern

Factor Pattern		
	Factor1	Factor2
Services	0.87899	-0.15847
HouseValue	0.74215	-0.57806
Employment	0.71447	0.67936
School	0.71370	-0.55515
Population	0.62533	0.76621

The variance explained by the extracted factors is summarized in Output 10.2.6, and the final communality estimates are shown in Output 10.2.7.

Output 10.2.6 Variance Explained (Initial Solution)

Variance Explained by Each Factor	
Factor1	Factor2
2.7343008	1.7160687

Output 10.2.7 Final Communalities (Initial Solution)

Final Communality Estimates: Total = 4.450370				
Population	School	Employment	Services	HouseValue
0.97811334	0.81756387	0.97199928	0.79774304	0.88494998

Next, you want to rotate the factor pattern such that most variables would have zero loadings on most factors. In this example, you specified a promax rotation with a varimax prerotation. To yield the prerotated factor pattern, the initial factor loading matrix is postmultiplied by an orthogonal transformation matrix that satisfies the varimax criterion. This orthogonal transformation matrix, as well as the varimax-rotated factor pattern, is shown in Output 10.2.8. The rotation leads to small loadings of Population and Employment on the first factor and small loadings of HouseValue and School on the second factor. Services appears to have a larger

loading on the first factor than it has on the second factor, although both loadings are substantial. Hence, Services appears to be factorially complex.

The REORDER option helps you see the variable clusters clearly in the factor pattern. The first factor is associated more with the first three variables, HouseValue, School, and Services. The second factor is associated more with the last two variables, Population and Employment. For orthogonal factor solutions such as the current varimax-rotated solution, you can also interpret the values in the factor loading matrix as correlations. For example, HouseValue and Factor1 have a high correlation of 0.94, whereas Population and Factor1 have a low correlation of 0.03.

Output 10.2.8 Orthogonal Transformation and Factor Pattern (Prerotated Solution)

The EFA Procedure

Orthogonal Transformation Matrix

	1	2
1	0.78895	0.61446
2	-0.61446	0.78895

Rotated Factor Pattern

	Factor1	Factor2
HouseValue	0.94072	-0.00004
School	0.90419	0.00055
Services	0.79085	0.41509
Population	0.02255	0.98874
Employment	0.14625	0.97499

The variance explained by the prerotated factors, as well as the final communality estimates, is shown in Output 10.2.9. The variance explained by the factors is more evenly distributed in the varimax-rotated solution than in the unrotated solution. Indeed, this is a typical fact for any kind of factor rotation. In the current example, before the varimax rotation, the two factors explain 2.73 and 1.72, respectively, of the common variance (see Output 10.2.6). After the varimax rotation, the two rotated factors explain 2.35 and 2.10, respectively, of the common variance. However, the total variance explained by the factors remains unchanged after the varimax rotation. This invariance property is also observed for the communalities of the variables after the rotation, as you can see by comparing the current communality estimates in Output 10.2.6 to those in Output 10.2.7.

Output 10.2.9 Variance Explained and Final Communalities (Prerotated Solution)

Variance Explained by Each Factor

Factor1	Factor2
2.3498567	2.1005128

Final Communality Estimates: Total = 4.450370

Population	School	Employment	Services	HouseValue
0.97811334	0.81756387	0.97199928	0.79774304	0.88494998

Some factor analysts believe that common factors are seldom orthogonal. Thus an obliquely rotated factor solution might be more desirable, or at least it should be attempted. In the current example, you

specified the ROTATE=PROMAX option to perform a promax rotation, which allows factors to be correlated. Output 10.2.10 shows the Procrustes target, followed by the Procrustes transformation matrix. This is the matrix that transforms the varimax factor pattern so that the rotated pattern is as close as possible to the Procrustes target, which is created by exponentiating the elements of the prerotated factor pattern (while keeping the signs of the values). However, because the variances of factors must be fixed at 1 during the oblique transformation, a normalized version of the Procrustes transformation matrix is the one that is actually used in the transformation. This normalized transformation matrix is shown at the bottom of Output 10.2.10.

Output 10.2.10 Promax Rotation: Procrustes Target and Transformation Matrices

The EFA Procedure

Target Matrix for Procrustes Transformation

	Factor1	Factor2
HouseValue	1.00000	-0.00000
School	1.00000	0.00000
Services	0.69421	0.10045
Population	0.00001	1.00000
Employment	0.00326	0.96793

Procrustes Transformation Matrix

	1	2
1	1.04117	-0.09865
2	-0.10572	0.96303

Normalized Oblique Transformation Matrix

	1	2
1	0.73803	0.54202
2	-0.70555	0.86528

Using this transformation matrix leads to the promax-rotated factor solution, as shown in Output 10.2.11. After the promax rotation, the factors are no longer uncorrelated. As shown in the "Interfactor Correlations" table, the correlation of the two factors is now 0.20. In the unrotated and the varimax solutions, the two factors are not correlated.

Output 10.2.11 Promax Rotation: Factor Correlations and Rotated Factor Pattern

Interfactor Correlations

	Factor1	Factor2
Factor1	1.00000	0.20188
Factor2	0.20188	1.00000

Output 10.2.11 *continued*

Rotated Factor Pattern (Standardized Regression Coefficients)		
	Factor1	Factor2
HouseValue	0.95558	-0.09792
School	0.91842	-0.09352
Services	0.76053	0.33932
Population	-0.07908	1.00192
Employment	0.04799	0.97509

Unlike orthogonal factor solutions, where you can interpret the factor loadings as correlations between variables and factors, in oblique factor solutions such as the promax solution, you have to turn to the factor structure matrix to examine the correlations between variables and factors. Output 10.2.12 shows the factor structures for the promax-rotated solution. In this example, the factor structure matrix reflects a pattern similar to that of the factor pattern matrix shown in Output 10.2.11. The critical difference is that you can have the correlation interpretation only by using the factor structure matrix. For example, in the factor structure matrix, the correlation between Population and Factor 2 is 0.986. The corresponding value shown in the factor pattern matrix is 1.002, which you certainly cannot interpret as a correlation coefficient.

Output 10.2.12 Factor Structure

Factor Structure (Correlations)		
	Factor1	Factor2
HouseValue	0.93582	0.09500
School	0.89954	0.09189
Services	0.82903	0.49286
Population	0.12319	0.98596
Employment	0.24484	0.98478

When factors are correlated, the common variance explained by each factor is not as straightforward as it is in the case when the factors are uncorrelated. The contributions from the other factors must be either eliminated or ignored. Both computations are shown in Output 10.2.13.

When you ignore the other factor, the common variance explained by the promax-rotated factors is 2.45 and 2.20, respectively. In contrast with orthogonal factor solutions (such as the prerotated varimax solution), the variance explained by these promax-rotated factors (ignoring other factors) does not sum to the total communality estimate of 4.45 (see Output 10.2.14).

Also shown in Output 10.2.13 is the "Variance Explained by Each Factor, Eliminating Other Factors" table. The values in this table represent unique variances explained by the factors, partialing out the variances explained by all other factors (or eliminating all other factors, as the table title suggests). In the current example, Factor1 explains 2.25 of the variable variances, partialing out all variable variances explained by Factor2.

Output 10.2.13 Variance Explained

Variance Explained by Each Factor, Ignoring Other Factors	
Factor1	**Factor2**
2.4473495	2.2022803

Variance Explained by Each Factor, Eliminating Other Factors	
Factor1	**Factor2**
2.2480892	2.0030200

The communalities of the variables, as shown in Output 10.2.14, do not change from rotation to rotation. They are still the same set of communalities in the initial, varimax-rotated, and promax-rotated solutions. This is a basic fact about factor rotations: they redistribute the variance explained by the factors; the total variance explained by the factors for any variable (that is, the communality of the variable) remains unchanged.

Output 10.2.14 Final Communalities

Final Communality Estimates: Total = 4.450370				
Population	**School**	**Employment**	**Services**	**HouseValue**
0.97811334	0.81756387	0.97199928	0.79774304	0.88494998

Example 10.3: Iterative Factor Extraction

(View the complete code for this example (efaex3.sas) in the example repository.)

This example shows how to use the EFA procedure to perform an iterative factor analysis. It uses the socioeconomics data set that is described in the examples for the FACTOR procedure in SAS/STAT software. These are the same data that are analyzed in "Example 10.2: Principal Factor Analysis" on page 355.

For this example, it is assumed that your libref is named mylib, but you can substitute any appropriately defined libref.

In Example 10.2, the socioeconomics data set is used in a principal factor analysis. In that analysis, squared multiple correlations are used for prior communality estimates. After two factors are extracted, the factor pattern is used to compute updated estimates of the communalities. You might wonder whether this process can be repeated. That is, can you use the final communalities as prior communalities in a subsequent analysis? This is a motivation behind iterative methods of factor extraction.

The following code uses PROC EFA to perform an *iterative* principal factor analysis with squared multiple correlations for the prior communality estimates:

```
proc efa data=mylib.socioeconomics nfactors=2 method=prinit heywood;
run;
```

This code is very similar to the code in Example 10.2. The major difference is the use of the METHOD=PRINIT option here to specify an iterative principal factor extraction. This means that the

prior communalities are used to compute an initial factor pattern, which is then used to compute new communalities, which is then used to compute a new factor pattern, and so on. This process continues until the change in the estimated communalities is sufficiently small. You can use the CONVERGE= option to set a specific value for the convergence threshold. The default value is 0.001.

Practically speaking, communalities are squared correlations and so must be bounded above by 1.0. However, during the iterative process of factor extraction, it is possible for a communality to meet or exceed this upper bound. When a communality reaches this bound, it is called a Heywood case. When a communality exceeds this bound, it is called an ultra-Heywood case. (For more information about communalities and Heywood cases, see the section "Heywood Cases and Other Anomalies of Communality Estimates" on page 344.) In this analysis, you use the HEYWOOD option to request that the communalities be bounded above by 1.0 during the iterative process. This means that if a communality exceeds 1.0 during the iterations, it is replaced by a value of 1.0. If you omit the HEYWOOD option, then by default PROC EFA terminates with an error when it encounters a Heywood or ultra-Heywood case.

Note that no rotation is requested in the current analysis. This is simply for clarity of exposition, so that this example can focus on iterative extraction methods. You can use the ROTATE= option to request a rotation for a factor pattern that is extracted using an iterative procedure in the same manner that you would for a noniterative extraction method.

The initial outputs that are produced by PROC EFA are shown in Output 10.3.1 through Output 10.3.4. These are identical to the outputs that are produced when you perform a noniterative principal factor analysis. For a discussion of these outputs, see Example 10.2.

Output 10.3.1 Model Information and Number of Observations

The EFA Procedure

Model Information	
Data Source	SOCIOECONOMICS
Data Type	Raw
Prior Communalities	Squared Multiple Correlations
Factor Extraction	Iterative Principal Factor Method
Factor Rotation	None

Number of Observations Read	12
Number of Observations Used	12

Output 10.3.2 Simple Statistics

Simple Statistics

Variable	Mean	Standard Deviation
Population	6241.66667	3439.99427
School	11.44167	1.78654
Employment	2333.33333	1241.21153
Services	120.83333	114.92751
HouseValue	17000	6367.53128

Output 10.3.3 Prior Communalities

The EFA Procedure

Prior Communality Estimates				
Population	School	Employment	Services	HouseValue
0.96859160	0.82228514	0.96918082	0.78572440	0.84701921

Output 10.3.4 Preliminary Eigenvalues

Preliminary Eigenvalues: Total = 4.39280116 Average = 0.87856023				
	Eigenvalue	Difference	Proportion	Cumulative
1	2.734301	1.018232	0.6225	0.6225
2	1.716069	1.676506	0.3907	1.0131
3	0.039563	0.064086	0.0090	1.0221
4	-0.024523	0.048084	-0.0056	1.0165
5	-0.072608		-0.0165	1.0000

The iterative factor extraction process is summarized in Output 10.3.5. You can see that a Heywood case is encountered in the eighth iteration of the algorithm and remains in place until the algorithm converges.

Output 10.3.5 Iteration History

Iteration	Change	Communalities				
1	0.0379	0.97811	0.81756	0.97200	0.79774	0.88495
2	0.0235	0.98398	0.80831	0.97162	0.80096	0.90845
3	0.0161	0.98814	0.79921	0.97003	0.80132	0.92453
4	0.0117	0.99143	0.79155	0.96804	0.80085	0.93622
5	0.0088	0.99421	0.78548	0.96597	0.80021	0.94501
6	0.0067	0.99665	0.78078	0.96397	0.79961	0.95171
7	0.0052	0.99886	0.77717	0.96207	0.79909	0.95687
8	0.0040	1.00000	0.77442	0.96030	0.79867	0.96086
9	0.0031	1.00000	0.77232	0.95884	0.79834	0.96395
10	0.0024	1.00000	0.77072	0.95785	0.79810	0.96635
11	0.0019	1.00000	0.76949	0.95717	0.79791	0.96822
12	0.0015	1.00000	0.76854	0.95671	0.79776	0.96967
13	0.0011	1.00000	0.76782	0.95640	0.79764	0.97080
14	0.0009	1.00000	0.76726	0.95618	0.79754	0.97168

The eigenvalues of the reduced correlation matrix, the factor pattern, and the variance explained by the extracted factors are summarized in Output 10.3.6, Output 10.3.7, and Output 10.3.8, respectively. The final communality estimates are displayed in Output 10.3.9. In that table, you can see that the final communality estimate for Population is an ultra-Heywood case. Although the HEYWOOD option causes PROC EFA to replace out-of-bound communality estimates with ones during the iterative steps, this is not sufficient to effectively constrain the final community estimates after factoring the reduced correlation matrix. Consequently, ultra-Heywood cases might still be observed for the iterative principal factor method even if you specify this option. The ultra-Heywood case implies that some unique factor has negative variance—a clear indication that the factor solution in this case is problematic, if not invalid.

Output 10.3.6 Final Eigenvalues

Eigenvalues of the Reduced Correlation Matrix: Total = 4.49264739 Average = 0.89852948				
	Eigenvalue	Difference	Proportion	Cumulative
1	2.755916	1.016328	0.6134	0.6134
2	1.739588	1.708479	0.3872	1.0006
3	0.031109	0.033620	0.0069	1.0076
4	-0.002511	0.028943	-0.0006	1.0070
5	-0.031455		-0.0070	1.0000

Output 10.3.7 Factor Pattern

Factor Pattern		
	Factor1	Factor2
Population	0.62254	0.78441
School	0.70213	-0.52371
Employment	0.70144	0.68130
Services	0.88116	-0.14526
HouseValue	0.77905	-0.60395

Output 10.3.8 Variance Explained

Variance Explained by Each Factor	
Factor1	Factor2
2.7559158	1.7395881

Output 10.3.9 Final Communalities

Final Communality Estimates: Total = 4.495504				
Population	School	Employment	Services	HouseValue
1.00285106	0.76725612	0.95618481	0.79753541	0.97167647

You might instead consider an alternative factor extraction method to see if you can obtain a different factor solution. The following code uses maximum likelihood (ML) estimation for factor extraction:

```
proc efa data=mylib.socioeconomics nfactors=2 method=ml heywood;
run;
```

Like iterative principal factor extraction, the ML method is an iterative method. The iteration history is summarized in Output 10.3.10. The ML method converges in only 6 iterations, compared to the 14 iterations that are required for the iterative principal factor extraction method. Both methods encounter a Heywood case; the ML method encounters the Heywood case in the very first iteration.

Output 10.3.10 Maximum Likelihood Iteration History

The EFA Procedure

Iteration	Criterion	Ridge	Change	Communalities			
1	0.4092306	0.0000	0.0908	1.00000	0.79229	0.93333	0.80068	0.93780
2	0.3088104	0.0000	0.0255	1.00000	0.79690	0.95885	0.81497	0.93545
3	0.3069206	0.0000	0.0130	1.00000	0.80872	0.95904	0.81890	0.92249
4	0.3067496	0.0000	0.0035	1.00000	0.80887	0.95952	0.81536	0.92334
5	0.3067333	0.0000	0.0013	1.00000	0.80995	0.95952	0.81589	0.92207
6	0.3067316	0.0000	0.0004	1.00000	0.80992	0.95956	0.81552	0.92219

In ML factor analysis, the eigenvalues of the *weighted* reduced correlation matrix are computed. The weights that are used are the reciprocals of the unique error variances (that is, one minus the communalities). When a Heywood case is encountered, the weight for that variable is not defined. For this reason, one of the eigenvalues in the "Eigenvalues of the Weighted Reduced Correlation Matrix" table in Output 10.3.11 is missing.

Output 10.3.11 Maximum Likelihood Final Eigenvalues

Eigenvalues of the Weighted Reduced Correlation Matrix: Total = 19.8268757 Average = 4.95671892				
	Eigenvalue	Difference	Proportion	Cumulative
1	.	.	.	.
2	19.826875	19.283728	1.0000	1.0000
3	0.543148	0.582869	0.0274	1.0274
4	-0.039721	0.463706	-0.0020	1.0254
5	-0.503427		-0.0254	1.0000

As with iterative principal factor extraction, the ML method encounters a Heywood case. However, unlike the iterative principal factor results, the converged solution for the ML method has only a Heywood case rather than an ultra-Heywood case. You can see this in the "Final Communality Estimates" table in Output 10.3.13. Although a Heywood case does not necessarily invalidate a factor solution, it should raise suspicion and requires further investigation.

Output 10.3.12 Maximum Likelihood Solution

Eigenvalues of the Weighted Reduced Correlation Matrix: Total = 19.8268757 Average = 4.95671892				
	Eigenvalue	Difference	Proportion	Cumulative
1	.	.	.	.
2	19.826875	19.283728	1.0000	1.0000
3	0.543148	0.582869	0.0274	1.0274
4	-0.039721	0.463706	-0.0020	1.0254
5	-0.503427		-0.0254	1.0000

Output 10.3.12 *continued*

Factor Pattern		
	Factor1	Factor2
Population	1.00000	0.00000
School	0.00975	0.89992
Employment	0.97245	0.11791
Services	0.43887	0.78926
HouseValue	0.02241	0.96004

Variance Explained by Each Factor		
Factor	Weighted	Unweighted
Factor1	24.436381	2.138861
Factor2	19.826875	2.368369

Output 10.3.13 Maximum Likelihood Final Communalities

Final Communality Estimates and Variable Weights		
Total Communality: Weighted = . Unweighted = 4.507229		
Variable	Communality	Weight
Population	1.00000000	.
School	0.80995115	5.26102907
Employment	0.95955909	24.7292489
Services	0.81553930	5.42072311
HouseValue	0.92217989	12.8522557

Example 10.4: Extracting and Interpreting Solutions with Maximum Likelihood Factor Analysis

(View the complete code for this example (efaex4.sas) in the example repository.)

This example shows how to use the EFA procedure to perform maximum likelihood factor analysis. It uses the jobratings data set that is described in the examples for the FACTOR procedure in SAS/STAT software. These are the same data that are analyzed in the section "Getting Started: EFA Procedure" on page 315.

For this example, it is assumed that your libref is named mylib, but you can substitute any appropriately defined libref.

In Example 10, the jobratings data set is analyzed to determine the number of latent factors that are likely to explain the data. The results of that analysis indicate that two or three factors are likely to explain the observed covariance among the variables. In the current example, PROC EFA is used to extract and rotate a factor pattern that has either two or three common factors.

First, you can use the following code to extract and rotate a factor pattern that has two factors:

```
proc efa data=mylib.jobratings nfactors=2 method=ml
   rotate=quartimin reorder fuzz=0.4;
run;
```

You use the NFACTORS= option to specify the number of factors to extract in the analysis. In this example, you specify the METHOD=ML option to perform a maximum likelihood (ML) factor analysis. The factors that are initially extracted by the ML method are orthogonal. To facilitate the interpretation of these factors, you specify the ROTATE=QUARTIMIN option to request factor rotation by the quartimin method. This is an oblique rotation, so the rotated factors might not be orthogonal. Finally, you specify the REORDER and FUZZ= options to further facilitate the interpretation of the factor solution. The REORDER option orders the rows of the various factor matrices such that variables whose highest absolute loading is on the first factor are displayed first, followed by variables whose highest absolute loading is on the second factor, and so on. The FUZZ=0.4 option causes all factor loadings less than 0.4 in magnitude to be printed as missing.

The first outputs that PROC EFA prints are the "Model Information" table and the "Number of Observations" table. These are shown in Output 10.4.1. Together, these tables summarize important information about the input data that are analyzed, as well as the primary analysis options that you specified. Next, the "Simple Statistics" table in Output 10.4.2 summarizes the means and standard deviations of the analysis variables.

Output 10.4.1 Model Information and Number of Observations

The EFA Procedure

Model Information	
Data Source	JOBRATINGS
Data Type	Raw
Prior Communalities	Squared Multiple Correlations
Factor Extraction	Maximum Likelihood Method
Factor Rotation	Quartimin

Number of Observations Read	103
Number of Observations Used	103

Output 10.4.2 Simple Statistics

Simple Statistics

Variable	Mean	Standard Deviation
Communication Skills	6.65049	1.76407
Problem Solving	6.63107	1.59035
Learning Ability	6.99029	1.33941
Judgment under Pressure	6.73786	1.73183
Observational Skills	6.93204	1.76158
Willingness to Confront Problems	7.29126	1.52516
Interest in People	6.70874	1.89235
Interpersonal Sensitivity	6.62136	1.76077
Desire for Self-Improvement	6.57282	1.72980
Appearance	7.00000	1.79869
Dependability	6.82524	1.91704
Physical Ability	7.20388	1.55525
Integrity	7.21359	1.84524

Prior communality estimates are shown in the table in Output 10.4.3. For this analysis, the prior communalities are the squared multiple correlations of each analysis variable with the other variables. This is the default setting for PROC EFA. You can use the PRIORS= option to change the default.

Output 10.4.3 Prior Communalities

The EFA Procedure

Prior Communality Estimates

Communication Skills	Problem Solving	Learning Ability	Judgment under Pressure	Observational Skills	Willingness to Confront Problems	Interest in People	Interpersonal Sensitivity
0.62981394	0.58657431	0.61009871	0.63766021	0.67187583	0.64779805	0.75641519	0.75584891

Desire for Self-Improvement	Appearance	Dependability	Physical Ability	Integrity
0.57460176	0.45505304	0.63449045	0.42245324	0.68195454

The prior communalities represent initial estimates of the common variance for each analysis variable. (For more information about communalities and common variance, see the section "The Common Factor Model" on page 337.) The prior communalities are used to compute the reduced correlation matrix.

Next, the eigenvalues of the reduced correlation matrix are computed. In fact, for ML factor analysis, the eigenvalues of the *weighted* reduced correlation matrix are computed. The weights that are used are the reciprocals of the unique error variances (that is, one minus the communalities). These eigenvalues are summarized in the "Preliminary Eigenvalues" table in Output 10.4.4. This table suggests that two factors are sufficient to explain approximately 96.6% of the common variance in the original data, whereas three factors are required in order for the cumulative variance explained to exceed 1.0. For a more detailed investigation of the tools that you can use to investigate the number of factors in the jobratings data set, see Example 10.

Output 10.4.4 Preliminary Eigenvalues

Preliminary Eigenvalues:
Total = 23.3301805 Average = 1.79462927

	Eigenvalue	Difference	Proportion	Cumulative
1	17.712722	12.878127	0.7592	0.7592
2	4.834595	3.455987	0.2072	0.9664
3	1.378608	0.739974	0.0591	1.0255
4	0.638634	0.096988	0.0274	1.0529
5	0.541646	0.355771	0.0232	1.0761
6	0.185875	0.197306	0.0080	1.0841
7	-0.011431	0.109243	-0.0005	1.0836
8	-0.120674	0.119664	-0.0052	1.0784
9	-0.240338	0.063988	-0.0103	1.0681
10	-0.304326	0.056731	-0.0130	1.0551
11	-0.361057	0.029331	-0.0155	1.0396
12	-0.390389	0.143296	-0.0167	1.0229
13	-0.533685		-0.0229	1.0000

ML factor analysis is an iterative procedure in which the prior communality estimates are used to compute the reduced the correlation matrix, then common factors are extracted from the reduced correlation matrix, and finally those common factors are used to compute updated communalities. This process is repeated until the communalities converge. A summary of this iterative process is shown in Output 10.4.5.

Output 10.4.5 Iteration History

Iteration	Criterion	Ridge	Change	Communalities									
1	1.1432750	0.0000	0.1624	0.56731	0.45839	0.54823	0.56927	0.69972	0.67780	0.80341	0.82730	0.56493	0.34689
				0.63350	0.26005	0.68041							
2	1.1413609	0.0000	0.0162	0.55965	0.44571	0.53917	0.56606	0.71591	0.68063	0.80206	0.82431	0.56910	0.34556
				0.63759	0.27106	0.68382							
3	1.1412755	0.0000	0.0030	0.55778	0.44268	0.53730	0.56480	0.71739	0.68336	0.80261	0.82351	0.56954	0.34464
				0.63859	0.27263	0.68463							
4	1.1412708	0.0000	0.0007	0.55737	0.44220	0.53659	0.56456	0.71792	0.68368	0.80282	0.82321	0.56953	0.34465
				0.63873	0.27311	0.68489							

PROC EFA computes several significance tests and measures of fit when you specify an ML factor analysis. These are shown in Output 10.4.6.

Output 10.4.6 Significance Tests and Measures of Fit (Two-Factor Model)

Significance Tests Based on 103 Observations

Test	DF	Chi-Square	Pr > ChiSq
H0: No common factors	78	852.5129	<.0001
HA: At least one common factor			
H0: 2 Factors are sufficient	53	108.9914	<.0001
HA: More factors are needed			

Output 10.4.6 *continued*

Chi-Square without Bartlett's Correction	116.40962
Akaike's Information Criterion	10.40962
Schwarz's Bayesian Criterion	-129.23102
Tucker and Lewis's Reliability Coefficient	0.89361

Output 10.4.7 Canonical Correlations

Squared Canonical Correlations	
Factor1	**Factor2**
0.94849033	0.85059176

The final weighted reduced correlation matrix is formed by placing the final communalities on the diagonal of the data correlation matrix. The eigenvalues of this final correlation matrix are shown in the "Eigenvalues of the Weighted Reduced Correlation Matrix" table in Output 10.4.8.

Output 10.4.8 Final Eigenvalues

Eigenvalues of the Weighted Reduced Correlation Matrix: Total = 24.1069015 Average = 1.85437704				
	Eigenvalue	**Difference**	**Proportion**	**Cumulative**
1	18.413832	12.720760	0.7638	0.7638
2	5.693071	4.622958	0.2362	1.0000
3	1.070113	0.613454	0.0444	1.0444
4	0.456659	0.058589	0.0189	1.0633
5	0.398070	0.241829	0.0165	1.0798
6	0.156241	0.147885	0.0065	1.0863
7	0.008356	0.116653	0.0003	1.0867
8	-0.108298	0.144262	-0.0045	1.0822
9	-0.252560	0.047610	-0.0105	1.0717
10	-0.300169	0.082954	-0.0125	1.0593
11	-0.383124	0.094605	-0.0159	1.0434
12	-0.477729	0.089831	-0.0198	1.0235
13	-0.567560		-0.0235	1.0000

The factor pattern that corresponds to these communalities is shown in Output 10.4.9. The missing values indicate that the absolute value of the corresponding element in the pattern does not exceed 0.4, as specified by the FUZZ= option. The variance explained by the extracted factors is summarized in Output 10.4.10, and the final communality estimates are shown in Output 10.4.11.

Output 10.4.9 Factor Pattern

Factor Pattern		
	Factor1	**Factor2**
Integrity	0.80231	.
Interest in People	0.80035	-0.40283
Dependability	0.79336	.
Interpersonal Sensitivity	0.75439	-0.50407
Judgment under Pressure	0.74785	.
Communication Skills	0.72073	.
Desire for Self-Improvement	0.70799	.
Problem Solving	0.64047	.
Observational Skills	0.61118	0.58687
Learning Ability	0.57532	0.45334
Appearance	0.51787	.
Physical Ability	0.48446	.
Willingness to Confront Problems	0.57156	0.59756

Values less than 0.4 are not printed.

Output 10.4.10 Variance Explained (Initial Solution)

Variance Explained by Each Factor		
Factor	**Weighted**	**Unweighted**
Factor1	18.413832	6.007072
Factor2	5.693071	1.632118

Output 10.4.11 Final Communalities (Initial Solution)

Final Communality Estimates and Variable Weights Total Communality: Weighted = 24.106903 Unweighted = 7.639189		
Variable	**Communality**	**Weight**
Communication Skills	0.55727959	2.25920908
Problem Solving	0.44207153	1.79275764
Learning Ability	0.53650447	2.15791760
Judgment under Pressure	0.56450349	2.29653380
Observational Skills	0.71795868	3.54514379
Willingness to Confront Problems	0.68375220	3.16130592
Interest in People	0.80283089	5.07138167
Interpersonal Sensitivity	0.82319143	5.65640994
Desire for Self-Improvement	0.56953539	2.32303202
Appearance	0.34463311	1.52589948
Dependability	0.63877379	2.76805050
Physical Ability	0.27320906	1.37572027
Integrity	0.68494568	3.17353979

To facilitate the interpretation of the factor solution, it is often helpful to rotate the initial factor pattern. For the current analysis, a factor rotation by the quartimin method was specified. This is an oblique rotation,

so the rotated factors are likely to be correlated. The oblique transformation matrix (that is, the matrix that maps the initial factor pattern into the rotated factor pattern) is shown in Output 10.4.12, as is the "Interfactor Correlations" table, which summarizes the correlations between the rotated factors. For this analysis, the two factors are moderately correlated with each other.

Output 10.4.12 Oblique Transformation Matrix

The EFA Procedure

Oblique Transformation Matrix		
	1	2
1	0.50933	0.64767
2	1.02518	-0.94389

Interfactor Correlations		
	Factor1	Factor2
Factor1	1.00000	0.48670
Factor2	0.48670	1.00000

The rotated factor pattern is shown in Output 10.4.13. Again, the FUZZ= option causes factor loadings whose absolute values are less than 0.4 to be printed as missing. Such smaller factor loadings are not considered to be salient, and printing the values as missing makes it easier to distinguish the relationships between the analysis variables and the computed factors. In the rotated factor pattern, almost all variables load primarily on one factor (Dependability is the exception). The factor pattern can be interpreted as the standardized regression coefficients for predicting the effects of the factors on the variables.

Output 10.4.13 Rotated Factor Pattern

Rotated Factor Pattern (Standardized Regression Coefficients)		
	Factor1	Factor2
Observational Skills	0.91294	.
Willingness to Confront Problems	0.90372	.
Learning Ability	0.75778	.
Desire for Self-Improvement	0.62848	.
Communication Skills	0.56648	.
Appearance	0.54721	.
Problem Solving	0.50924	.
Dependability	0.50323	0.42255
Physical Ability	0.44792	.
Interpersonal Sensitivity	.	0.96438
Interest in People	.	0.89859
Integrity	.	0.71131
Judgment under Pressure	.	0.55262

Values less than 0.4 are not printed.

The factor structure is shown in Output 10.4.14. The factor structure represents the correlation between the factors and the variables. This can be an important aid for interpretation when factors are correlated, as in the current example.

Output 10.4.14 Factor Structure

Factor Structure (Correlations)		
	Factor1	**Factor2**
Observational Skills	0.83599	.
Willingness to Confront Problems	0.80937	.
Learning Ability	0.73087	.
Desire for Self-Improvement	0.73162	0.51779
Communication Skills	0.70432	0.55893
Appearance	0.58344	.
Problem Solving	0.62911	0.49414
Dependability	0.70889	0.66747
Physical Ability	0.51049	.
Interpersonal Sensitivity	.	0.89988
Interest in People	0.43202	0.89600
Integrity	0.54666	0.80888
Judgment under Pressure	0.57572	0.70192

Values less than 0.4 are not printed.

When factors are correlated, the common variance explained by each factor is not as straightforward as it is when the factors are uncorrelated. You must either eliminate or ignore the contributions from the other factors. Both computations are shown in Output 10.4.15.

Output 10.4.15 Variance Explained

Variance Explained by Each Factor, Ignoring Other Factors		
Factor	**Weighted**	**Unweighted**
Factor1	14.341750	5.349164
Factor2	15.895579	4.506741

Variance Explained by Each Factor, Eliminating Other Factors		
Factor	**Weighted**	**Unweighted**
Factor1	8.211324	3.132449
Factor2	9.765153	2.290025

Finally, the final communality estimates for the rotated solution are shown in Output 10.4.16.

Output 10.4.16 Variance Explained

Final Communality Estimates and Variable Weights

Total Communality:
Weighted = 24.106903 Unweighted = 7.639189

Variable	Communality	Weight
Communication Skills	0.55727959	2.25920908
Problem Solving	0.44207153	1.79275764
Learning Ability	0.53650447	2.15791760
Judgment under Pressure	0.56450349	2.29653380
Observational Skills	0.71795868	3.54514379
Willingness to Confront Problems	0.68375220	3.16130592
Interest in People	0.80283089	5.07138167
Interpersonal Sensitivity	0.82319143	5.65640994
Desire for Self-Improvement	0.56953539	2.32303202
Appearance	0.34463311	1.52589948
Dependability	0.63877379	2.76805050
Physical Ability	0.27320906	1.37572027
Integrity	0.68494568	3.17353979

To fully investigate the possibility that three factors rather than just two should be extracted, you can repeat the preceding analysis and change only the number of factors. The following code performs such an analysis:

```
proc efa data=mylib.jobratings nfactors=3 method=ml
   rotate=quartimin reorder fuzz=0.4;
run;
```

Much of the initial analysis is the same as when two factors are extracted. First, the prior communalities are computed and used to form the reduced correlation matrix. This reduced correlation matrix is the input for an iterative process in which the final communalities and the initial factor pattern are computed.

The tables in Output 10.4.17 show the tests of significance and measures of fit when three factors are extracted from the jobratings data set. You can see that Akaike's information criterion obtains a lower value for the three-factor model than for the two-factor model. On the other hand, Schwarz's Bayesian criterion obtains a lower value for the two-factor model. Thus there is still ambiguity regarding the number of factors that should be estimated from the data.

Output 10.4.17 Significance Tests and Measures of Fit (Three-Factor Model)

The EFA Procedure

Significance Tests Based on 103 Observations

Test	DF	Chi-Square	Pr > ChiSq
H0: No common factors	78	852.5129	<.0001
HA: At least one common factor			
H0: 3 Factors are sufficient	42	63.3931	0.0181
HA: More factors are needed			

Chi-Square without Bartlett's Correction	68.18380
Akaike's Information Criterion	-15.81620
Schwarz's Bayesian Criterion	-126.47482
Tucker and Lewis's Reliability Coefficient	0.94870

The initial factor pattern for the three-factor solution is shown in Output 10.4.18. As before, the quartimin rotation is applied to this initial factor pattern. The "Oblique Transformation Matrix" and "Interfactor Correlations" tables are shown in Output 10.4.19. For the three-factor solution, the first and third factors are moderately correlated with each other.

Output 10.4.18 Factor Pattern

Factor Pattern

	Factor1	Factor2	Factor3
Integrity	0.79639	.	.
Interest in People	0.79293	-0.42170	.
Dependability	0.78828	.	.
Judgment under Pressure	0.76129	.	.
Interpersonal Sensitivity	0.75185	-0.50923	.
Communication Skills	0.73337	.	.
Desire for Self-Improvement	0.70353	.	.
Problem Solving	0.67936	.	-0.48758
Observational Skills	0.61088	0.57367	.
Learning Ability	0.58935	0.46440	.
Appearance	0.51770	.	.
Physical Ability	0.48145	.	.
Willingness to Confront Problems	0.57538	0.58409	.

Values less than 0.4 are not printed.

Output 10.4.19 Oblique Transformation Matrix

The EFA Procedure

Oblique Transformation Matrix

	1	2	3
1	0.41909	0.58067	0.27177
2	0.82170	-0.92004	0.17477
3	0.79867	0.17078	-1.14008

Interfactor Correlations

	Factor1	Factor2	Factor3
Factor1	1.00000	0.39510	0.52069
Factor2	0.39510	1.00000	0.32454
Factor3	0.52069	0.32454	1.00000

The rotated factor pattern is shown in Output 10.4.20. This factor pattern is qualitatively similar to the rotated factor pattern for the two-factor solution. Each factor explains a subset of the analysis variables, and each analysis variable loads primarily on one or two of the common factors.

Output 10.4.20 Rotated Factor Pattern

Rotated Factor Pattern (Standardized Regression Coefficients)			
	Factor1	Factor2	Factor3
Observational Skills	0.88114	.	.
Willingness to Confront Problems	0.77606	.	.
Desire for Self-Improvement	0.65025	.	.
Dependability	0.54878	0.41841	.
Physical Ability	0.52291	.	.
Learning Ability	0.52041	.	.
Appearance	0.51472	.	.
Interpersonal Sensitivity	.	0.89025	.
Interest in People	.	0.86708	.
Integrity	.	0.71002	.
Judgment under Pressure	.	0.46714	0.44509
Problem Solving	.	.	0.77844
Communication Skills	.	.	0.45834

Values less than 0.4 are not printed.

Finally, the common variance explained by the factors is summarized in the tables in Output 10.4.15.

Output 10.4.21 Variance Explained

Variance Explained by Each Factor, Ignoring Other Factors		
Factor	Weighted	Unweighted
Factor1	15.184488	5.030941
Factor2	16.519540	4.038321
Factor3	11.157355	3.441515

Variance Explained by Each Factor, Eliminating Other Factors		
Factor	Weighted	Unweighted
Factor1	6.125759	2.123909
Factor2	10.228793	2.177506
Factor3	3.043214	0.904367

In this example, even after you extract and rotate a two-factor model and a three-factor model, there remains some ambiguity about which solution you should prefer. In this case, you might justify your decision by determining which solution is easier to interpret or is more consistent with existing theory or future experiments. You should also consider the possibility that the common factor model is not appropriate for the current data. In that case, you might consider a more complex model, such as a structural equation model.

References

Akaike, H. (1973). "Information Theory and an Extension of the Maximum Likelihood Principle." In *Proceedings of the Second International Symposium on Information Theory*, edited by B. N. Petrov and F. Csáki, 267–281. Budapest: Akademiai Kiado.

Akaike, H. (1974). "A New Look at the Statistical Model Identification." *IEEE Transactions on Automatic Control* AC-19:716–723.

Akaike, H. (1987). "Factor Analysis and AIC." *Psychometrika* 52:317–332.

Bickel, P. J., and Doksum, K. A. (1977). *Mathematical Statistics*. San Francisco: Holden-Day.

Cattell, R. B. (1978). *The Scientific Use of Factor Analysis*. New York: Plenum.

Crawford, C. B., and Ferguson, G. A. (1970). "A General Rotation Criterion and Its Use in Orthogonal Rotation." *Psychometrika* 35:321–332.

Cureton, E. E. (1968). *A Factor Analysis of Project TALENT Tests and Four Other Test Batteries.* Interim Report 4 to the U.S. Office of Education, Cooperative Research Project No. 3051. Palo Alto, CA: Project TALENT Office, American Institutes for Research and University of Pittsburgh.

Cureton, E. E., and Mulaik, S. A. (1975). "The Weighted Varimax Rotation and the Promax Rotation." *Psychometrika* 40:183–195.

Fuller, W. A. (1987). *Measurement Error Models*. New York: John Wiley & Sons.

Geweke, J., and Singleton, K. J. (1980). "Interpreting the Likelihood Ratio Statistic in Factor Models When Sample Size Is Small." *Journal of the American Statistical Association* 75:133–137.

Glorfeld, L. W. (1995). "An Improvement on Horn's Parallel Analysis Methodology for Selecting the Correct Number of Factors to Retain." *Educational and Psychological Measurement* 55:377–393.

Gorsuch, R. L. (1974). *Factor Analysis*. Philadelphia: W. B. Saunders.

Harman, H. H. (1976). *Modern Factor Analysis*. 3rd ed. Chicago: University of Chicago Press.

Horn, J. L. (1965). "A Rationale and Test for the Number of Factors in Factor Analysis." *Psychometrika* 30:179–185.

Jennrich, R. I. (1973). "Standard Errors for Obliquely Rotated Factor Loadings." *Psychometrika* 38:593–604.

Kim, J. O., and Mueller, C. W. (1978a). *Factor Analysis: Statistical Methods and Practical Issues.* Vol. 07-014 of Sage University Paper Series on Quantitative Applications in the Social Sciences. Beverly Hills, CA: Sage Publications.

Kim, J. O., and Mueller, C. W. (1978b). *Introduction to Factor Analysis: What It Is and How to Do It.* Vol. 07-013 of Sage University Paper Series on Quantitative Applications in the Social Sciences. Beverly Hills, CA: Sage Publications.

Lawley, D. N., and Maxwell, A. E. (1971). *Factor Analysis as a Statistical Method.* New York: Macmillan.

Mardia, K. V., Kent, J. T., and Bibby, J. M. (1979). *Multivariate Analysis*. London: Academic Press.

Morrison, D. F. (1976). *Multivariate Statistical Methods*. 2nd ed. New York: McGraw-Hill.

Mulaik, S. A. (1972). *The Foundations of Factor Analysis*. New York: McGraw-Hill.

Rao, C. R. (1955). "Estimation and Tests of Significance in Factor Analysis." *Psychometrika* 20:93–111.

Schwarz, G. (1978). "Estimating the Dimension of a Model." *Annals of Statistics* 6:461–464.

Spearman, C. (1904). "General Intelligence Objectively Determined and Measured." *American Journal of Psychology* 15:201–293.

Stewart, D. W. (1981). "The Application and Misapplication of Factor Analysis in Marketing Research." *Journal of Marketing Research* 18:51–62.

Tucker, L. R., and Lewis, C. (1973). "A Reliability Coefficient for Maximum Likelihood Factor Analysis." *Psychometrika* 38:1–10.

Velicer, W. F. (1976). "Determining the Number of Components from the Matrix of Partial Correlations." *Psychometrika* 41:321–327.

Velicer, W. F., Eaton, C. A., and Fava, J. L. (2000). "Construct Explication through Factor or Component Analysis: A Review and Evaluation of Alternative Procedures for Determining the Number of Factors or Components." In *Problems and Solutions in Human Assessment: Honoring Douglas N. Jackson at Seventy*, edited by R. D. Goffin and E. Helmes, 41–71. Boston: Kluwer.

Chapter 11
FREQTAB Procedure

Contents

Overview: FREQTAB Procedure **380**
PROC FREQTAB Features 381
PROC FREQTAB Compared with the FREQ Procedure 381
Using CAS Sessions and CAS Engine Librefs 382
Getting Started: FREQTAB Procedure **383**
Syntax: FREQTAB Procedure **390**
PROC FREQTAB Statement 390
BY Statement 393
EXACT Statement 393
OUTPUT Statement 403
TABLES Statement 415
TEST Statement 467
WEIGHT Statement 471
Details: FREQTAB Procedure **472**
Inputting Frequency Counts 472
Missing Values 473
Statistical Computations 475
Definitions and Notation 475
Chi-Square Tests and Statistics 476
Measures of Association 482
Binomial Proportion 491
Sensitivity and Specificity 500
Risks and Risk Differences 501
Common Risk Difference 511
Odds Ratio and Relative Risks 516
Cochran-Armitage Test for Trend 526
Jonckheere-Terpstra Test 527
Tests and Measures of Agreement 529
Cochran-Mantel-Haenszel Statistics 536
Gail-Simon Test for Qualitative Interactions 546
Exact Statistics 547
Output Data Sets 551
Displayed Output 554
ODS Table Names 565
ODS Graphics 569
References **570**

Overview: FREQTAB Procedure

The FREQTAB procedure produces one-way to *n*-way frequency and crosstabulation (contingency) tables, together with a variety of tests and measures to analyze these tables.

For one-way frequency tables, PROC FREQTAB provides chi-square goodness-of-fit tests for equal proportions and for specified null proportions. It also provides several types of confidence limits for binomial proportions and binomial proportion tests (which include noninferiority and equivalence tests).

For two-way crosstabulation tables, PROC FREQTAB computes a variety of statistics to examine the relationships between the two classification variables. These statistics include the following:

- chi-square tests and measures
- measures and tests of association
- measures and tests of agreement
- tests for trend
- Cochran-Mantel-Haenszel statistics

For 2×2 tables, PROC FREQTAB provides several confidence limit types and tests for the following statistics:

- risk (proportion) differences
- odds ratios
- relative risks

For multiway (*n*-way) tables, PROC FREQTAB performs stratified analysis by computing statistics within strata (two-way tables) and across strata. These statistics include Cochran-Mantel-Haenszel statistics and measures of agreement. For multiway 2×2 tables, PROC FREQTAB provides confidence limits and tests for the common (stratified) risk difference, odds ratio, and relative risks.

By default, PROC FREQTAB computes asymptotic standard errors, confidence limits, and tests. Exact tests and confidence limits are also available for many statistics and measures.

PROC FREQTAB Features

The FREQTAB procedure offers the following functionality:

- provides TABLES statements to specify frequency and crosstabulation tables
- provides options for statistical analyses of frequency and crosstabulation tables
- provides an EXACT statement for exact tests and confidence limits
- provides a TEST statement for tests of measures of association and agreement
- provides a WEIGHT statement for weighted analyses
- produces ODS tables of tabulation and statistical analysis results
- uses ODS graphics to produce statistical plots as part of the output

Because the FREQTAB procedure constructs the frequency and crosstabulation tables on CAS, it also does the following:

- enables you to construct tables on a cluster of machines that distribute the data and the computations
- enables you to construct tables in single-machine mode on CAS
- exploits all the available cores and concurrent threads for table construction. For information about how PROC FREQTAB uses threads, see the section "Multithreading" on page 102 in Chapter 2, "Shared Concepts."

PROC FREQTAB Compared with the FREQ Procedure

The FREQTAB procedure provides tabulation and statistical analysis capabilities that are very similar to the capabilities of the FREQ procedure in SAS/STAT software. Table 11.1 shows the major functional similarities and differences between these two procedures.

Table 11.1 Comparison of PROC FREQTAB and PROC FREQ

Feature	PROC FREQTAB	PROC FREQ
Provides statistical analysis of frequency and crosstabulation tables	Yes	Yes
Provides OUT=, OUTPUT, and ODS output data sets	Yes	Yes
Threading	Executes on multiple threads	Executes on a single thread
Supports MISSING option	In PROC FREQTAB statement	In TABLES statements
Supports SPARSE option	No	Yes
Supports CHISQ(TESTP=*data-table*)	No	Yes

Using CAS Sessions and CAS Engine Librefs

SAS Cloud Analytic Services (CAS) is the analytic server and associated cloud services in SAS Viya. This section describes how to create a CAS session and set up a CAS engine libref that you can use to connect to the CAS session. It assumes that you have a CAS server already available; contact your system administrator if you need help starting and terminating a server. This CAS server is identified by specifying the host on which it runs and the port on which it listens for communications. To simplify your interactions with this CAS server, the host information and port information for the server are stored as SAS option values that are retrieved automatically whenever this CAS server needs to be accessed. You can examine the host and port values for the server at your site by using the following statements:

```
proc options option=(CASHOST CASPORT);
run;
```

In addition to starting a CAS server, your system administrator might also have created a CAS session and a CAS engine libref for your use. You can define your own sessions and CAS engine librefs that connect to the CAS server as shown in the following statements:

```
cas mysess;
libname mylib cas sessref=mysess;
```

The CAS statement creates the CAS session named mysess, and the LIBNAME statement creates the mylib CAS engine libref that you use to connect to this session. It is not necessary to explicitly name the CASHOST and CASPORT of the CAS server in the CAS statement, because these values are retrieved from the corresponding SAS option values.

If you have created the mysess session, you can terminate it by using the TERMINATE option in the CAS statement as follows:

```
cas mysess terminate;
```

For more information about the CAS and LIBNAME statements, see the section “Introduction to Shared Concepts” on page 6 in Chapter 2, “Shared Concepts.”

Getting Started: FREQTAB Procedure

(View the complete code for this example at https://github.com/sascommunities/doc-supplement-casstat/tree/main/Examples/freqtgs.sas.)

This example shows you how to use the FREQTAB procedure to construct crosstabulation tables and produce chi-square statistics and measures of association.

The data table mylib.School_Survey consists of hypothetical data that are based on a customer satisfaction survey for a school information system. This data table includes the following variables for the survey respondents: ID (respondent identification), Response, SchoolType, and State. Figure 11.1 displays the first 20 observations of the data table.

Figure 11.1 School_Survey Data Table (First 20 Observations)

Obs	ID	Response	SchoolType	State
1	26629	Satisfied	Middle School (Public)	TN
2	25834	Neutral	High School (Public)	TN
3	14821	Very Satisfied	High School (Private)	NC
4	4153	Neutral	Middle School (Private)	GA
5	6185	Very Satisfied	High School (Private)	GA
6	6164	Unsatisfied	High School (Public)	GA
7	6386	Neutral	Middle School (Public)	GA
8	1011	Neutral	High School (Private)	GA
9	28916	Very Satisfied	High School (Private)	TN
10	10601	Neutral	High School (Public)	NC
11	17900	Satisfied	High School (Private)	NC
12	2935	Unsatisfied	High School (Private)	GA
13	9799	Unsatisfied	High School (Private)	NC
14	23754	Very Satisfied	Middle School (Public)	TN
15	9516	Neutral	Middle School (Public)	NC
16	4614	Very Unsatisfied	Middle School (Private)	GA
17	12443	Unsatisfied	Middle School (Public)	NC
18	5356	Very Satisfied	High School (Public)	GA
19	15189	Unsatisfied	High School (Private)	NC
20	3309	Satisfied	High School (Public)	GA

NOTE: Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections "Using CAS Sessions and CAS Engine Librefs" on page 7 and "Loading a SAS Data Set onto a CAS Server" on page 7 in Chapter 2, "Shared Concepts."

The following PROC FREQTAB statements request a two-way crosstabulation of SchoolType and Response. These statements assume that your libref is named mylib, but you can substitute any appropriately defined libref.

```
proc freqtab data=mylib.School_Survey;
   tables SchoolType * Response /
      crosslist chisq measures(cl);
run;
```

The PROC FREQTAB statement invokes the procedure and identifies the data table to be analyzed. The TABLES statement requests a two-way crosstabulation of SchoolType and Response. The table request syntax in PROC FREQTAB is identical to the table request syntax in the FREQ procedure. You can specify one-way, two-way, and multiway table requests. You can specify more than one table request in the same TABLES statement, and you can specify multiple TABLES statements in the same invocation of the procedure.

The CROSSLIST option displays the crosstabulation table in column format; by default, PROC FREQTAB displays crosstabulation tables in table cell format. The CHISQ option requests chi-square tests, and the MEASURES option requests measures of association for this two-way crosstabulation table. The MEASURES(CL) option requests confidence limits for the measures of association.

Figure 11.2 shows the crosstabulation table of SchoolType by Response. You can customize the contents of this table by specifying options in the TABLES statement. For example, the NOCOL option suppresses the column percentages, the EXPECTED option displays the expected cell frequencies (under independence), and the STDRES option displays the standardized residuals.

Figure 11.2 Crosstabulation Table

The FREQTAB Procedure

Table of SchoolType by Response

SchoolType	Response	Frequency	Percent	Row Percent	Column Percent
Middle School (Private)	Very Satisfied	594	1.83	15.77	12.05
	Satisfied	823	2.54	21.85	11.32
	Neutral	1183	3.64	31.41	12.51
	Unsatisfied	846	2.61	22.46	11.16
	Very Unsatisfied	320	0.99	8.50	9.93
	Total	3766	11.60	100.00	
Middle School (Public)	Very Satisfied	1261	3.88	13.94	25.58
	Satisfied	1811	5.58	20.02	24.91
	Neutral	2494	7.68	27.56	26.37
	Unsatisfied	2364	7.28	26.13	31.18
	Very Unsatisfied	1118	3.44	12.36	34.70
	Total	9048	27.88	100.00	
High School (Private)	Very Satisfied	1459	4.49	18.23	29.60
	Satisfied	2160	6.65	26.99	29.72
	Neutral	2422	7.46	30.26	25.61
	Unsatisfied	1363	4.20	17.03	17.98
	Very Unsatisfied	599	1.85	7.48	18.59
	Total	8003	24.66	100.00	
High School (Public)	Very Satisfied	1615	4.98	13.87	32.77
	Satisfied	2475	7.63	21.26	34.05
	Neutral	3359	10.35	28.85	35.51
	Unsatisfied	3008	9.27	25.84	39.68
	Very Unsatisfied	1185	3.65	10.18	36.78
	Total	11642	35.87	100.00	
Total	Very Satisfied	4929	15.19		100.00
	Satisfied	7269	22.39		100.00
	Neutral	9458	29.14		100.00
	Unsatisfied	7581	23.36		100.00
	Very Unsatisfied	3222	9.93		100.00
	Total	32459	100.00		

Figure 11.3 shows the chi-square statistics, and Figure 11.4 shows the measures of association for the table of SchoolType and Response.

Figure 11.3 Chi-Square Statistics

Statistics for Table of SchoolType by Response

Statistic	DF	Value	Prob
Chi-Square	12	505.1446	<.0001
Likelihood Ratio Chi-Square	12	512.2769	<.0001
Mantel-Haenszel Chi-Square	1	0.0050	0.9438
Phi Coefficient		0.1247	
Contingency Coefficient		0.1238	
Cramer's V		0.0720	

Figure 11.4 Measures of Association

Statistic	Value	ASE	95% Confidence Limits	
Gamma	0.0024	0.0061	-0.0097	0.0144
Kendall's Tau-b	0.0018	0.0046	-0.0073	0.0108
Stuart's Tau-c	0.0018	0.0046	-0.0072	0.0108
Somers' D C|R	0.0018	0.0048	-0.0075	0.0112
Somers' D R|C	0.0017	0.0044	-0.0070	0.0104
Pearson Correlation	-0.0004	0.0055	-0.0112	0.0104
Spearman Correlation	0.0018	0.0056	-0.0090	0.0127
Lambda Asymmetric C|R	0.0000	0.0000	0.0000	0.0000
Lambda Asymmetric R|C	0.0000	0.0000	0.0000	0.0000
Lambda Symmetric	0.0000	0.0000	0.0000	0.0000
Uncertainty Coefficient C|R	0.0051	0.0004	0.0042	0.0060
Uncertainty Coefficient R|C	0.0060	0.0005	0.0050	0.0070
Uncertainty Coefficient Symmetric	0.0055	0.0005	0.0046	0.0064

Sample Size = 32459

PROC FREQTAB can produce plots as part of its output by using ODS Graphics. The following statements show how to produce a frequency plot and a mosaic plot for a two-way crosstabulation table. The ODS GRAPHICS ON statement enables ODS Graphics, which is required in order to produce plots.

The PLOTS=FREQPLOT option in the TABLES statement requests a frequency plot of the two-way table of Response by SchoolType. The TWOWAY=CLUSTER *plot-option* requests a cluster grouped bar chart. The ORIENT=H *plot-option* specifies horizontal orientation, which places the variable levels on the Y-axis and the percentages on the X-axis. The SCALE=GROUPPERCENT *plot-option* requests a plot of group (column) percentages (instead of frequencies).

The PLOTS=MOSAICPLOT option requests a mosaic plot of Response by SchoolType. The COLORSTAT=STDRES *plot-option* colors the mosaic plot tiles according to the standardized residuals of the corresponding table cells.

```
ods graphics on;
proc freqtab data=mylib.School_Survey;
   tables Response * SchoolType /
      plots=(freqplot(twoway=cluster orient=h scale=grouppercent)
```

```
            mosaicplot(colorstat=stdres));
run;
ods graphics off;
```

Figure 11.5 displays the two-way frequency plot of SchoolType and Response. Figure 11.6 displays the mosaic plot of this crosstabulation.

Figure 11.5 Frequency Plot (Grouped Bar Chart)

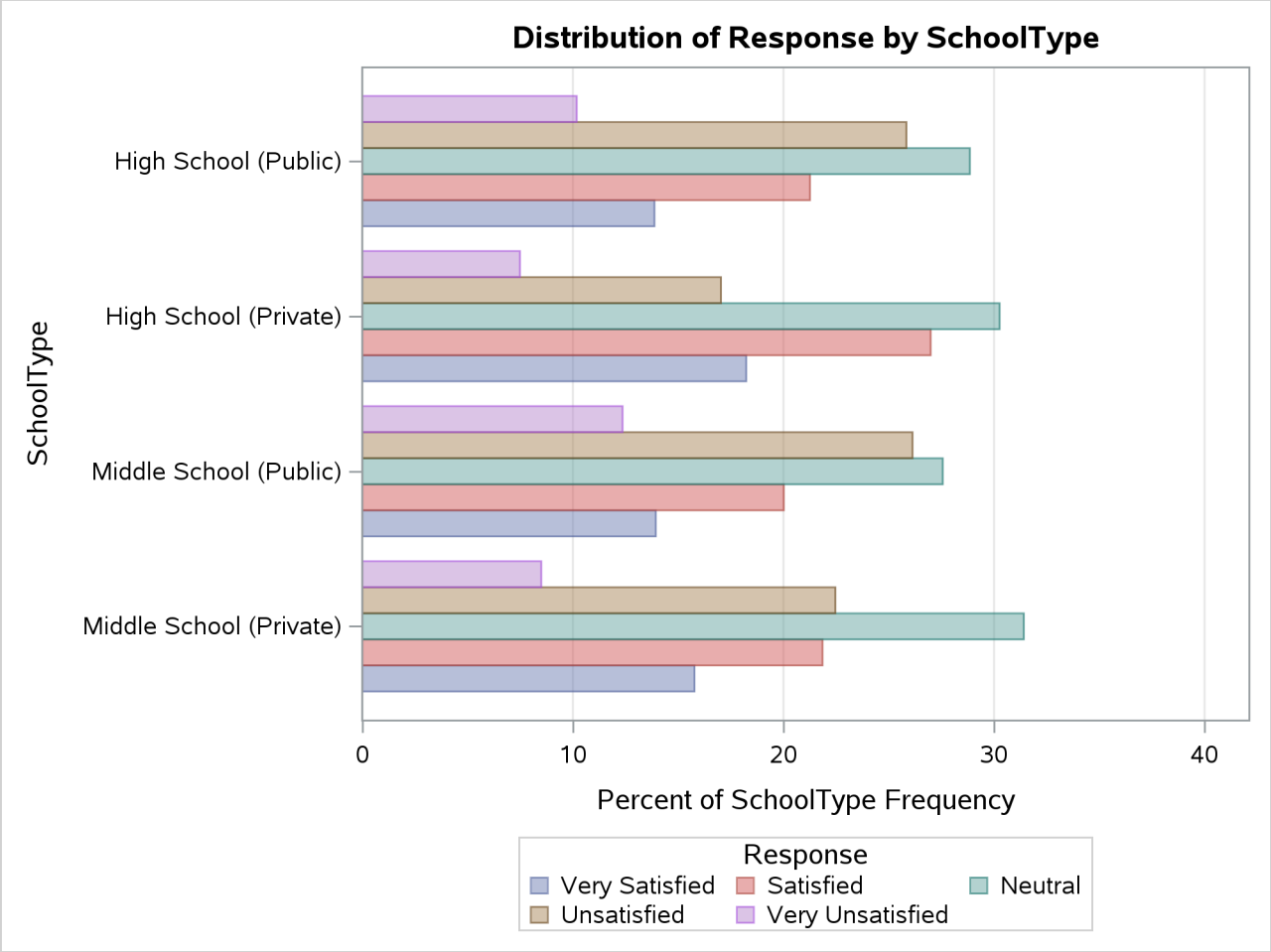
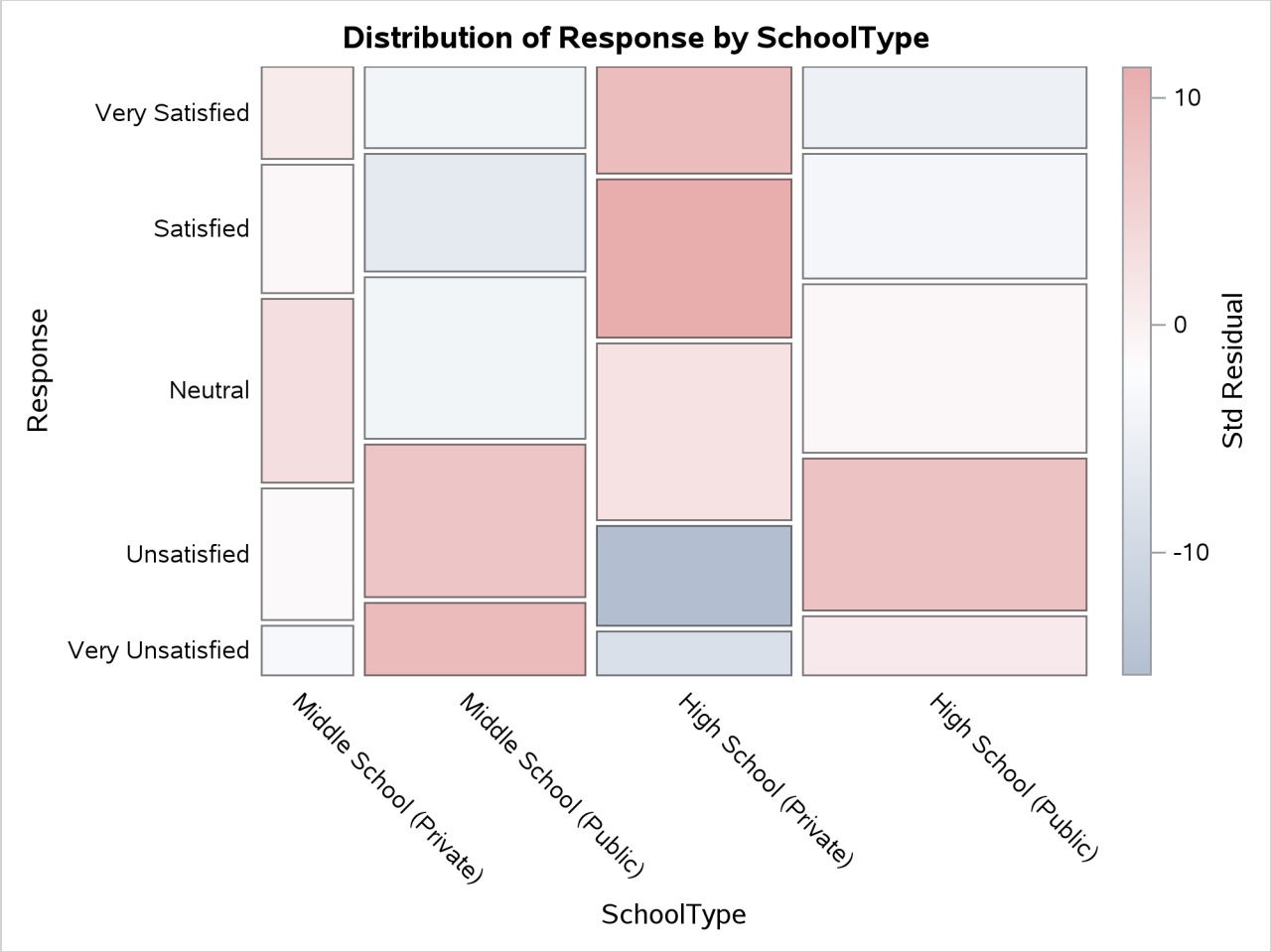


Figure 11.6 Mosaic Plot



The following PROC FREQTAB statements produce a multiway table of State by SchoolType by Response. The TABLES statement requests the multiway table in which SchoolType is the row variable, Response is the column variable, and State is the stratum variable. This request produces a separate two-way table of SchoolType by Response for each level (stratum) of the variable State. The CMH option in the TABLES statement requests stratified analysis, which adjusts for the effect of State.

```
proc freqtab data=mylib.School_Survey;
   tables State * SchoolType * Response / cmh;
run;
```

Figure 11.7 displays the two-way table of SchoolType and Response for State = 'GA'. PROC FREQTAB also produces two-way tables for the other states, but these results are not displayed here. Figure 11.8 displays the CMH summary statistics.

Figure 11.7 Two-Way Table for State=GA

The FREQTAB Procedure

Frequency Percent Row Pct Col Pct	Table 1 of SchoolType by Response Controlling for State=GA					
			Response			
SchoolType	Very Satisfied	Satisfied	Neutral	Unsatisfied	Very Unsatisfied	Total
Middle School (Private)	146	193	293	243	82	957
	1.77	2.34	3.56	2.95	1.00	11.63
	15.26	20.17	30.62	25.39	8.57	
	12.17	11.05	12.03	12.19	9.58	
Middle School (Public)	281	406	625	631	295	2238
	3.41	4.93	7.59	7.67	3.58	27.19
	12.56	18.14	27.93	28.19	13.18	
	23.42	23.24	25.67	31.66	34.46	
High School (Private)	369	537	651	357	174	2088
	4.48	6.52	7.91	4.34	2.11	25.37
	17.67	25.72	31.18	17.10	8.33	
	30.75	30.74	26.74	17.91	20.33	
High School (Public)	404	611	866	762	305	2948
	4.91	7.42	10.52	9.26	3.71	35.82
	13.70	20.73	29.38	25.85	10.35	
	33.67	34.97	35.56	38.23	35.63	
Total	1200	1747	2435	1993	856	8231
	14.58	21.22	29.58	24.21	10.40	100.00

Figure 11.8 CMH Test of No Association

The FREQTAB Procedure

Summary Statistics for SchoolType by Response Controlling for State

Cochran-Mantel-Haenszel Statistics (Based on Table Scores)				
Statistic	Alternative Hypothesis	DF	Value	Prob
1	Nonzero Correlation	1	0.0086	0.9262
2	Row Mean Scores Differ	3	406.7786	<.0001
3	General Association	12	508.7656	<.0001

Total Sample Size = 32459

Syntax: FREQTAB Procedure

The following statements are available in the FREQTAB procedure:

PROC FREQTAB < *options* > ;
BY *variables* ;
EXACT *statistic-options* < / *computation-options* > ;
OUTPUT < **OUT=***data-table* > *output-options* ;
TABLES *requests* < / *options* > ;
TEST *options* ;
WEIGHT *variable* < / *option* > ;

The PROC FREQTAB statement is the only required statement for the FREQTAB procedure. If you specify the following statements, PROC FREQTAB produces a one-way frequency table for each variable in the most recently created data table.

```
proc freqtab;
run;
```

Table 11.2 summarizes the basic functions of the procedure statements. The following sections provide detailed syntax information for the BY, EXACT, OUTPUT, TABLES, TEST, and WEIGHT statements in alphabetical order after the description of the PROC FREQTAB statement.

Table 11.2 Summary of PROC FREQTAB Statements

Statement	Description
BY	Provides separate analyses for each BY group
EXACT	Requests exact tests
OUTPUT	Requests an output data set of statistics
TABLES	Specifies tables and requests analyses
TEST	Requests tests for measures of association and agreement
WEIGHT	Identifies a weight variable

PROC FREQTAB Statement

PROC FREQTAB < *options* > ;

The PROC FREQTAB statement invokes the FREQTAB procedure. Optionally, it also identifies the input data table. By default, the procedure uses the most recently created data table.

Table 11.3 lists the *options* available in the PROC FREQTAB statement. Descriptions of the *options* follow in alphabetical order.

Table 11.3 PROC FREQTAB Statement Options

Option	Description
COMPRESS	Begins the next one-way table on the current page
DATA=	Names the input data table
FORMCHAR=	Specifies outline and cell-divider characters for cell-format crosstabulation tables
MISSING	Treats missing values as a valid level
MISSPRINT	Displays missing levels
NLEVELS	Displays the number of levels for each TABLES statement variable
NOPRINT	Suppresses all displayed output
ORDER=	Specifies the order for reporting variable values
PAGE	Displays one table per page

You can specify the following *options*:

COMPRESS
begins display of the next one-way frequency table on the same page as the preceding one-way table if there is enough space to begin the table. By default, the next one-way table begins on the current page only if the entire table fits on that page. The COMPRESS option is not valid with the PAGE option.

DATA=*libref.data-table*
names the input data table for PROC FREQTAB to use. The default is the most recently created data table. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to the data, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 382.

data-table specifies the name of the input data table.

FORMCHAR(1,2,7)=*'formchar-string'*
defines the characters to use to construct cell outlines and dividers for crosstabulation tables. This option affects only the SAS monospace output destination.

PROC FREQTAB uses 3 of the 20 formatting characters that SAS provides. You can specify three characters in *formchar-string* to draw the vertical separators (1), the horizontal separators (2), and the vertical-horizontal intersections (7) in crosstabulation tables. By default, PROC FREQTAB uses FORMCHAR(1,2,7)='|-+'. Table 11.4 summarizes the formatting characters that PROC FREQTAB uses.

Table 11.4 Formatting Characters Used by PROC FREQTAB

Position	Default	Used to Draw
1	|	Vertical separators
2	-	Horizontal separators
7	+	Intersections of vertical and horizontal separators

To produce crosstabulation tables that have no cell outlines or dividers, you can specify all blanks in *formchar-string*—for example, FORMCHAR(1,2,7)=‘ ’ (three blanks).

You can use any characters in *formchar-string*. If you use hexadecimal characters, you must put **x** after the closing quotation mark. For information about which hexadecimal codes to use for which characters, see the documentation for your hardware.

For more information about formatting characters, see the TABULATE procedure in the *Base SAS Procedures Guide*.

MISSING

treats missing values as a valid nonmissing level for all classification variables that you specify in the TABLES statements. This option displays missing levels in frequency and crosstabulation tables and includes the frequencies (of missing levels) in the computation of the frequency total, percentages, and statistics.

By default (if you do not specify the MISSING or MISSPRINT option), PROC FREQTAB excludes an observation from a table if the observation contains a missing value for any variable in the table request. For more information, see the section “Missing Values” on page 473.

MISSPRINT

displays missing levels in frequency tables but does not include the corresponding frequencies in the computation of totals, percentages, or statistics. This option is available for one-way frequency tables and for tables in list format (which you can request by specifying the LIST option in the TABLES statement).

By default (if you do not specify the MISSING or MISSPRINT option), PROC FREQTAB excludes an observation from a table if the observation contains a missing value for any variable in the table request. For more information, see the section “Missing Values” on page 473.

NLEVELS

displays the “Number of Variable Levels” table, which provides the number of levels for each variable that you name in a TABLES statement. PROC FREQTAB determines the variable levels by using the formatted variable values.

NOPRINT

suppresses the display of all output. You can use the NOPRINT option when you only want to create an output data set. For more information about the output data sets that PROC FREQTAB produces, see the section “Output Data Sets” on page 551. The NOPRINT option temporarily disables the Output Delivery System (ODS). For more information, see the *SAS Output Delivery System: User’s Guide*.

NOTE: A NOPRINT option is also available in the TABLES statement. This option suppresses display of the crosstabulation tables but allows display of the statistics tables.

ORDER=< FORMATTED | FREQ | INTERNAL > < (*variable1*='*level1*' '*level2*' ... *variable2*= ...) >

specifies the order of the variable levels in the frequency and crosstabulation tables, which you request in the TABLES statement. The following table shows how to interpret the ORDER= keywords:

Value of ORDER=	Levels Ordered By
FORMATTED	External formatted value (in ascending order)
FREQ	Descending frequency count
INTERNAL	Unformatted value

By default, ORDER=INTERNAL. The formatted and internal orders are machine-dependent. When you specify ORDER=FORMATTED, numeric variables that have no explicit format are sorted by their unformatted (internal) value.

When you specify ORDER=FREQ, levels that have larger frequencies (more observations) appear first in the order. ORDER=FREQ is not available when you specify a WEIGHT statement.

You can also specify a particular order for variables by listing, in parentheses, each variable followed by its levels in quotes in the desired order. For example, the following option specifies that the levels of all variables except the pet variable are to be sorted by their formatted value:

```
order = formatted (pet = 'dog' 'cat')
```

Any level of the pet variable that you do not specify is added to the end of the list in formatted order.

The ORDER= option does not apply to missing values, which are always ordered first.

For more information about sort order, see the chapter on the SORT procedure in the *Base SAS Procedures Guide* and the discussion of BY-group processing in *SAS Programmers Guide: Essentials*.

PAGE

displays only one table per page. Otherwise, PROC FREQTAB displays multiple tables per page as space permits. The PAGE option is not valid when you specify the COMPRESS option.

BY Statement

BY *variables* **;**

You can specify a BY statement in PROC FREQTAB to obtain separate analyses of observations in groups that are defined by the values of the BY variables. If you specify more than one BY statement, only the last one specified is used. For more information, see the discussion of BY-group processing in the "Grouping Data" section of *SAS Programmers Guide: Essentials*.

EXACT Statement

EXACT *statistic-options* < / *computation-options* > **;**

The EXACT statement requests exact tests and confidence limits for selected statistics. The *statistic-options* identify which statistics to compute, and the *computation-options* specify options for computing exact statistics. For more information, see the section "Exact Statistics" on page 547.

NOTE: PROC FREQTAB computes exact tests by using fast and efficient algorithms that are superior to direct enumeration. Exact tests are appropriate when a data set is small, sparse, skewed, or heavily tied. For some large problems, computation of exact tests might require a large amount of time and memory. Consider using asymptotic tests for such problems. Alternatively, when asymptotic methods might not be sufficient for such large problems, consider using Monte Carlo estimation of exact *p*-values. You can request Monte Carlo estimation by specifying the MC *computation-option* in the EXACT statement. For more information, see the section "Computational Resources" on page 549.

Statistic Options

The *statistic-options* specify which exact tests and confidence limits to compute. Table 11.5 lists the available *statistic-options* and the exact statistics that are computed. Descriptions of the *statistic-options* follow Table 11.5 in alphabetical order.

For one-way tables, PROC FREQTAB provides exact p-values for the binomial proportion test, the chi-square goodness-of-fit test, and the likelihood ratio chi-square test. PROC FREQTAB also provides exact (Clopper-Pearson) confidence limits for the binomial proportion.

For two-way tables, PROC FREQTAB provides exact p-values for the following tests: Pearson chi-square test, likelihood ratio chi-square test, Mantel-Haenszel chi-square test, Fisher's exact test, Jonckheere-Terpstra test, Cochran-Armitage test for trend, and the symmetry test. PROC FREQTAB also provides exact p-values for tests of the following statistics: Pearson correlation coefficient, Spearman correlation coefficient, Kendall's tau-b, Stuart's tau-c, Somers' $D(C|R)$, Somers' $D(R|C)$, simple kappa coefficient, and weighted kappa coefficient.

For 2×2 tables, PROC FREQTAB provides the exact McNemar's test, exact confidence limits for the odds ratio, and Barnard's unconditional exact test for the risk (proportion) difference. PROC FREQTAB also provides exact unconditional confidence limits for the risk (proportion) difference and for the relative risk (ratio of proportions). For stratified 2×2 tables, PROC FREQTAB provides Zelen's exact test for equal odds ratios, exact confidence limits for the common odds ratio, and an exact test for the common odds ratio.

Most of the *statistic-option* names listed in Table 11.5 are identical to the corresponding option names in the TABLES and OUTPUT statements. You can request exact computations for groups of statistics by using *statistic-options* that are identical to the TABLES statement options CHISQ, MEASURES, and AGREE. For example, when you specify the CHISQ *statistic-option* in the EXACT statement, PROC FREQTAB computes exact p-values for the Pearson chi-square, likelihood ratio chi-square, and Mantel-Haenszel chi-square tests for two-way tables. You can request an exact test for an individual statistic by specifying the corresponding *statistic-option* from the list in Table 11.5.

Using the EXACT Statement with the TABLES Statement

You must use a TABLES statement with the EXACT statement. If you use only one TABLES statement, you do not need to specify the same options in both the TABLES and EXACT statements; when you specify a *statistic-option* in the EXACT statement, PROC FREQTAB automatically invokes the corresponding TABLES statement option. However, when you use an EXACT statement with multiple TABLES statements, you must specify options in the TABLES statements to request statistics. PROC FREQTAB then provides exact tests or confidence limits for those statistics that you also specify in the EXACT statement.

Table 11.5 EXACT Statement Statistic Options

Statistic Option	Exact Statistics
AGREE	McNemar's test (for 2×2 tables), simple kappa test, weighted kappa test
BARNARD	Barnard's test (for 2×2 tables)
BINOMIAL | BIN	Binomial proportion tests for one-way tables
CHISQ	Chi-square goodness-of-fit test for one-way tables; Pearson chi-square, likelihood ratio chi-square, and Mantel-Haenszel chi-square tests for two-way tables

Table 11.5 *continued*

Statistic Option	Exact Statistics
COMOR	Confidence limits for the common odds ratio, common odds ratio test (for $h \times 2 \times 2$ tables)
EQOR | ZELEN	Zelen's test for equal odds ratios (for $h \times 2 \times 2$ tables)
FISHER	Fisher's exact test
JT	Jonckheere-Terpstra test
KAPPA	Test for the simple kappa coefficient
KENTB | TAUB	Test for Kendall's tau-b
LRCHI	Likelihood ratio chi-square test (one-way and two-way tables)
MCNEM	McNemar's test (for 2×2 tables)
MEASURES	Tests for the Pearson correlation and Spearman correlation, confidence limits for the odds ratio (for 2×2 tables)
MHCHI	Mantel-Haenszel chi-square test
OR | ODDSRATIO	Confidence limits for the odds ratio (for 2×2 tables)
PCHI	Pearson chi-square test (one-way and two-way tables)
PCORR	Test for the Pearson correlation coefficient
RELRISK	Confidence limits for the relative risk (for 2×2 tables)
RISKDIFF	Confidence limits for the risk difference (for 2×2 tables)
SCORR	Test for the Spearman correlation coefficient
SMDCR	Test for Somers' $D(C \mid R)$
SMDRC	Test for Somers' $D(R \mid C)$
STUTC | TAUC	Test for Stuart's tau-c
SYMMETRY | BOWKER	Symmetry test
TREND	Cochran-Armitage test for trend
WTKAPPA | WTKAP	Test for the weighted kappa coefficient

You can specify the following *statistic-options*:

AGREE

requests McNemar's exact test, an exact test for the simple kappa coefficient, and an exact test for the weighted kappa coefficient. For more information, see the sections "Tests and Measures of Agreement" on page 529 and "Exact Statistics" on page 547.

For McNemar's test, you can specify the null hypothesis ratio of discordant proportions by using the AGREE(MNULLRATIO=) option in the TABLES statement; by default, MNULLRATIO=1. For the weighted kappa coefficient, you can request Fleiss-Cohen weights by specifying the AGREE(WT=FC) option in the TABLES statement; by default, PROC FREQTAB computes the weighted kappa coefficient by using Cicchetti-Allison agreement weights.

McNemar's test is available for 2×2 tables. Kappa coefficients are defined only for square two-way tables, where the number of rows equals the number of columns. If your table is not square because some observations have weights of 0, you can specify the ZEROS option in the WEIGHT statement to include these observations in the analysis. For more information, see the section "Tables with Zero-Weight Rows or Columns" on page 536.

For 2×2 tables, the weighted kappa coefficient is equivalent to the simple kappa coefficient, and PROC FREQTAB displays only analyses for the simple kappa coefficient.

BARNARD

requests Barnard's exact unconditional test for the risk (proportion) difference for 2×2 tables. For more information, see the section "Barnard's Unconditional Exact Test" on page 510.

To request exact unconditional confidence limits for the risk difference. you can specify the RISKDIFF option in the EXACT statement. The RISKDIFF option in the TABLES statement provides asymptotic tests and several types of confidence limits for the risk difference. For more information, see the section "Risks and Risk Differences" on page 501.

BINOMIAL

BIN

requests an exact test for the binomial proportion (for one-way tables). For more information, see the section "Binomial Tests" on page 495. You can specify the null hypothesis proportion by using the BINOMIAL(P=) option in the TABLES statement; by default, P=0.5.

The BINOMIAL option in the TABLES statement provides exact (Clopper-Pearson) confidence limits for the binomial proportion by default. You can specify the BINOMIAL(CL=MIDP) option in the TABLES statement to request exact mid-p confidence limits for the binomial proportion. The BINOMIAL option in the TABLES statement also provides asymptotic (Wald) tests and several other confidence limit types for the binomial proportion. For more information, see the section "Binomial Proportion" on page 491.

CHISQ

requests the following exact chi-square tests for two-way tables: Pearson chi-square, likelihood ratio chi-square, and Mantel-Haenszel chi-square. For more information, see the section "Chi-Square Tests and Statistics" on page 476. The CHISQ option in the TABLES statement provides asymptotic tests for these statistics.

For one-way tables, the CHISQ option requests an exact chi-square goodness-of-fit test. You can specify null hypothesis proportions for this test by using the CHISQ(TESTP=) option in the TABLES statement. By default, the one-way chi-square test is based on the null hypothesis of equal proportions. For more information, see the section "Chi-Square Test for One-Way Tables" on page 477.

COMOR

requests an exact test and exact confidence limits for the common odds ratio for multiway 2×2 tables. For more information, see the section "Exact Confidence Limits for the Common Odds Ratio" on page 544. The CMH option in the TABLES statement provides Mantel-Haenszel and logit estimates of the common odds ratio along with their asymptotic confidence limits.

EQOR

ZELEN

requests Zelen's exact test for equal odds ratios for multiway 2×2 tables. For more information, see the section "Zelen's Exact Test for Equal Odds Ratios" on page 543. The CMH option in the TABLES statement provides an (asymptotic) Breslow-Day test for homogeneity of odds ratios.

FISHER

requests Fisher's exact test. For more information, see the sections "Fisher's Exact Test" on page 480 and "Exact Statistics" on page 547. For 2×2 tables, the CHISQ option in the TABLES statement provides Fisher's exact test. For general $R \times C$ tables, Fisher's exact test is also known as the Freeman-Halton test.

JT

requests an exact Jonckheere-Terpstra test. For more information, see the sections "Jonckheere-Terpstra Test" on page 527 and "Exact Statistics" on page 547. The JT option in the TABLES statement provides an asymptotic Jonckheere-Terpstra test.

KAPPA

requests an exact test for the simple kappa coefficient. For more information, see the sections "Simple Kappa Coefficient" on page 530 and "Exact Statistics" on page 547. The AGREE option in the TABLES statement provides the simple kappa estimate, standard error, and confidence limits. The KAPPA option in the TEST statement provides an asymptotic test for the simple kappa coefficient.

Kappa coefficients are defined only for square two-way tables, where the number of rows equals the number of columns. If your table is not square because some observations have weights of 0, you can specify the ZEROS option in the WEIGHT statement to include these observations in the analysis. For more information, see the section "Tables with Zero-Weight Rows or Columns" on page 536.

KENTB

TAUB

requests an exact test for Kendall's tau-*b*. For more information, see the sections "Kendall's Tau-b" on page 484 and "Exact Statistics" on page 547. The MEASURES option in the TABLES statement provides an estimate and standard error of Kendall's tau-*b*. The KENTB option in the TEST statement provides an asymptotic test for Kendall's tau-*b*.

LRCHI

requests an exact test for the likelihood ratio chi-square for two-way tables. For more information, see the sections "Likelihood Ratio Chi-Square Test" on page 479 and "Exact Statistics" on page 547. The CHISQ option in the TABLES statement provides an asymptotic likelihood ratio chi-square test for two-way tables.

For one-way tables, the LRCHI option requests an exact likelihood ratio goodness-of-fit test. You can specify null hypothesis proportions by using the CHISQ(TESTP=) option in the TABLES statement. By default, the one-way test is based on the null hypothesis of equal proportions. For more information, see the section "Likelihood Ratio Chi-Square Test for One-Way Tables" on page 478.

MCNEM

requests an exact McNemar's test. For more information, see the sections "McNemar's Test" on page 529 and "Exact Statistics" on page 547. You can specify the null hypothesis ratio of discordant proportions by using the AGREE(MNULLRATIO=) option in the TABLES statement; by default, MNULLRATIO=1. The AGREE option in the TABLES statement provides an asymptotic McNemar's test.

MEASURES

requests exact tests for the Pearson and Spearman correlations. For more information, see the sections "Pearson Correlation Coefficient" on page 485, "Spearman Rank Correlation Coefficient" on page 486, and "Exact Statistics" on page 547. The PCORR and SCORR options in the TEST statement provide asymptotic tests for the Pearson and Spearman correlations, respectively.

The MEASURES option also requests exact confidence limits for the odds ratio for 2×2 tables. For more information, see the subsection Exact Confidence Limits in the section "Confidence Limits for the Odds Ratio" on page 516. You can also request exact confidence limits for the odds ratio by specifying the OR option in the EXACT statement.

MHCHI

requests an exact test for the Mantel-Haenszel chi-square. For more information, see the sections "Mantel-Haenszel Chi-Square Test" on page 480 and "Exact Statistics" on page 547. The CHISQ option in the TABLES statement provides an asymptotic Mantel-Haenszel chi-square test.

OR

ODDSRATIO

requests exact confidence limits for the odds ratio for 2×2 tables. For more information, see the subsection "Exact Confidence Limits" in the section "Confidence Limits for the Odds Ratio" on page 516.

You can request exact mid-p confidence limits for the odds ratio by specifying the OR(CL=MIDP) option in the TABLES statement. The OR(CL=) option in the TABLES statement also provides other types of confidence limits for the odds ratio. For more information, see the section "Confidence Limits for the Odds Ratio" on page 516.

The ALPHA= option in the TABLES statement determines the confidence level of the exact confidence limits; by default, ALPHA=0.05, which produces 95% confidence limits for the odds ratio.

PCHI

requests an exact test for the Pearson chi-square for two-way tables. For more information, see the sections "Pearson Chi-Square Test for Two-Way Tables" on page 477 and "Exact Statistics" on page 547. The CHISQ option in the TABLES statement provides an asymptotic Pearson chi-square test.

For one-way tables, the PCHI option requests an exact chi-square goodness-of-fit test. You can specify null hypothesis proportions by using the CHISQ(TESTP=) option in the TABLES statement. By default, the goodness-of-fit test is based on the null hypothesis of equal proportions. For more information, see the section "Chi-Square Test for One-Way Tables" on page 477.

PCORR

requests an exact test for the Pearson correlation coefficient. For more information, see the sections "Pearson Correlation Coefficient" on page 485 and "Exact Statistics" on page 547. The MEASURES option in the TABLES statement provides the estimate and standard error of the Pearson correlation. The PCORR option in the TEST statement provides an asymptotic test for the Pearson correlation.

RELRISK < (*options*) >

requests exact unconditional confidence limits for the relative risk for 2×2 tables. By default, the exact confidence limits are computed by inverting two separate one-sided exact tests that are based on the score statistic (Chan and Zhang 1999). For more information, see the subsection "Exact Unconditional Confidence Limits" in the section "Confidence Limits for the Relative Risk" on page 519.

The RELRISK(CL=) option in the TABLES statement provides additional types of confidence limits for the relative risk. For more information, see the section "Confidence Limits for the Risk Difference" on page 502.

The ALPHA= option in the TABLES statement determines the confidence level; by default, ALPHA=0.05, which produces 95% confidence limits for the relative risk.

You can specify the following *options*:

COLUMN=1 | 2 | BOTH
specifies the table column of the relative risk. By default, COLUMN=1, which provides exact confidence limits for the column 1 relative risk. COLUMN=BOTH provides exact confidence limits for both column 1 and column 2 relative risks.

METHOD=NOSCORE | SCORE | SCORE2
specifies the computation method for the exact confidence limits. By default, METHOD=SCORE.

You can specify one of the following methods:

NOSCORE
computes the exact confidence limits by inverting two separate one-sided exact tests that are based on the unstandardized relative risk (Santner and Snell 1980). For more information, see the subsection "Exact Unconditional Confidence Limits" in the section "Confidence Limits for the Relative Risk" on page 519.

SCORE
computes the exact confidence limits by inverting two separate one-sided exact tests that are based on the score statistic (Chan and Zhang 1999). For more information, see the subsection "Exact Unconditional Confidence Limits" in the section "Confidence Limits for the Relative Risk" on page 519.

SCORE2
computes the exact confidence limits by inverting a single two-sided exact test that is based on the score statistic (Agresti and Min 2001). For more information, see the subsection "Exact Unconditional Confidence Limits" in the section "Confidence Limits for the Relative Risk" on page 519.

RISKDIFF < (*options*) >
requests exact unconditional confidence limits for the risk difference for 2×2 tables. By default, the exact confidence limits are computed by inverting two separate one-sided exact tests that are based on the score statistic (Chan and Zhang 1999). For more information, see the subsection "Exact Unconditional Confidence Limits" in the section "Confidence Limits for the Risk Difference" on page 502.

The RISKDIFF(CL=) option in the TABLES statement provides additional types of confidence limits for the risk difference. For more information, see the section "Confidence Limits for the Risk Difference" on page 502.

The ALPHA= option in the TABLES statement determines the confidence level; by default, ALPHA=0.05, which produces 95% confidence limits for the risk difference.

You can specify the following *options*:

COLUMN=1 | 2 | BOTH
specifies the table column of the risk difference. By default, COLUMN=BOTH and the exact confidence limits are displayed in the 'Risk Estimates' tables. If you specify the RISKDIFF(NORISKS) option in the TABLES statement to suppress the 'Risk Estimates' tables, by default, COLUMN=1 and the exact confidence limits are displayed in the 'Risk Difference Confidence Limits' table.

METHOD=NOSCORE | SCORE | SCORE2

specifies the computation method for the exact confidence limits. By default, METHOD=SCORE.

You can specify one of the following methods:

NOSCORE

computes the exact confidence limits by inverting two separate one-sided exact tests that are based on the unstandardized risk difference (Santner and Snell 1980). For more information, see the subsection "Exact Unconditional Confidence Limits" in the section "Confidence Limits for the Risk Difference" on page 502.

SCORE

computes the exact confidence limits by inverting two separate one-sided exact tests that are based on the score statistic (Chan and Zhang 1999). For more information, see the subsection "Exact Unconditional Confidence Limits" in the section "Confidence Limits for the Risk Difference" on page 502.

SCORE2

computes the exact confidence limits by inverting a single two-sided exact test that is based on the score statistic (Agresti and Min 2001). For more information, see the subsection "Exact Unconditional Confidence Limits" in the section "Confidence Limits for the Risk Difference" on page 502.

SCORR

requests an exact test for the Spearman correlation coefficient. For more information, see the sections "Spearman Rank Correlation Coefficient" on page 486 and "Exact Statistics" on page 547. The MEASURES option in the TABLES statement provides the estimate and standard error of the Spearman correlation. The SCORR option in the TEST statement provides an asymptotic test for the Spearman correlation.

SMDCR

requests an exact test for Somers' $D(C|R)$. For more information, see the sections "Somers' D" on page 485 and "Exact Statistics" on page 547. The MEASURES option in the TABLES statement provides the estimate and standard error of Somers' $D(C|R)$. The SMDCR option in the TEST statement provides an asymptotic test for Somers' $D(C|R)$.

SMDRC

requests an exact test for Somers' $D(R|C)$. For more information, see the sections "Somers' D" on page 485 and "Exact Statistics" on page 547. The MEASURES option in the TABLES statement provides the estimate and standard error of Somers' $D(R|C)$. The SMDRC option in the TEST statement provides an asymptotic test for Somers' $D(C|R)$.

STUTC

TAUC

requests an exact test for Stuart's tau-c. For more information, see the sections "Stuart's Tau-c" on page 484 and "Exact Statistics" on page 547. The MEASURES option in the TABLES statement provides the estimate and standard error of Stuart's tau-c. The STUTC option in the TEST statement provides an asymptotic test for Stuart's tau-c.

SYMMETRY

BOWKER

requests an exact symmetry test. This test is available for square $R \times R$ two-way tables where the table dimension R is greater than 2. For more information, see the section "Exact Symmetry Test" on page 530. The AGREE option in the TABLES statement provides an asymptotic symmetry test.

TREND

requests the exact Cochran-Armitage test for trend. For more information, see the sections "Cochran-Armitage Test for Trend" on page 526 and "Exact Statistics" on page 547. The TREND option in the TABLES statement provides an asymptotic Cochran-Armitage test for trend. This test is available for tables of dimensions $2 \times C$ or $R \times 2$.

WTKAPPA

WTKAP

requests an exact test for the weighted kappa coefficient. For more information, see the sections "Weighted Kappa Coefficient" on page 532 and "Exact Statistics" on page 547. By default, PROC FREQTAB computes the weighted kappa coefficient by using Cicchetti-Allison agreement weights. You can request Fleiss-Cohen agreement weights by specifying the AGREE(WT=FC) option in the TABLES statement.

Kappa coefficients are defined only for square two-way tables, where the number of rows equals the number of columns. If your table is not square because some observations have weights of 0, you can specify the ZEROS option in the WEIGHT statement to include these observations in the analysis. For more information, see the section "Tables with Zero-Weight Rows or Columns" on page 536.

For 2×2 tables, the weighted kappa coefficient is equivalent to the simple kappa coefficient, and PROC FREQTAB displays only analyses for the simple kappa coefficient.

Computation Options

The *computation-options* specify options for computing exact statistics. You can specify the following *computation-options*:

ALPHA=α

specifies the level of the confidence limits for Monte Carlo p-value estimates. The value of α must be between 0 and 1; a confidence level of α produces $100(1-\alpha)\%$ confidence limits. By default ALPHA=0.01, which produces 99% confidence limits for the Monte Carlo estimates.

This option invokes the MC option.

MAXTIME=*value*

specifies the maximum clock time (in seconds) that PROC FREQTAB can use to compute an exact p-value. If the procedure does not complete the computation within the specified time, the computation terminates. The maximum time *value* must be a positive number. This option is available for exact p-value computation and for Monte Carlo estimation of exact p-values. For more information, see the section "Computational Resources" on page 549.

MC

requests Monte Carlo estimation of exact *p*-values instead of direct exact *p*-value computation. Monte Carlo estimation can be useful for large problems where exact computations require a substantial amount of time and memory but asymptotic approximations might not be sufficient. For more information, see the section "Monte Carlo Estimation" on page 550.

This option is available for all EXACT *statistic-options* except the BINOMIAL option and the following options that apply only to 2×2 or $h \times 2 \times 2$ tables: BARNARD, COMOR, EQOR, MCNEM, OR, RELRISK, and RISKDIFF. PROC FREQTAB always computes exact tests or confidence limits (not Monte Carlo estimates) for these statistics.

The ALPHA=, N=, and SEED= options invoke the MC option.

MIDP

requests exact mid *p*-values for the exact tests. The exact mid *p*-value is defined as the exact *p*-value minus half the exact point probability. For more information, see the section "Definition of *p*-Values" on page 548.

The MIDP option is available for all EXACT statement *statistic-options* except the following: BARNARD, EQOR, OR, RELRISK, and RISKDIFF. You cannot specify both the MIDP option and the MC option.

N=*n*

specifies the number of samples for Monte Carlo estimation. The value of *n* must be a positive integer. Larger values of *n* produce more precise estimates of exact *p*-values. Because larger values of *n* generate more samples, the computation time increases. By default, N=10,000.

This option invokes the MC option.

PFORMAT=*format-name* **| EXACT**

specifies the display format for exact *p*-values. PROC FREQTAB applies this format to one- and two-sided exact *p*-values, exact point probabilities, and exact mid *p*-values. By default, PROC FREQTAB displays exact *p*-values in the PVALUE6.4 format.

You can provide a *format-name* or you can specify PFORMAT=EXACT to control the format of exact *p*-values. The value of *format-name* can be any standard SAS numeric format or a user-defined format. The format length must not exceed 24. For more information, see the discussion of the FORMAT procedure in the *Base SAS Procedures Guide* and the discussion of SAS formats in *SAS Formats and Informats: Reference*.

If you specify PFORMAT=EXACT, PROC FREQTAB uses the 6.4 format to display exact *p*-values that are greater than or equal to 0.001; the procedure uses the E10.3 format to display values that are between 0.000 and 0.001.

POINT

requests exact point probabilities for the exact tests. The exact point probability is the exact probability that the test statistic equals the observed value. For more information, see the section "Definition of *p*-Values" on page 548.

The POINT option is available for all EXACT statement *statistic-options* except the following: BARNARD, EQOR, OR, RELRISK, and RISKDIFF. You cannot specify both the POINT option and the MC option.

SEED=*number*

specifies the initial seed for random number generation for Monte Carlo estimation. The value of the SEED= option must be an integer. If you do not specify the SEED= option or if the SEED= value is negative or 0, PROC FREQTAB uses the time of day from the computer's clock to obtain the initial seed.

This option invokes the MC option.

OUTPUT Statement

OUTPUT <**OUT=***data-table*> *output-options* **;**

The OUTPUT statement creates an output data table that stores statistics that PROC FREQTAB computes. Table 11.6 lists the statistics that are available in the output data table. You can identify which statistics to include by specifying *output-options*.

When you specify an OUTPUT statement, you must also specify a TABLES statement, which defines tables and requests statistical analyses. The OUTPUT statement stores statistics for a single table request. If you use multiple TABLES statements, the contents of the output data set correspond to the last TABLES statement. If you use multiple table requests in a single TABLES statement, the contents of the output data set correspond to the last table request. Only one OUTPUT statement is allowed in a single invocation of PROC FREQTAB.

For a one-way or two-way table, the output data set contains one observation that stores statistics for the table. For a multiway table, the output data set contains an observation for each two-way table (stratum) of the multiway table. If you request summary statistics for the multiway table, the output data set also contains an observation that includes the across-strata summary statistics. If you specify a BY statement, the output data set contains an observation or set of observations for each BY group. For more information, see the section "Contents of the OUTPUT Statement Output Data Set" on page 553.

The output data set (which the OUTPUT statement produces) is not the same as the output data set that the OUT= option in the TABLES statement produces. The OUTPUT statement creates a data set that contains statistics (such as the Pearson chi-square and its *p*-value), while the OUT= option in the TABLES statement creates a data set that contains frequency table counts and percentages. For more information, see the section "Output Data Sets" on page 551.

You can also use the Output Delivery System (ODS) to store statistics that PROC FREQTAB computes. ODS can create a SAS data set from any table that PROC FREQTAB produces. For more information, see the section "ODS Table Names" on page 565 and Chapter 23, "Using the Output Delivery System" (*SAS/STAT User's Guide*).

You can specify the following *options* in the OUTPUT statement:

OUT=*data-table*

specifies the name of the output data table. When you use an OUTPUT statement but do not use the OUT= option, PROC FREQTAB creates a data set and names it by using the DATA*n* convention.

output-options

specify the statistics to include in the output data set. Table 11.6 lists the *output-options* that are available in the OUTPUT statement together with the TABLES statement options that are required to produce the statistics. Descriptions of the *output-options* follow the table in alphabetical order.

You can specify *output-options* to request individual statistics, or you can request groups of statistics by using *output-options* that are identical to the group options in the TABLES statement (for example, the CHISQ, MEASURES, CMH, AGREE, and ALL options).

When you specify an *output-option*, the output data set includes statistics from the corresponding analysis. In addition to the estimate or test statistic, the output data set includes associated values such as standard errors, confidence limits, p-values, and degrees of freedom. For more information, see the section "Contents of the OUTPUT Statement Output Data Set" on page 553.

To store a statistic in the output data set, you must also request computation of that statistic with the appropriate TABLES, EXACT, or TEST statement option. For example, the PCHI *output-option* includes the Pearson chi-square in the output data set. You must also request computation of the Pearson chi-square by specifying the CHISQ option in the TABLES statement. Or, if you use only one TABLES statement, you can request computation of the Pearson chi-square by specifying the PCHI or CHISQ option in the EXACT statement. Table 11.6 lists the TABLES statement options that are required to produce the OUTPUT data set statistics.

Table 11.6 OUTPUT Statement Output Options

Output Option	Output Data Set Statistics	Required TABLES Statement Option
AGREE	McNemar's test (2×2 tables), Bowker's test, simple and weighted kappas; for multiple strata, overall simple and weighted kappas, tests for equal kappas, and Cochran's Q ($2 \times 2 \times \cdots \times 2$ tables)	AGREE
AJCHI	Continuity-adjusted chi-square (2×2 tables)	CHISQ
ALL	CHISQ, MEASURES, and CMH statistics; N (number of observations)	ALL
BDCHI	Breslow-Day test ($h \times 2 \times 2$ tables)	CMH, CMH1, or CMH2
BINOMIAL | BIN	Binomial statistics (one-way tables)	BINOMIAL
CHISQ	For one-way tables, goodness-of-fit test; for two-way tables, Pearson, likelihood ratio, continuity-adjusted, and Mantel-Haenszel chi-squares, Fisher's exact test (2×2 tables), phi and contingency coefficients, Cramér's V	CHISQ
CMH	Cochran-Mantel-Haenszel (CMH) correlation, row mean scores (ANOVA), and general association statistics; for 2×2 tables, logit and Mantel-Haenszel common odds ratios and relative risks, Breslow-Day test	CMH
CMH1	CMH statistics, except row mean scores (ANOVA) and general association statistics	CMH or CMH1
CMH2	CMH statistics, except general association statistic	CMH or CMH2
CMHCOR	CMH correlation statistic	CMH, CMH1, or CMH2
CMHGA	CMH general association statistic	CMH
CMHRMS	CMH row mean scores (ANOVA) statistic	CMH or CMH2

Table 11.6 *continued*

Output Option	Output Data Set Statistics	Required TABLES Statement Option
COCHQ	Cochran's Q ($2 \times 2 \times \cdots \times 2$ tables)	AGREE or CQ
CONTGY	Contingency coefficient	CHISQ
CRAMV	Cramér's V	CHISQ
EQKAP	Test for equal simple kappas	AGREE
EQOR | ZELEN	Zelen's test for equal odds ratios ($h \times 2 \times 2$ tables)	CMH and EXACT EQOR
EQWKP	Test for equal weighted kappas	AGREE
FISHER	Fisher's exact test	CHISQ or FISHER [1]
GAILSIMON | GS	Gail-Simon test	GAILSIMON
GAMMA	Gamma	MEASURES
JT	Jonckheere-Terpstra test	JT
KAPPA	Simple kappa coefficient	AGREE
KENTB | TAUB	Kendall's tau-b	MEASURES
LAMCR	Lambda asymmetric $(C\|R)$	MEASURES
LAMDAS	Lambda symmetric	MEASURES
LAMRC	Lambda asymmetric $(R\|C)$	MEASURES
LGOR	Logit common odds ratio	CMH, CMH1, or CMH2
LGRRC1	Logit common relative risk, column 1	CMH, CMH1, or CMH2
LGRRC2	Logit common relative risk, column 2	CMH, CMH1, or CMH2
LRCHI	Likelihood ratio chi-square	CHISQ
MCNEM	McNemar's test (2×2 tables)	AGREE
MEASURES	Gamma, Kendall's tau-b, Stuart's tau-c, Somers' $D(C\|R)$ and $D(R\|C)$, Pearson and Spearman correlations, lambda asymmetric $(C\|R)$ and $(R\|C)$, lambda symmetric, uncertainty coefficients $(C\|R)$ and $(R\|C)$, symmetric uncertainty coefficient; odds ratio and relative risks (2×2 tables)	MEASURES
MHCHI	Mantel-Haenszel chi-square	CHISQ
MHOR | COMOR	Mantel-Haenszel common odds ratio	CMH, CMH1, or CMH2
MHRRC1	Mantel-Haenszel common relative risk, column 1	CMH, CMH1, or CMH2
MHRRC2	Mantel-Haenszel common relative risk, column 2	CMH, CMH1, or CMH2
N	Number of nonmissing observations	
NMISS	Number of missing observations	
OR | ODDSRATIO	Odds ratio (2×2 tables)	MEASURES, OR, or RELRISK
PCHI	Chi-square goodness-of-fit test (one-way tables), Pearson chi-square (two-way tables)	CHISQ
PCORR	Pearson correlation coefficient	MEASURES
PHI	Phi coefficient	CHISQ
PLCORR	Polychoric correlation coefficient	PLCORR
RDIF1	Column 1 risk difference (row 1 – row 2)	RISKDIFF

[1] CHISQ computes Fisher's exact test for 2×2 tables. Use the FISHER option to compute Fisher's exact test for general $r \times c$ tables.

Table 11.6 *continued*

Output Option	Output Data Set Statistics	Required TABLES Statement Option
RDIF2	Column 2 risk difference (row 1 – row 2)	RISKDIFF
RELRISK	Odds ratio and relative risks (2×2 tables)	MEASURES or RELRISK
RISKDIFF	Risks and risk differences (2×2 tables)	RISKDIFF
RISKDIFF1	Risks and risk difference, column 1	RISKDIFF
RISKDIFF2	Risks and risk difference, column 2	RISKDIFF
RRC1 | RELRISK1	Relative risk, column 1	MEASURES or RELRISK
RRC2 | RELRISK2	Relative risk, column 2	MEASURES or RELRISK
RSK1 | RISK1	Column 1 overall risk	RISKDIFF
RSK11 | RISK11	Column 1 risk for row 1	RISKDIFF
RSK12 | RISK12	Column 2 risk for row 1	RISKDIFF
RSK2 | RISK2	Column 2 overall risk	RISKDIFF
RSK21 | RISK21	Column 1 risk for row 2	RISKDIFF
RSK22 | RISK22	Column 2 risk for row 2	RISKDIFF
SCORR	Spearman correlation coefficient	MEASURES
SMDCR	Somers' $D(C\|R)$	MEASURES
SMDRC	Somers' $D(R\|C)$	MEASURES
STUTC | TAUC	Stuart's tau-c	MEASURES
TREND	Cochran-Armitage test for trend	TREND
TSYMM | BOWKER	Bowker's symmetry test	AGREE
U	Symmetric uncertainty coefficient	MEASURES
UCR	Uncertainty coefficient $(C\|R)$	MEASURES
URC	Uncertainty coefficient $(R\|C)$	MEASURES
WTKAPPA | WTKAP	Weighted kappa coefficient	AGREE

You can specify the following *output-options*:

AGREE

includes the following tests and measures of agreement in the output data set: McNemar's test (for 2×2 tables), Bowker's symmetry test, the simple kappa coefficient, and the weighted kappa coefficient. For multiway tables, the AGREE option also includes the following statistics in the output data set: overall simple and weighted kappa coefficients, tests for equal simple and weighted kappa coefficients, and Cochran's Q test (for $2 \times 2 \times \cdots \times 2$ tables).

AGREE statistics are computed only for square tables, where the number of rows equals the number of columns. PROC FREQTAB provides Bowker's symmetry test and weighted kappa coefficients for tables larger than 2×2. (For 2×2 tables, Bowker's test is identical to McNemar's test, and the weighted kappa coefficient equals the simple kappa coefficient.) Cochran's Q is available for $2 \times 2 \times \cdots \times 2$ tables.

The AGREE option in the TABLES statement requests computation of tests and measures of agreement. For more information, see the section "Tests and Measures of Agreement" on page 529.

AJCHI

includes the continuity-adjusted chi-square in the output data set. The continuity-adjusted chi-square is available for 2×2 tables and is provided by the CHISQ option in the TABLES statement. For more information, see the section "Continuity-Adjusted Chi-Square Test" on page 479.

ALL

includes all statistics that are requested by the CHISQ, MEASURES, and CMH *output-options* in the output data set. ALL also includes the number of nonmissing observations (total frequency), which can be requested separately by the N *output-option*.

BDCHI

includes the Breslow-Day test in the output data set. The Breslow-Day test for homogeneity of odds ratios is computed for multiway 2×2 tables and is provided by the CMH, CMH1, and CMH2 options in the TABLES statement. For more information, see the section "Breslow-Day Test for Homogeneity of the Odds Ratios" on page 542.

BINOMIAL

BIN

includes the following binomial statistics for one-way tables: number of observations (total frequency), number of successes (frequency of designated level), binomial proportion estimate, confidence limits, and tests. The BINOMIAL option in the TABLES statement requests computation of binomial statistics, which are available for one-way tables. For more information, see the section "Binomial Proportion" on page 491.

CHISQ

includes the following chi-square tests and measures in the output data set for two-way tables: Pearson chi-square, likelihood ratio chi-square, Mantel-Haenszel chi-square, phi coefficient, contingency coefficient, and Cramér's V. For 2×2 tables, CHISQ also includes Fisher's exact test and the continuity-adjusted chi-square in the output data set. For more information, see the section "Chi-Square Tests and Statistics" on page 476. For one-way tables, CHISQ includes the chi-square goodness-of-fit test in the output data set. For more information, see the section "Chi-Square Test for One-Way Tables" on page 477. The CHISQ option in the TABLES statement requests computation of these statistics.

If you specify the CHISQ(WARN=OUTPUT) option in the TABLES statement, the CHISQ option also includes the variable WARN_PCHI in the output data set. This variable indicates the validity warning for the asymptotic Pearson chi-square test.

CMH

includes the following Cochran-Mantel-Haenszel statistics in the output data set: correlation, row mean scores (ANOVA), and general association. For 2×2 tables, the CMH option also includes the Mantel-Haenszel and logit estimates of the common odds ratio and relative risks. For multiway (stratified) 2×2 tables, the CMH option includes the Breslow-Day test for homogeneity of odds ratios. The CMH option in the TABLES statement requests computation of these statistics. For more information, see the section "Cochran-Mantel-Haenszel Statistics" on page 536.

If you specify the CMH(MANTELFLEISS) option in the TABLES statement, the CMH option includes the Mantel-Fleiss analysis in the output data set. The variables MF_CMH and WARN_CMH contain the Mantel-Fleiss criterion and the warning indicator, respectively.

CMH1
includes the CMH statistics in the output data set, with the exception of the row mean scores (ANOVA) statistic and the general association statistic. The CMH1 option in the TABLES statement requests computation of these statistics. For more information, see the section "Cochran-Mantel-Haenszel Statistics" on page 536.

CMH2
includes the CMH statistics in the output data set, with the exception of the general association statistic. The CMH2 option in the TABLES statement requests computation of these statistics. For more information, see the section "Cochran-Mantel-Haenszel Statistics" on page 536.

CMHCOR
includes the Cochran-Mantel-Haenszel correlation statistic in the output data set. The CMH option in the TABLES statement requests computation of this statistic. For more information, see the section "Correlation Statistic" on page 538.

CMHGA
includes the Cochran-Mantel-Haenszel general association statistic in the output data set. The CMH option in the TABLES statement requests computation of this statistic. For more information, see the section "General Association Statistic" on page 539.

CMHRMS
includes the Cochran-Mantel-Haenszel row mean scores (ANOVA) statistic in the output data set. The CMH option in the TABLES statement requests computation of this statistic. For more information, see the section "ANOVA (Row Mean Scores) Statistic" on page 538.

COCHQ
includes Cochran's Q test in the output data set. Both the AGREE option and the CQ option in the TABLES statement request computation of this test, which is available for multiway tables in which each variable has two levels ($2 \times 2 \times \cdots \times 2$ tables). For more information, see the section "Cochran's Q Test" on page 535.

CONTGY
includes the contingency coefficient in the output data set. The CHISQ option in the TABLES statement requests computation of the contingency coefficient. For more information, see the section "Contingency Coefficient" on page 481.

CRAMV
includes Cramér's V in the output data set. The CHISQ option in the TABLES statement requests computation of Cramér's V. For more information, see the section "Cramér's V" on page 481.

EQKAP
includes the test for equal simple kappa coefficients in the output data set. The AGREE option in the TABLES statement requests computation of this test, which is available for multiway, square ($h \times r \times r$) tables. For more information, see the section "Tests for Equal Kappa Coefficients" on page 535.

EQOR

ZELEN

includes Zelen's exact test for equal odds ratios in the output data set. The EQOR option in the EXACT statement requests computation of this test, which is available for multiway 2×2 tables. For more information, see the section "Zelen's Exact Test for Equal Odds Ratios" on page 543.

EQWKP

includes the test for equal weighted kappa coefficients in the output data set. The AGREE option in the TABLES statement requests computation of this test. The test for equal weighted kappas is available for multiway, square ($h \times r \times r$) tables where $r > 2$. For more information, see the section "Tests for Equal Kappa Coefficients" on page 535.

FISHER

includes Fisher's exact test in the output data set. For 2×2 tables, the CHISQ option in the TABLES statement provides Fisher's exact test. For tables larger than 2×2, the FISHER option in the EXACT statement provides Fisher's exact test. For more information, see the section "Fisher's Exact Test" on page 480.

GAILSIMON

GS

includes the Gail-Simon test for qualitative interaction in the output data set. The GAILSIMON option in the TABLES statement requests computation of this test. For more information, see the section "Gail-Simon Test for Qualitative Interactions" on page 546.

GAMMA

includes the gamma statistic in the output data set. The MEASURES option in the TABLES statement requests computation of the gamma statistic. For more information, see the section "Gamma" on page 483.

JT

includes the Jonckheere-Terpstra test in the output data set. The JT option in the TABLES statement requests the Jonckheere-Terpstra test. For more information, see the section "Jonckheere-Terpstra Test" on page 527.

KAPPA

includes the simple kappa coefficient in the output data set. The AGREE option in the TABLES statement requests computation of kappa, which is available for square tables (where the number of rows equals the number of columns). For multiway square tables, the KAPPA option also includes the overall kappa coefficient in the output data set. For more information, see the sections "Simple Kappa Coefficient" on page 530 and "Overall Kappa Coefficient" on page 535.

KENTB

TAUB

includes Kendall's tau-*b* in the output data set. The MEASURES option in the TABLES statement requests computation of Kendall's tau-*b*. For more information, see the section "Kendall's Tau-b" on page 484.

LAMCR
includes the asymmetric lambda $\lambda(C|R)$ in the output data set. The MEASURES option in the TABLES statement requests computation of lambda. For more information, see the section "Lambda (Asymmetric)" on page 489.

LAMDAS
includes the symmetric lambda in the output data set. The MEASURES option in the TABLES statement requests computation of lambda. For more information, see the section "Lambda (Symmetric)" on page 490.

LAMRC
includes the asymmetric lambda $\lambda(R|C)$ in the output data set. The MEASURES option in the TABLES statement requests computation of lambda. For more information, see the section "Lambda (Asymmetric)" on page 489.

LGOR
includes the logit estimate of the common odds ratio in the output data set. The CMH option in the TABLES statement requests computation of this statistic, which is available for 2×2 tables. For more information, see the section "Adjusted Odds Ratio and Relative Risk Estimates" on page 540.

LGRRC1
includes the logit estimate of the common relative risk (column 1) in the output data set. The CMH option in the TABLES statement requests computation of this statistic, which is available for 2×2 tables. For more information, see the section "Adjusted Odds Ratio and Relative Risk Estimates" on page 540.

LGRRC2
includes the logit estimate of the common relative risk (column 2) in the output data set. The CMH option in the TABLES statement requests computation of this statistic, which is available for 2×2 tables. For more information, see the section "Adjusted Odds Ratio and Relative Risk Estimates" on page 540.

LRCHI
includes the likelihood ratio chi-square in the output data set. The CHISQ option in the TABLES statement requests computation of the likelihood ratio chi-square. For more information, see the section "Likelihood Ratio Chi-Square Test" on page 479.

MCNEM
includes McNemar's test (for 2×2 tables) in the output data set. The AGREE option in the TABLES statement requests computation of McNemar's test. For more information, see the section "McNemar's Test" on page 529.

MEASURES
includes the following measures of association in the output data set: gamma, Kendall's tau-*b*, Stuart's tau-*c*, Somers' $D(C|R)$, Somers' $D(R|C)$, Pearson and Spearman correlation coefficients, lambda (symmetric and asymmetric), and uncertainty coefficients (symmetric and asymmetric). For 2×2 tables, the MEASURES option also includes the odds ratio, column 1 relative risk, and column 2 relative risk. The MEASURES option in the TABLES statement requests computation of these statistics. For more information, see the section "Measures of Association" on page 482.

MHCHI

includes the Mantel-Haenszel chi-square in the output data set. The CHISQ option in the TABLES statement requests computation of the Mantel-Haenszel chi-square. For more information, see the section "Mantel-Haenszel Chi-Square Test" on page 480.

MHOR

COMOR

includes the Mantel-Haenszel estimate of the common odds ratio in the output data set. The CMH option in the TABLES statement requests computation of this statistic, which is available for 2×2 tables. For more information, see the section "Adjusted Odds Ratio and Relative Risk Estimates" on page 540.

MHRRC1

includes the Mantel-Haenszel estimate of the common relative risk (column 1) in the output data set. The CMH option in the TABLES statement requests computation of this statistic, which is available for 2×2 tables. For more information, see the section "Adjusted Odds Ratio and Relative Risk Estimates" on page 540.

MHRRC2

includes the Mantel-Haenszel estimate of the common relative risk (column 2) in the output data set. The CMH option in the TABLES statement requests computation of this statistic, which is available for 2×2 tables. For more information, see the section "Adjusted Odds Ratio and Relative Risk Estimates" on page 540.

N

includes the number of nonmissing observations (total frequency) for the table request.

NMISS

includes the number (frequency) of missing observations for the table request. For more information, see the section "Missing Values" on page 473.

OR

ODDSRATIO

RROR

includes the odds ratio (for 2×2 tables) in the output data set. The MEASURES, OR, and RELRISK options in the TABLES statement request this statistic. For more information, see the section "Odds Ratio" on page 516.

PCHI

includes the Pearson chi-square in the output data set for two-way tables. For more information, see the section "Pearson Chi-Square Test for Two-Way Tables" on page 477. For one-way tables, the PCHI option includes the chi-square goodness-of-fit test in the output data set. For more information, see the section "Chi-Square Test for One-Way Tables" on page 477. The CHISQ option in the TABLES statement requests computation of these statistics.

If you specify the CHISQ(WARN=OUTPUT) option in the TABLES statement, the PCHI option also includes the variable WARN_PCHI in the output data set. This variable indicates the validity warning for the asymptotic Pearson chi-square test.

PCORR
includes the Pearson correlation coefficient in the output data set. The MEASURES option in the TABLES statement requests computation of the Pearson correlation. For more information, see the section "Pearson Correlation Coefficient" on page 485.

PHI
includes the phi coefficient in the output data set. The CHISQ option in the TABLES statement requests computation of the phi coefficient. For more information, see the section "Phi Coefficient" on page 481.

PLCORR
includes the polychoric correlation coefficient in the output data set. For 2×2 tables, this statistic is known as the tetrachoric correlation coefficient. The PLCORR option in the TABLES statement requests computation of the polychoric correlation. For more information, see the section "Polychoric Correlation" on page 488.

RDIF1
includes the column 1 risk difference (row 1 – row 2) in the output data set. The RISKDIFF option in the TABLES statement requests computation of risks and risk differences, which are available for 2×2 tables. For more information, see the section "Risks and Risk Differences" on page 501.

RDIF2
includes the column 2 risk difference (row 1 – row 2) in the output data set. The RISKDIFF option in the TABLES statement requests computation of risks and risk differences, which are available for 2×2 tables. For more information, see the section "Risks and Risk Differences" on page 501.

RELRISK
includes the column 1 and column 2 relative risks (for 2×2 tables) in the output data set. The MEASURES and RELRISK options in the TABLES statement request these statistics. For more information, see the section "Relative Risks" on page 519.

RISKDIFF
includes risks (binomial proportions) and risk differences for 2×2 tables in the output data set. These statistics include the row 1 risk, row 2 risk, total (overall) risk, and risk difference (row 1 – row 2) for column 1 and column 2. The RISKDIFF option in the TABLES statement requests computation of these statistics. For more information, see the section "Risks and Risk Differences" on page 501.

RISKDIFF1
includes column 1 risks (binomial proportions) and risk differences for 2×2 tables in the output data set. These statistics include the row 1 risk, row 2 risk, total (overall) risk, and risk difference (row 1 – row 2). The RISKDIFF option in the TABLES statement requests computation of these statistics. For more information, see the section "Risks and Risk Differences" on page 501.

RISKDIFF2
includes column 2 risks (binomial proportions) and risk differences for 2×2 tables in the output data set. These statistics include the row 1 risk, row 2 risk, total (overall) risk, and risk difference (row 1 – row 2). The RISKDIFF option in the TABLES statement requests computation of these statistics. For more information, see the section "Risks and Risk Differences" on page 501.

RRC1

RELRISK1

includes the column 1 relative risk in the output data set. The MEASURES and RELRISK options in the TABLES statement request relative risks, which are available for 2×2 tables. For more information, see the section "Odds Ratio and Relative Risks" on page 516.

RRC2

RELRISK2

includes the column 2 relative risk in the output data set. The MEASURES and RELRISK options in the TABLES statement request relative risks, which are available for 2×2 tables. For more information, see the section "Odds Ratio and Relative Risks" on page 516.

RSK1

RISK1

includes the overall column 1 risk in the output data set. The RISKDIFF option in the TABLES statement requests computation of risks and risk differences, which are available for 2×2 tables. For more information, see the section "Risks and Risk Differences" on page 501.

RSK11

RISK11

includes the column 1 risk for row 1 in the output data set. The RISKDIFF option in the TABLES statement requests computation of risks and risk differences, which are available for 2×2 tables. For more information, see the section "Risks and Risk Differences" on page 501.

RSK12

RISK12

includes the column 2 risk for row 1 in the output data set. The RISKDIFF option in the TABLES statement requests computation of risks and risk differences, which are available for 2×2 tables. For more information, see the section "Risks and Risk Differences" on page 501.

RSK2

RISK2

includes the overall column 2 risk in the output data set. The RISKDIFF option in the TABLES statement requests computation of risks and risk differences. For more information, see the section "Risks and Risk Differences" on page 501.

RSK21

RISK21

includes the column 1 risk for row 2 in the output data set. The RISKDIFF option in the TABLES statement requests computation of risks and risk differences, which are available for 2×2 tables. For more information, see the section "Risks and Risk Differences" on page 501.

RSK22

RISK22

includes the column 2 risk for row 2 in the output data set. The RISKDIFF option in the TABLES statement requests computation of risks and risk differences, which are available for 2×2 tables. For more information, see the section "Risks and Risk Differences" on page 501.

SCORR
includes the Spearman correlation coefficient in the output data set. The MEASURES option in the TABLES statement requests computation of the Spearman correlation. For more information, see the section "Spearman Rank Correlation Coefficient" on page 486.

SMDCR
includes Somers' $D(C|R)$ in the output data set. The MEASURES option in the TABLES statement requests computation of Somers' D. For more information, see the section "Somers' D" on page 485.

SMDRC
includes Somers' $D(R|C)$ in the output data set. The MEASURES option in the TABLES statement requests computation of Somers' D. For more information, see the section "Somers' D" on page 485.

STUTC
TAUC
includes Stuart's tau-c in the output data set. The MEASURES option in the TABLES statement requests computation of tau-c. For more information, see the section "Stuart's Tau-c" on page 484.

TREND
includes the Cochran-Armitage test for trend in the output data set. The TREND option in the TABLES statement requests computation of the trend test. This test is available for tables of dimension $2 \times C$ or $R \times 2$. For more information, see the section "Cochran-Armitage Test for Trend" on page 526.

TSYMM
BOWKER
includes Bowker's symmetry test in the output data set. The AGREE option in the TABLES statement requests computation of Bowker's test. For more information, see the section "Bowker's Symmetry Test" on page 529.

U
includes the uncertainty coefficient (symmetric) in the output data set. The MEASURES option in the TABLES statement requests computation of the uncertainty coefficient. For more information, see the section "Uncertainty Coefficient (Symmetric)" on page 491.

UCR
includes the asymmetric uncertainty coefficient $U(C|R)$ in the output data set. The MEASURES option in the TABLES statement requests computation of the uncertainty coefficient. For more information, see the section "Uncertainty Coefficients (Asymmetric)" on page 490.

URC
includes the asymmetric uncertainty coefficient $U(R|C)$ in the output data set. The MEASURES option in the TABLES statement requests computation of the uncertainty coefficient. For more information, see the section "Uncertainty Coefficients (Asymmetric)" on page 490.

WTKAPPA
WTKAP
includes the weighted kappa coefficient in the output data set. The AGREE option in the TABLES statement requests computation of weighted kappa, which is available for square tables larger than 2×2. For multiway tables, the WTKAPPA option also includes the overall weighted kappa coefficient in the output data set. For more information, see the sections "Weighted Kappa Coefficient" on page 532 and "Overall Kappa Coefficient" on page 535.

TABLES Statement

TABLES *requests* < / *options* > **;**

The TABLES statement requests one-way to *n*-way frequency and crosstabulation tables and statistics for those tables.

If you omit the TABLES statement, PROC FREQTAB generates one-way frequency tables for all data set variables that are not listed in the other statements.

The following argument is required in the TABLES statement.

requests

specify the frequency and crosstabulation tables to produce. A request is composed of one variable name or several variable names separated by asterisks. To request a one-way frequency table, use a single variable. To request a two-way crosstabulation table, use an asterisk between two variables. To request a multiway table (an *n*-way table, where $n > 2$), separate the variables with asterisks. The unique values of these variables form the rows, columns, and strata of the table. You can include up to 50 variables in a single multiway table request.

For two-way to multiway tables, the values of the last variable form the crosstabulation table columns, and the values of the next-to-last variable form the rows. Each level (or combination of levels) of the other variables forms one stratum. PROC FREQTAB produces a separate crosstabulation table for each stratum. For example, a specification of A*B*C*D in a TABLES statement produces *k* tables, where *k* is the number of different combinations of values for A and B. Each table lists the values for C down the side and the values for D across the top.

You can use multiple TABLES statements in the PROC FREQTAB step. PROC FREQTAB builds all the table requests in one pass of the data, so that there is essentially no loss of efficiency. You can also specify any number of table requests in a single TABLES statement. To specify multiple table requests quickly, use a grouping syntax by placing parentheses around several variables and joining other variables or variable combinations. For example, the statements shown in Table 11.7 illustrate grouping syntax.

Table 11.7 Grouping Syntax

TABLES Request	Equivalent to
A*(B C)	A*B A*C
(A B)*(C D)	A*C B*C A*D B*D
(A B C)*D	A*D B*D C*D
A – C	A B C
(A – C)*D	A*D B*D C*D

The TABLES statement variables are one or more variables from the DATA= input data table. These variables can be either character or numeric, but the procedure treats them as categorical variables. PROC FREQTAB uses the formatted values of the TABLES variable to determine the categorical variable levels. For more information, see the discussion of the FORMAT procedure in the *Base SAS Procedures Guide* and the discussion of SAS formats in *SAS Formats and Informats: Reference*.

By default, the frequency or crosstabulation table lists the values of both character and numeric variables in ascending order based on internal (unformatted) variable values. You can change the order of the values in the table by specifying the ORDER= option in the PROC FREQTAB statement. To list the values in ascending order by formatted value, use ORDER=FORMATTED.

Without Options

If you request a one-way frequency table without specifying any options, PROC FREQTAB provides the following for each table level or cell: frequency, cumulative frequency, percentage of the total frequency, and cumulative percentage. If you request a two-way or multiway (n-way) crosstabulation table without specifying any options, PROC FREQTAB provides the following for each table cell; frequency, percentage of the total frequency, percentage of the row frequency, and percentages of the column frequency. PROC FREQTAB also displays the frequency of missing values after the table. For more information, see the section "Displayed Output" on page 554.

Options

Table 11.8 lists the *options* available in the TABLES statement. Descriptions of the *options* follow in alphabetical order.

Table 11.8 TABLES Statement Options

Option	Description
Control Statistical Analysis	
AGREE	Requests tests and measures of classification agreement
ALL	Requests tests and measures of association produced by the CHISQ, MEASURES, and CMH options
ALPHA=	Sets confidence level for confidence limits
BINOMIAL | BIN	Requests binomial proportions, confidence limits, and tests for one-way tables
CHISQ	Requests chi-square tests and measures based on chi-square
CL	Requests confidence limits for MEASURES statistics
CMH	Requests all Cochran-Mantel-Haenszel statistics
CMH1	Requests CMH correlation statistic, adjusted odds ratios, and adjusted relative risks
CMH2	Requests CMH correlation and row mean scores (ANOVA) statistics, adjusted odds ratios, and adjusted relative risks
COMMONRISKDIFF	Requests common risk difference for $h \times 2 \times 2$ tables
CQ	Requests Cochran's Q test
FISHER	Requests Fisher's exact test for tables larger than 2×2
GAILSIMON	Requests Gail-Simon test for qualitative interactions
JT	Requests Jonckheere-Terpstra test
MEASURES	Requests measures of association
OR	Requests the odds ratio for 2×2 tables
PLCORR	Requests polychoric correlation
RELRISK	Requests relative risks for 2×2 tables
RISKDIFF	Requests risks and risk differences for 2×2 tables
SCORES=	Specifies type of row and column scores

Table 11.8 *continued*

Option	Description
SENSPEC	Requests sensitivity and specificity for 2 × 2 tables
TREND	Requests Cochran-Armitage test for trend
Control Additional Table Information	
CELLCHI2	Displays cell contributions to the Pearson chi-square statistic
CUMCOL	Displays cumulative column percentages
DEVIATION	Displays deviations of cell frequencies from expected values
EXPECTED	Displays expected cell frequencies
PRINTKWTS	Displays kappa coefficient weights
SCOROUT	Displays row and column scores
TOTPCT	Displays percentages of total frequency for n-way tables ($n > 2$)
Control Displayed Output	
CROSSLIST	Displays crosstabulation tables in ODS column format
FORMAT=	Formats frequencies in crosstabulation tables
FORMATP=	Formats percentages in crosstabulation tables
LIST	Displays two-way and n-way tables in list format
MAXLEVELS=	Specifies maximum number of levels to display in one-way tables
NOCOL	Suppresses display of column percentages
NOCUM	Suppresses display of cumulative frequencies and percentages
NOFREQ	Suppresses display of frequencies
NOPERCENT	Suppresses display of percentages
NOPRINT	Suppresses display of crosstabulation tables but displays statistics
NOROW	Suppresses display of row percentages
NOSPARSE	Suppresses zero-frequency levels in the CROSSLIST table, LIST table, and OUT= data table
NOWARN	Suppresses log warning message for the chi-square test
ONEWAY	Specifies content of one-way frequency tables
Produce Statistical Graphics	
PLOTS=	Requests plots from ODS Graphics
Create an Output Data Set	
OUT=	Names an output data table to contain frequency counts
OUTCUM	Includes cumulative frequencies and percentages in the output data set
OUTEXPECT	Includes expected frequencies in the output data set
OUTPCT	Includes row, column, and two-way table percentages in the output data set

You can specify the following *options*:

AGREE < (*agree-options*) >

requests tests and measures of classification agreement for square tables. This option provides the simple and weighted kappa coefficients along with their standard errors and confidence limits. For multiway tables, this option also produces the overall simple and weighted kappa coefficients (along with their standard errors and confidence limits) and tests for equal kappas among strata. For 2×2 tables, this option provides McNemar's test; for square tables that have more than two response categories (levels), this option provides Bowker's symmetry test. For multiway tables in which each variable has two levels, this option also produces Cochran's Q test. (You can request Cochran's Q test separately by specifying the CQ option.) For more information, see the section "Tests and Measures of Agreement" on page 529.

Measures of agreement can be computed only for square tables, where the number of rows equals the number of columns. If your table is not square because some observations have weights of 0, you can specify the ZEROS option in the WEIGHT statement to include these observations in the analysis. For more information, see the section "Tables with Zero-Weight Rows or Columns" on page 536.

For 2×2 tables, the weighted kappa coefficient is equivalent to the simple kappa coefficient, and PROC FREQTAB displays only analyses for the simple kappa coefficient.

You can specify the confidence level in the ALPHA= option. By default, ALPHA=0.05, which produces 95% confidence limits.

You can specify the EXACT statement to request McNemar's exact test (for 2×2 tables), an exact symmetry test, and exact tests for the simple and weighted kappa coefficients. For more information, see the section "Exact Statistics" on page 547.

You can specify the following *agree-options*:

AC1

requests the AC1 agreement coefficient. For more information, see the section "AC1 Agreement Coefficient" on page 534.

DFSYM=*df* | ADJUST

controls the degrees of freedom for Bowker's symmetry test. You can specify the value of the degrees of freedom (*df*), or you can specify DFSYM=ADJUST to adjust the degrees of freedom for empty table cells. The value of *df* must be a positive number. By default, *df* is $R(R-1)/2$, where R is the dimension of the two-way table.

When you specify DFSYM=ADJUST, the degrees of freedom are reduced by the number of off-diagonal table-cell pairs that have a total frequency of 0. By default, the degrees of freedom count all off-diagonal table-cell pairs. For more information, see the section "Bowker's Symmetry Test" on page 529.

KAPPADETAILS

DETAILS

displays the "Kappa Details" table, which includes the following statistics for the simple kappa coefficient: observed agreement, chance-expected agreement, maximum kappa, and the B_n measure. If the two-way table is 2×2, the "Kappa Details" table also includes the prevalence index and the bias index. For more information, see the section "Simple Kappa Coefficient" on page 530.

If the two-way table is larger than 2×2, this option also displays the "Weighted Kappa Details" table, which includes the observed agreement and chance-expected agreement components of the

weighted kappa coefficient. For more information, see the section "Weighted Kappa Coefficient" on page 532.

MNULLRATIO=*value*
specifies the null *value* of the ratio of discordant proportions for McNemar's test. By default, MNULLRATIO=1. For more information, see the section "McNemar's Test" on page 529.

NULLKAPPA=*value*
requests the simple kappa coefficient test and specifies the null *value* for the test. The null value must be between –1 and 1. By default, NULLKAPPA=0. For more information, see the section "Simple Kappa Coefficient" on page 530.

This option is not available when you specify the KAPPA option in the EXACT statement, which requests an exact test for the kappa coefficient.

NULLWTKAPPA=*value*
requests the weighted kappa coefficient test and specifies the null *value* for the test. The null value must be between –1 and 1. By default, NULLWTKAPPA=0. For more information, see the section "Weighted Kappa Coefficient" on page 532.

This option is not available when you specify the WTKAPPA option in the EXACT statement, which requests an exact test for the weighted kappa coefficient.

PABAK
requests the prevalence-adjusted bias-adjusted kappa coefficient. For more information, see the section "Prevalence-Adjusted Bias-Adjusted Kappa" on page 534.

PRINTKWTS
displays the agreement weights that PROC FREQTAB uses to compute the weighted kappa coefficient. Agreement weights reflect the relative agreement between pairs of variable levels. By default, PROC FREQTAB uses Cicchetti-Allison agreement weights. If you specify the WT=FC option, the procedure uses Fleiss-Cohen agreement weights. For more information, see the section "Weighted Kappa Coefficient" on page 532.

TABLES=RESTORE
displays the following agreement tables (which are produced by the AGREE option) in factoid (label-value) format: "McNemar's Test," "Symmetry Test," "Monte Carlo Exact Symmetry Test," "Simple Kappa Coefficient," "Simple Kappa Test," "Weighted Kappa Coefficient," "Weighted Kappa Test," "Monte Carlo Estimates for the Exact Test," "Overall Kappa Coefficient," "Tests for Equal Kappa Coefficients," and "Cochran's Q."

By default, PROC FREQTAB displays all agreement tables in tabular format.

WT=FC
specifies Fleiss-Cohen agreement weights in the computation of the weighted kappa coefficient. Agreement weights reflect the relative agreement between pairs of variable levels. By default, PROC FREQTAB uses Cicchetti-Allison agreement weights to compute the weighted kappa coefficient. For more information, see the section "Weighted Kappa Coefficient" on page 532.

WTKAPPADETAILS
displays the "Weighted Kappa Details" table, which includes the observed agreement and chance-expected agreement components of the weighted kappa coefficient. This information is available for two-way tables that are larger than 2×2. For more information, see the section "Weighted Kappa Coefficient" on page 532.

ALL
requests all tests and measures that are produced by the CHISQ, MEASURES, and CMH options. You can control the number of CMH statistics to compute by specifying the CMH1 or CMH2 option.

ALPHA=α
specifies the level of confidence limits. The value of α must be between 0 and 1; a confidence level of α produces $100(1-\alpha)\%$ confidence limits. By default ALPHA=0.05, which produces 95% confidence limits.

This option applies to confidence limits that you request in the TABLES statement. The ALPHA= option in the EXACT statement applies to confidence limits for Monte Carlo estimates of exact p-values, which you request by specifying the MC option in the EXACT statement.

BINOMIAL < (*binomial-options*) >
BIN < (*binomial-options*) >
requests the binomial proportion for one-way tables. By default, PROC FREQTAB provides the asymptotic standard error, asymptotic Wald and exact (Clopper-Pearson) confidence limits, and the asymptotic equality test for the binomial proportion.

You can specify *binomial-options* in parentheses after the BINOMIAL option. The LEVEL= *binomial-option* identifies the variable level for which to compute the proportion. If you do not specify this option, PROC FREQTAB computes the proportion for the first level that appears in the one-way frequency table. The P= *binomial-option* specifies the null proportion for the binomial tests. If you do not specify this option, PROC FREQTAB uses 0.5 as the null proportion for the binomial tests.

You can also specify *binomial-options* to request additional tests and confidence limits for the binomial proportion. The EQUIV, NONINF, and SUP *binomial-options* request tests of equivalence, noninferiority, and superiority, respectively. The CL= *binomial-option* requests confidence limits for the binomial proportion.

You can specify the level for the binomial confidence limits in the ALPHA= option. By default, ALPHA=0.05, which produces 95% confidence limits. As part of the noninferiority, superiority, and equivalence analyses, PROC FREQTAB provides null-based equivalence limits that have a confidence coefficient of $100(1-2\alpha)\%$ (Schuirmann 1999). In these analyses, the default of ALPHA=0.05 produces 90% equivalence limits. For more information, see the sections "Noninferiority Test" on page 496 and "Equivalence Test" on page 498.

To request exact tests for the binomial proportion, you can specify the BINOMIAL option in the EXACT statement. PROC FREQTAB computes exact p-values for all binomial tests that you request, which can include noninferiority, superiority, and equivalence tests, in addition to the equality test that the BINOMIAL option produces by default.

For more information, see the section "Binomial Proportion" on page 491.

Table 11.9 summarizes the *binomial-options*.

Table 11.9 BINOMIAL Options

Option	Description
CORRECT	Requests continuity correction
LEVEL=	Specifies the variable level
OUTLEVEL	Includes the level in the output data sets
P=	Specifies the null proportion
Request Confidence Limits	
CL=AGRESTICOULL | AC	Requests Agresti-Coull confidence limits
CL=BLAKER	Requests Blaker confidence limits
CL=EXACT | CLOPPERPEARSON	Requests exact (Clopper-Pearson) confidence limits
CL=JEFFREYS	Requests Jeffreys confidence limits
CL=LIKELIHOODRATIO | LR	Requests likelihood ratio confidence limits
CL=LOGIT	Requests logit confidence limits
CL=MIDP	Requests exact mid-*p* confidence limits
CL=WALD	Requests Wald confidence limits
CL=WILSON | SCORE	Requests Wilson (score) confidence limits
Request Tests	
EQUIV | EQUIVALENCE	Requests an equivalence test
MARGIN=	Specifies the test margin
NONINF | NONINFERIORITY	Requests a noninferiority test
SUP | SUPERIORITY	Requests a superiority test
VAR=NULL | SAMPLE	Specifies the test variance

You can specify the following *binomial-options*:

CL=*type* **|** **(***types***)**
requests confidence limits for the binomial proportion. You can specify one or more *types* of confidence limits. When you specify only one *type*, you can omit the parentheses around the request. PROC FREQTAB displays the confidence limits in the "Binomial Confidence Limits" table.

The ALPHA= option determines the level of the confidence limits that the CL= *binomial-option* provides. By default, ALPHA=0.05, which produces 95% confidence limits for the binomial proportion.

You can specify the CL= *binomial-option* with or without requests for binomial tests. The confidence limits that CL= produces do not depend on the tests that you request and do not use the value of the test margin (which you can specify in the MARGIN= *binomial-option*).

If you do not specify the CL= *binomial-option*, the BINOMIAL option displays Wald and exact (Clopper-Pearson) confidence limits in the "Binomial Proportion" table.

You can specify the following *types*:

AGRESTICOULL

AC

requests Agresti-Coull confidence limits for the binomial proportion. For more information, see the section "Agresti-Coull Confidence Limits" on page 493.

BLAKER

requests Blaker confidence limits for the binomial proportion. For more information, see the section "Blaker Confidence Limits" on page 493.

EXACT

CLOPPERPEARSON

requests exact (Clopper-Pearson) confidence limits for the binomial proportion. For more information, see the section "Exact (Clopper-Pearson) Confidence Limits" on page 492.

If you do not specify the CL= *binomial-option*, PROC FREQTAB displays Wald and exact (Clopper-Pearson) confidence limits in the "Binomial Proportion" table. To request exact tests for the binomial proportion, you can specify the BINOMIAL option in the EXACT statement.

JEFFREYS < (*option*) >

requests Jeffreys confidence limits for the binomial proportion. For more information, see the section "Jeffreys Confidence Limits" on page 493.

You can specify the following *option*:

MODIFY

provides modified Jeffreys confidence limits (Brown, Cai, and DasGupta 2001) to improve the coverage of the confidence limits for proportions near 0 or 1. For more information, see the section "Jeffreys Confidence Limits" on page 493.

LIKELIHOODRATIO

LR

requests likelihood ratio confidence limits for the binomial proportion. For more information, see the section "Likelihood Ratio Confidence Limits" on page 494.

LOGIT

requests logit confidence limits for the binomial proportion. For more information, see the section "Logit Confidence Limits" on page 494.

MIDP

requests exact mid-*p* confidence limits for the binomial proportion. For more information, see the section "Mid-*p* Confidence Limits" on page 494.

WALD < (CORRECT) >

requests Wald confidence limits for the binomial proportion. For more information, see the section "Wald Confidence Limits" on page 492.

If you specify CL=WALD(CORRECT), the Wald confidence limits include a continuity correction. If you specify the CORRECT *binomial-option*, both the Wald confidence limits and the Wald tests include continuity corrections.

If you do not specify the CL= *binomial-option*, PROC FREQTAB displays Wald and exact (Clopper-Pearson) confidence limits in the "Binomial Proportion" table.

WILSON <(*option*)>

SCORE <(*option*)>

requests Wilson confidence limits for the binomial proportion. These are also known as *score* confidence limits. For more information, see the section "Wilson (Score) Confidence Limits" on page 495.

You can specify one of the following *options*:

ADAPT

provides adapted Wilson confidence limits (Agresti and Coull 1998) to improve the coverage of the confidence limits for proportions near 0 or 1. Adapted confidence limits are computed when the number of successes is 1 or $n - 1$. For more information, see the section "Wilson (Score) Confidence Limits" on page 495.

CORRECT

includes a continuity correction in the Wilson confidence limits. For more information, see the section "Wilson (Score) Confidence Limits" on page 495.

MODIFY

provides modified Wilson confidence limits (Brown, Cai, and DasGupta 2001) to improve the coverage of the confidence limits for proportions near 0 or 1. For more information, see the section "Wilson (Score) Confidence Limits" on page 495.

CORRECT

includes a continuity correction in the Wald confidence limits, Wald tests, and Wilson confidence limits.

You can request continuity corrections individually for Wald or Wilson confidence limits by specifying the CL=WALD(CORRECT) or CL=WILSON(CORRECT) *binomial-option*, respectively.

EQUIV

EQUIVALENCE

requests a test of equivalence for the binomial proportion. For more information, see the section "Equivalence Test" on page 498. You can specify the equivalence test margins, the null proportion, and the variance type in the MARGIN=, P=, and VAR= *binomial-options*, respectively. To request an exact equivalence test, you can specify the BINOMIAL option in the EXACT statement.

LEVEL=*level-number* | '*level-value*'

specifies the variable level for the binomial proportion. You can specify the *level-number*, which is the order in which the level appears in the one-way frequency table. Or you can specify the *level-value*, which is the formatted value of the variable level. The *level-number* must be a positive integer. You must enclose the *level-value* in single quotes.

By default, PROC FREQTAB computes the binomial proportion for the first variable level that appears in the one-way frequency table.

MARGIN=*value* | **(***lower***,** *upper***)**

specifies the margin for the noninferiority, superiority, and equivalence tests, which you can request by specifying the NONINF, SUP, and EQUIV *binomial-options*, respectively. By default, MARGIN=0.2.

For noninferiority and superiority tests, specify a single *value* in the MARGIN= option. The MARGIN= *value* must be a positive number. You can specify *value* as a number between 0 and 1. Or you can specify *value* in percentage form as a number between 1 and 100, and PROC FREQTAB converts that number to a proportion. PROC FREQTAB treats the value 1 as 1%.

For noninferiority and superiority tests, the test limits must be between 0 and 1. The limits are determined by the null proportion value (which you can specify in the P= *binomial-option*) and by the margin value. The noninferiority limit is the null proportion minus the margin. By default, the null proportion is 0.5 and the margin is 0.2, which produces a noninferiority limit of 0.3. The superiority limit is the null proportion plus the margin, which is 0.7 by default.

For an equivalence test, you can specify a single MARGIN= *value*, or you can specify both *lower* and *upper* values. If you specify a single MARGIN= *value*, it must be a positive number, as described previously. If you specify a single MARGIN= *value* for an equivalence test, PROC FREQTAB uses –*value* as the lower margin and *value* as the upper margin for the test. If you specify both *lower* and *upper* values for an equivalence test, you can specify them in proportion form as numbers between –1 and 1. Or you can specify them in percentage form as numbers between –100 and 100, and PROC FREQTAB converts the numbers to proportions. The value of *lower* must be less than the value of *upper*.

The equivalence limits must be between 0 and 1. The equivalence limits are determined by the null proportion value (which you can specify in the P= *binomial-option*) and by the margin values. The lower equivalence limit is the null proportion plus the lower margin. By default, the null proportion is 0.5 and the lower margin is –0.2, which produces a lower equivalence limit of 0.3. The upper equivalence limit is the null proportion plus the upper margin, which is 0.7 by default.

For more information, see the sections "Noninferiority Test" on page 496 and "Equivalence Test" on page 498.

NONINF

NONINFERIORITY

requests a test of noninferiority for the binomial proportion. For more information, see the section "Noninferiority Test" on page 496. You can specify the noninferiority test margin, the null proportion, and the variance type in the MARGIN=, P=, and VAR= *binomial-options*, respectively. To request an exact noninferiority test, you can specify the BINOMIAL option in the EXACT statement.

OUTLEVEL

includes the variables LevelNumber and LevelValue in all ODS output data sets that PROC FREQTAB produces when you specify the BINOMIAL option in the TABLES statement. The OUTLEVEL option also includes the variables LevelNumber and LevelValue in the statistics output data set that PROC FREQTAB produces when you specify the BINOMIAL option in the OUTPUT statement.

The LevelNumber and LevelValue variables identify the analysis variable level for which PROC FREQTAB computes the binomial proportion. The value of LevelNumber is the order of the level

in the one-way frequency table. The value of LevelValue is the formatted value of the level. You can specify the OUTLEVEL *binomial-option* with or without the LEVEL= *binomial-option*.

P=*value*
specifies the null hypothesis proportion for the binomial tests. The null proportion *value* must be a positive number. You can specify *value* as a number between 0 and 1. Or you can specify *value* in percentage form (as a number between 1 and 100), and PROC FREQTAB converts that number to a proportion. PROC FREQTAB treats the value 1 as 1%. By default, P=0.5.

SUP
SUPERIORITY
requests a test of superiority for the binomial proportion. For more information, see the section "Superiority Test" on page 498. You can specify the superiority test margin, the null proportion, and the variance type in the MARGIN=, P=, and VAR= *binomial-options*, respectively. To request an exact superiority test, you can specify the BINOMIAL option in the EXACT statement.

VAR=NULL | SAMPLE
specifies the type of variance to use in the Wald tests of noninferiority, superiority, and equivalence. If you specify VAR=SAMPLE, PROC FREQTAB computes the variance estimate by using the sample proportion. If you specify VAR=NULL, PROC FREQTAB computes a test-based variance by using the null hypothesis proportion (which you can specify in the P= *binomial-option*). For more information, see the sections "Noninferiority Test" on page 496 and "Equivalence Test" on page 498. By default, VAR=SAMPLE.

CELLCHI2
displays cell chi-squares in the crosstabulation table. A cell chi-square is the table cell's contribution to the Pearson chi-square statistic. The cell chi-square is computed as $(frequency - expected)^2/expected$, where *frequency* is the table cell frequency (count) and *expected* is the expected cell frequency, which is computed under the null hypothesis that the row and column variables are independent. For more information, see the section "Pearson Chi-Square Test for Two-Way Tables" on page 477.

This option applies to two-way and multiway tables that are displayed in the default crosstabulation cell format or in CROSSLIST format. This option has no effect for one-way frequency tables or for tables that are produced by the LIST option. To display cell chi-squares in one-way frequency tables, you can specify the ONEWAY(CELLCHI2) option.

CHISQ < (*chisq-options***) >**
requests chi-square tests of homogeneity or independence and measures of association that are based on the chi-square statistic. For two-way tables, the chi-square tests include the Pearson chi-square, likelihood ratio chi-square, and Mantel-Haenszel chi-square tests. The chi-square measures include the phi coefficient, contingency coefficient, and Cramér's V. For 2×2 tables, the CHISQ option also provides Fisher's exact test and the continuity-adjusted chi-square test. For more information, see the section "Chi-Square Tests and Statistics" on page 476.

For one-way tables, the CHISQ option provides the Pearson chi-square goodness-of-fit test. You can also request the likelihood ratio goodness-of-fit test for one-way tables by specifying the LRCHI *chisq-option* in parentheses after the CHISQ option. By default, the one-way chi-square tests are based on the null hypothesis of equal proportions. Alternatively, you can provide null hypothesis proportions or frequencies by specifying the TESTP= or TESTF= *chisq-option*, respectively. See the section "Chi-Square Test for One-Way Tables" on page 477 for more information.

To request Fisher's exact test for tables larger than 2×2, specify the FISHER option in the EXACT statement. Exact p-values are also available for the Pearson, likelihood ratio, and Mantel-Haenszel chi-square tests. See the description of the EXACT statement for more information.

You can specify the following *chisq-options*:

DF=*df*

specifies the degrees of freedom for the chi-square tests. The value of *df* must not be 0. If the value of *df* is positive, PROC FREQTAB uses *df* as the degrees of freedom for the chi-square tests. If the value of *df* is negative, PROC FREQTAB uses *df* to adjust the default degrees of freedom for the chi-square tests.

By default for one-way tables, the value of *df* is $(n - 1)$, where n is the number of variable levels in the table. By default for two-way tables, the value of *df* is $(r - 1)\ (c - 1)$, where r is the number of rows in the table and c is the number of columns. See the sections "Chi-Square Test for One-Way Tables" on page 477 and "Chi-Square Tests and Statistics" on page 476 for more information.

If you specify a negative value of *df*, PROC FREQTAB adjusts the default degrees of freedom by adding the (negative) value of *df* to the default value to produce the adjusted degrees of freedom. The adjusted degrees of freedom must be positive.

The DF= *chisq-option* specifies or adjusts the degrees of freedom for the following chi-square tests: the Pearson and likelihood ratio goodness-of-fit tests for one-way tables; and the Pearson, likelihood ratio, and Mantel-Haenszel chi-square tests for two-way tables.

LRCHI

requests the likelihood ratio goodness-of-fit test for one-way tables. See the section "Likelihood Ratio Chi-Square Test for One-Way Tables" on page 478 for more information.

By default, this test is based on the null hypothesis of equal proportions. You can provide null hypothesis proportions or frequencies by specifying the TESTP= or TESTF= *chisq-option*, respectively. You can request an exact likelihood ratio goodness-of-fit test by specifying the LRCHI option in the EXACT statement.

TESTF=(*values***)**

specifies null hypothesis frequencies for chi-square tests for one-way tables (goodness-of-fit tests). For more information, see the section "Chi-Square Test for One-Way Tables" on page 477.

You can separate the *values* by using blanks or commas, and you must enclose the list of values in parentheses. The *values* must be positive numbers. The number of *values* must equal the number of variable levels in the one-way table. The sum of the *values* must equal the total frequency in the one-way table. List the *values* in the same order in which the corresponding variable levels appear in the output.

TESTP=(*values***)**

specifies null hypothesis proportions for chi-square tests for one-way tables (goodness-of-fit tests). For more information, see the section "Chi-Square Test for One-Way Tables" on page 477.

You can separate the *values* by using blanks or commas, and you must enclose the list of values in parentheses. The *values* must be positive numbers. The number of *values* must equal the number of variable levels in the one-way table. List the *values* in the same order in which the corresponding variable levels appear in the output.

You can specify *values* in probability form as numbers between 0 and 1, where the proportions sum to 1. Or you can specify *values* in percentage form as numbers between 0 and 100, where the percentages sum to 100. PROC FREQTAB treats the value 1 as the percentage form 1%.

WARN=*type* | **(***types***)**
controls the warning message for the validity of the asymptotic Pearson chi-square test. By default, PROC FREQTAB displays a warning message when more than 20% of the table cells have expected frequencies that are less than 5. If you specify the NOPRINT option in the PROC FREQTAB statement, the procedure displays the warning in the log; otherwise, the procedure displays the warning as a footnote in the chi-square table. You can use the WARN= option to suppress the warning and to include a warning indicator in the output data set.

You can specify one or more of the following *types* in the WARN= option. If you specify more than one *type* value, enclose the values in parentheses after WARN=. For example, `warn = (output noprint)`.

Value of WARN=	Description
OUTPUT	Adds a warning indicator variable to the output data set
NOLOG	Suppresses the chi-square warning message in the log
NOPRINT	Suppresses the chi-square warning message in the display
NONE	Suppresses the chi-square warning message entirely

If you specify the WARN=OUTPUT option, the ODS output data set ChiSq contains a variable named Warning that equals 1 for the Pearson chi-square observation when more than 20% of the table cells have expected frequencies that are less than 5 and equals 0 otherwise. If you specify WARN=OUTPUT and also specify the CHISQ option in the OUTPUT statement, the statistics output data set contains a variable named WARN_PCHI that indicates the warning.

The WARN=NOLOG option has the same effect as the NOWARN option in the TABLES statement.

CL
requests confidence limits for the measures of association, which you can request by specifying the MEASURES option. For more information, see the sections "Measures of Association" on page 482 and "Confidence Limits" on page 482. You can set the level of the confidence limits by using the ALPHA= option; by default, ALPHA=0.05, which produces 95% confidence limits.

If you omit the MEASURES option, the CL option invokes MEASURES. The CL option is equivalent to the MEASURES(CL) option.

CMH <(*cmh-options***)>**
requests Cochran-Mantel-Haenszel statistics, which test for association between the row and column variables after adjusting for the remaining variables in a multiway table. The Cochran-Mantel-Haenszel statistics include the nonzero correlation statistic, the row mean scores (ANOVA) statistic, and the general association statistic. In addition, for 2×2 tables, the CMH option provides the adjusted Mantel-Haenszel and logit estimates of the odds ratio and relative risks, together with their confidence limits. For stratified 2×2 tables, the CMH option provides the Breslow-Day test for homogeneity of odds ratios. (To request Tarone's adjustment for the Breslow-Day test, specify the BDT *cmh-option*.) For more information, see the section "Cochran-Mantel-Haenszel Statistics" on page 536.

You can use the CMH1 or CMH2 option to control the number of CMH statistics that PROC FREQTAB computes.

For stratified 2×2 tables, you can request Zelen's exact test for equal odds ratios by specifying the EQOR option in the EXACT statement. For more information, see the section "Zelen's Exact Test for Equal Odds Ratios" on page 543. You can request exact confidence limits for the common odds ratio by specifying the COMOR option in the EXACT statement. This option also provides a common odds ratio test. For more information, see the section "Exact Confidence Limits for the Common Odds Ratio" on page 544.

You can specify the following *cmh-options* in parentheses after the CMH option. These *cmh-options* are also available with the CMH1 or CMH2 option.

BDT
requests Tarone's adjustment in the Breslow-Day test for homogeneity of odds ratios. For more information, see the section "Breslow-Day Test for Homogeneity of the Odds Ratios" on page 542.

GAILSIMON < (COLUMN=1 | 2) >
GS < (COLUMN=1 | 2) >
requests the Gail-Simon test for qualitative interaction, which applies to stratified 2×2 tables. For more information, see the section "Gail-Simon Test for Qualitative Interactions" on page 546.

The COLUMN= option specifies the column of the risk differences to use to compute the Gail-Simon test. By default, PROC FREQTAB uses column 1 risk differences. If you specify COLUMN=2, PROC FREQTAB uses column 2 risk differences.

The GAILSIMON *cmh-option* is equivalent to the GAILSIMON option in the TABLES statement.

I2
requests the I-square measure of heterogeneity for stratified 2×2 tables. I-square is computed from a Q test that is based on odds ratios. The I2 *cmh-option* invokes the QOR *cmh-option*. For more information, see the section "I-Square Measure of Heterogeneity" on page 543.

MANTELFLEISS
MF
requests the Mantel-Fleiss criterion for the Mantel-Haenszel statistic for stratified 2×2 tables. For more information, see the section "Mantel-Fleiss Criterion" on page 539.

QOR
requests a Q test for heterogeneity of odds ratios for stratified 2×2 tables. For more information, see the section "Q Test for Homogeneity of Odds Ratios" on page 542.

CMH1 < (*cmh-options*) >
requests the Cochran-Mantel-Haenszel correlation statistic. This option does not provide the CMH row mean scores (ANOVA) statistic or the general association statistic, which are provided by the CMH option. For tables larger than 2×2, the CMH1 option requires less memory than the CMH option, which can require an enormous amount of memory for large tables.

For 2×2 tables, the CMH1 option also provides the adjusted Mantel-Haenszel and logit estimates of the odds ratio and relative risks, together with their confidence limits. For stratified 2×2 tables, the CMH1 option provides the Breslow-Day test for homogeneity of odds ratios.

The *cmh-options* for CMH1 are the same as the *cmh-options* that are available with the CMH option. For more information, see the description of the CMH option.

CMH2 < (*cmh-options*) >
requests the Cochran-Mantel-Haenszel correlation statistic and the row mean scores (ANOVA) statistic. This option does not provide the CMH general association statistic, which is provided by the CMH option. For tables larger than 2×2, the CMH2 option requires less memory than the CMH option, which can require an enormous amount of memory for large tables.

For 2×2 tables, the CMH1 option also provides the adjusted Mantel-Haenszel and logit estimates of the odds ratio and relative risks, together with their confidence limits. For stratified 2×2 tables, the CMH1 option provides the Breslow-Day test for homogeneity of odds ratios.

The *cmh-options* for CMH2 are the same as the *cmh-options* that are available with the CMH option. For more information, see the description of the CMH option.

COMMONRISKDIFF < *options* >
requests the common (stratified) risk difference for multiway 2×2 tables, where the risk difference is the difference between the row 1 proportion and the row 2 proportion in a 2×2 table. By default, this option provides Mantel-Haenszel and summary score estimates of the common risk difference, together with their confidence limits. For more information, see the section "Common Risk Difference" on page 511.

You can specify the following *options* to request confidence limit types and tests for the common risk difference:

CL=*type* | (*types*)
requests confidence limits for the common risk difference. You can specify one or more *types* of confidence limits. When you specify only one *type*, you can omit the parentheses. You can specify CL=NONE to suppress the "Confidence Limits for the Common Risk Difference" table.

You can specify the confidence level in the ALPHA= option. By default, ALPHA=0.05, which produces 95% confidence limits for the common risk difference.

You can specify one or more of the following *types*:

K

KLINGENBERG
requests Klingenberg confidence limits for the Mantel-Haenszel common risk difference. For more information, see the section "Klingenberg Confidence Limits" on page 512.

MH
requests Mantel-Haenszel confidence limits, which are computed by using Mantel-Haenszel stratum weights and the Sato variance estimator (Sato 1989). For more information, see the section "Mantel-Haenszel Confidence Limits and Test" on page 511.

MN
requests Miettinen-Nurminen confidence limits for the common risk difference (Miettinen and Nurminen 1985).

MNMH
requests confidence limits that are computed by using the Miettinen-Nurminen method with Mantel-Haenszel weights (Lu 2008).

MR

MINRISK
requests minimum risk confidence limits, which are computed by using minimum risk weights. For more information, see the section "Minimum Risk Confidence Limits and Test" on page 513.

NEWCOMBE
requests stratified Newcombe confidence limits that use Mantel-Haenszel weights to combine the stratum components. For more information, see the section "Stratified Newcombe Confidence Limits" on page 515.

NEWCOMBEMR
requests stratified Newcombe confidence limits that use minimum risk weights to combine the stratum components. For more information, see the section "Stratified Newcombe Confidence Limits" on page 515.

NONE
suppresses the "Confidence Limits for the Common Risk Difference" table.

SCORE
requests summary score confidence limits. For more information, see the section "Summary Score Confidence Limits" on page 514.

COLUMN=1 | 2
specifies the table column for which to compute the common risk difference statistics.

By default, COLUMN=1 unless you specify the column for the risk difference plot or for the stratum-level risk difference statistics. In this case, when you omit the COMMONRISKDIFF(COLUMN=) option, PROC FREQTAB provides the common risk difference statistics for the same column that you specify for the plot or for the stratum-level statistics.

CORRECT=NO
removes the continuity correction in the minimum risk confidence limits and in the minimum risk test, which you can request by specifying the CL=MR and TEST=MR options, respectively. For more information, see the section "Minimum Risk Confidence Limits and Test" on page 513.

PRINTWTS < =*type* | (*types*) >
displays the stratum weights together with the stratum risk differences and frequencies. By default, this option displays the weight type or types for the confidence limits and tests that you request. Optionally, you can specify the weight type to display.

You can specify one or more of the following *types*:

MH
displays Mantel-Haenszel stratum weights. For more information, see the section "Mantel-Haenszel Confidence Limits and Test" on page 511.

MN
displays Miettinen-Nurminen stratum weights.

MR
displays minimum risk stratum weights. For more information, see the section "Minimum Risk Confidence Limits and Test" on page 513.

SCORE
displays summary score stratum weights. For more information, see the section "Summary Score Confidence Limits" on page 514.

TEST < =*type* | **(***types***) >**
requests common risk difference tests. You can specify one or more *types*. When you specify only one *type*, you can omit the parentheses. If you do not specify *types*, this option provides tests that correspond to the confidence limit *types* that you specify in the CL= option.

You can specify one or more of the following *types*:

MH
requests a Mantel-Haenszel test, which is computed by using Mantel-Haenszel stratum weights and the Sato variance estimator (Sato 1989). For more information, see the section "Mantel-Haenszel Confidence Limits and Test" on page 511.

MR < (VAR=SAMPLE) >

MINRISK < (VAR=SAMPLE) >
requests the minimum risk test, which is computed by using minimum risk weights. If you specify VAR=SAMPLE, PROC FREQTAB uses the sample (observed) variance estimate instead of a null variance estimate to compute the minimum risk test statistic. For more information, see the section "Minimum Risk Confidence Limits and Test" on page 513.

SCORE
requests the summary score test. For more information, see the section "Summary Score Confidence Limits" on page 514.

CQ

COCHQ
requests Cochran's Q test of marginal homogeneity, which can be computed for multiway tables in which each variable has two levels ($2 \times 2 \times \cdots \times 2$ tables). For more information, see the section "Cochran's Q Test" on page 535.

The AGREE option provides Cochran's Q test together with several other measures of agreement. You can use the CQ option to compute Cochran's Q test without the other measures of agreement.

CROSSLIST <(*options*)>

displays crosstabulation tables by using an ODS column format instead of the default crosstabulation cell format. In CROSSLIST tables, the rows correspond to the crosstabulation table cells, and the columns correspond to descriptive statistics such as frequencies and percentages. For more information about the contents of CROSSLIST tables, see the section "Two-Way and Multiway Tables" on page 556.

By default, CROSSLIST tables display all levels of the column variable within each level of the row variable, including any levels that have frequencies of 0. By default, multiway CROSSLIST tables display all levels of the row variable within each stratum, including any row levels that have frequencies of 0 in the stratum. To suppress variable levels that have frequencies of 0, you can specify the NOSPARSE option.

You cannot specify both the LIST option and the CROSSLIST option in the same TABLES statement.

You can specify the following *options*:

CELLCHI2

displays cell chi-squares in the CROSSLIST table. A cell chi-square is the table cell's contribution to the Pearson chi-square statistic. The cell chi-square is computed as $(frequency - expected)^2/expected$, where *frequency* is the table cell frequency (count) and *expected* is the expected cell frequency, which is computed under the null hypothesis that the row and column variables are independent. For more information, see the section "Pearson Chi-Square Test for Two-Way Tables" on page 477.

DEVIATION

displays deviations in the CROSSLIST table. A deviation is the difference between the observed table cell frequency and the expected frequency (*frequency* – *expected*). The expected frequencies are computed under the null hypothesis that the row and column variables are independent. For more information, see the section "Pearson Chi-Square Test for Two-Way Tables" on page 477. You can display the expected frequencies in the CROSSLIST table by specifying the CROSSLIST(EXPECTED) option.

EXPECTED

displays expected frequencies in the CROSSLIST table. Expected cell frequencies are computed as the product of the corresponding row totals and column totals divided by the total frequency of the two-way table. These are the expected frequencies under the null hypothesis that the row and column variables are independent. For more information, see the section "Pearson Chi-Square Test for Two-Way Tables" on page 477.

FORMAT=*format-name*

specifies the format for frequencies, expected frequencies, and deviations in the CROSSLIST table.

You can specify any standard SAS numeric format, or you can specify a numeric format that is defined by the FORMAT procedure. The format length must not exceed 24. By default, PROC FREQTAB uses the BEST8. format to display frequencies in the CROSSLIST table.

To change formats in any table that PROC FREQTAB displays, you can use PROC TEMPLATE. For more information, see the chapter "The TEMPLATE Procedure" in the *SAS Output Delivery System: User's Guide*.

FORMATP=*format-name*
specifies the format for percentages in the CROSSLIST table.

You can specify any standard SAS numeric format, or you can specify a numeric format that is defined by the FORMAT procedure. The format length must not exceed 24. By default, PROC FREQTAB uses the format 6.2 to display percentages in the CROSSLIST table.

To change formats in any table that PROC FREQTAB displays, you can use PROC TEMPLATE. For more information, see the chapter "The TEMPLATE Procedure" in the *SAS Output Delivery System: User's Guide*.

NOCOL
suppresses the display of column percentages in the CROSSLIST table.

NOFREQ
suppresses the display of frequencies in the CROSSLIST table.

NOPERCENT
suppresses the display of table percentages (the Percent column) in the CROSSLIST table. To suppress the display of row or column percentages, you can specify the CROSSLIST(NOROW) or CROSSLIST(NOCOL) option, respectively.

NOROW
suppresses the display of row percentages in the CROSSLIST table.

PEARSONRES
displays Pearson residuals in the CROSSLIST table. The Pearson residual is the square root of the table cell's contribution to the Pearson chi-square statistic. The Pearson residual is computed as $(frequency - expected)/\sqrt{expected}$, where *frequency* is the table cell frequency (count) and *expected* is the expected table cell frequency, which is computed under the null hypothesis that the row and column variables are independent. For more information, see the section "Pearson Chi-Square Test for Two-Way Tables" on page 477.

You can display the expected values, deviations, and cell chi-squares in the CROSSLIST table by specifying the CROSSLIST(EXPECTED), CROSSLIST(DEVIATION), and CROSSLIST(CELLCHI2) options, respectively.

STDRES
displays standardized residuals in the CROSSLIST table. The standardized residual is the ratio of (*frequency* – *expected*) to its standard error, where *frequency* is the table cell frequency (count) and *expected* is the expected table cell frequency, which is computed under the null hypothesis that the row and column variables are independent. For more information, see the section "Standardized Residuals" on page 478.

You can display the expected values and deviations in the CROSSLIST table by specifying the CROSSLIST(EXPECTED) and CROSSLIST(DEVIATION) options, respectively.

TOTPCT

displays percentages of the total multiway table frequency for multiway tables (n-way tables where $n > 2$). By default, the CROSSLIST table displays percentages of the two-way table (stratum) frequency but does not display percentages of the total multiway table frequency. For more information, see the section "Two-Way and Multiway Tables" on page 556.

CUMCOL

displays cumulative column percentages in the crosstabulation table. This option applies to two-way and multiway tables that are displayed in the default crosstabulation cell format.

DEVIATION

displays deviations in the crosstabulation table. A deviation is the difference between the observed table cell frequency and the expected frequency (*frequency – expected*). The expected frequencies are computed under the null hypothesis that the row and column variables are independent. For more information, see the section "Pearson Chi-Square Test for Two-Way Tables" on page 477. You can display the expected values by specifying the EXPECTED option.

This option applies to two-way and multiway tables that are displayed in the default crosstabulation cell format or in CROSSLIST format. To display deviations in one-way frequency tables, you can specify the ONEWAY(DEVIATION) option.

EXPECTED

displays expected frequencies in the crosstabulation table. Expected table cell frequencies are computed as the product of the corresponding row totals and column totals divided by the total frequency of the two-way table. These are the expected frequencies under the null hypothesis that the row and column variables are independent. For more information, see the section "Pearson Chi-Square Test for Two-Way Tables" on page 477.

This option applies to two-way and multiway tables that are displayed in the default crosstabulation cell format or in CROSSLIST format. To display expected frequencies in one-way frequency tables, you can specify the ONEWAY(EXPECTED) option.

FISHER

requests Fisher's exact test for tables that are larger than 2×2. (For 2×2 tables, the CHISQ option provides Fisher's exact test.) This test is also known as the Freeman-Halton test. See the sections "Fisher's Exact Test" on page 480 and "Exact Statistics" on page 547 for more information.

If you omit the CHISQ option in the TABLES statement, the FISHER option invokes CHISQ. You can also request Fisher's exact test by specifying the FISHER option in the EXACT statement.

NOTE: PROC FREQTAB computes exact tests by using fast and efficient algorithms that are superior to direct enumeration. Exact tests are appropriate when a data set is small, sparse, skewed, or heavily tied. For some large problems, computation of exact tests might require a substantial amount of time and memory. Consider using asymptotic tests for such problems. Alternatively, when asymptotic methods might not be sufficient for such large problems, consider using Monte Carlo estimation of exact p-values. You can request Monte Carlo estimation by specifying the MC *computation-option* in the EXACT statement. See the section "Computational Resources" on page 549 for more information.

FORMAT=*format-name*
specifies the format for frequencies, expected frequencies, and deviations in crosstabulation tables.

You can specify any standard SAS numeric format, or you can specify a numeric format that is defined by the FORMAT procedure. The format length must not exceed 24.

This option applies to crosstabulation tables that are displayed in default crosstabulation cell form, CROSSLIST form, or LIST form. You can use the ONEWAY(FORMAT=) option to specify the format for frequencies in one-way tables.

To change display formats in any table that PROC FREQTAB produces, you can use PROC TEMPLATE. For more information, see the chapter "The TEMPLATE Procedure" in the *SAS Output Delivery System: User's Guide*.

FORMATP=*format-name*
specifies the format for percentages in crosstabulation tables.

You can specify any standard SAS numeric format, or you can specify a numeric format that is defined by the FORMAT procedure. The format length must not exceed 24.

This option applies to crosstabulation tables that are displayed in default crosstabulation cell form, CROSSLIST form, or LIST form. You can use the ONEWAY(FORMATP=) option to specify the format for percentages in one-way frequency tables.

To change display formats in any table that PROC FREQTAB produces, you can use PROC TEMPLATE. For more information, see the chapter "The TEMPLATE Procedure" in the *SAS Output Delivery System: User's Guide*.

GAILSIMON < (COLUMN=1 | 2) >
GS < (COLUMN=1 | 2) >
requests the Gail-Simon test for qualitative interaction, which applies to stratified 2×2 tables. For more information, see the section "Gail-Simon Test for Qualitative Interactions" on page 546.

The COLUMN= option specifies the column of the risk differences to use to compute the Gail-Simon test. By default, PROC FREQTAB uses column 1 risk differences. If you specify COLUMN=2, PROC FREQTAB uses column 2 risk differences.

JT
requests the Jonckheere-Terpstra test. For more information, see the section "Jonckheere-Terpstra Test" on page 527. To request exact *p*-values for the Jonckheere-Terpstra test, specify the JT option in the EXACT statement. See the section "Exact Statistics" on page 547 for more information.

LIST < (*options*) >
displays two-way and multiway tables by using a list format instead of the default crosstabulation cell format. This option displays an entire multiway table in a single table instead of separate two-way (stratum) tables. In list format, each row of the table corresponds to a single crosstabulation table cell. For more information, see the section "Two-Way and Multiway Tables" on page 556.

By default, a table in list format does not display cells for which the frequency is 0 unless you specify the ZEROS option in the WEIGHT statement and do not specify the NOSPARSE option in the TABLES statement.

The LIST option is not available in the same TABLES statement together with statistic options. When you specify statistic options in a TABLES statement, you can display the crosstabulation tables by

using the default format or the CROSSLIST format. However, you can specify the LIST option and statistic options in different TABLES statements in the same invocation of PROC FREQTAB.

You can specify the following *options*:

FORMAT=*format-name*
specifies the format for frequencies and cumulative frequencies in the LIST crosstabulation table.

You can specify any standard SAS numeric format, or you can specify a numeric format that is defined by the FORMAT procedure. The format length must not exceed 24. By default, PROC FREQTAB uses the BEST8. format to display frequencies in the LIST crosstabulation table.

To change display formats in any table that PROC FREQTAB produces, you can use PROC TEMPLATE. For more information, see the chapter "The TEMPLATE Procedure" in the *SAS Output Delivery System: User's Guide*.

FORMATP=*format-name*
specifies the format for percentages and cumulative percentages in the LIST crosstabulation table.

You can specify any standard SAS numeric format, or you can specify a numeric format that is defined by the FORMAT procedure. The format length must not exceed 24. By default, PROC FREQTAB uses the format 6.2 to display percentages in the LIST table.

To change display formats in any table that PROC FREQTAB produces, you can use PROC TEMPLATE. For more information, see the chapter "The TEMPLATE Procedure" in the *SAS Output Delivery System: User's Guide*.

NOCUM
suppresses the display of cumulative frequencies and cumulative percentages in the LIST crosstabulation table.

NOFREQ
suppresses the display of frequencies and cumulative frequencies in the LIST crosstabulation table.

NOPERCENT
suppresses the display of percentages and cumulative percentages in the LIST crosstabulation table.

MAXLEVELS=*n*
specifies the maximum number of variable levels to display in one-way frequency tables. The value of *n* must be a positive integer. PROC FREQTAB displays the first *n* variable levels, matching the order in which the levels appear in the one-way frequency table. (The ORDER= option controls the order of the variable levels. By default, ORDER=INTERNAL, which orders the variable levels by unformatted value.)

The MAXLEVELS= option also applies to one-way frequency plots, which you can request by specifying the PLOTS=FREQPLOT option when ODS Graphics is enabled.

If you specify the MISSPRINT option to display missing levels in frequency tables, MAXLEVELS=*n* displays the first *n* nonmissing levels.

The MAXLEVELS= option does not apply to the OUT= output data set, which includes all variable levels. The MAXLEVELS= option does not affect the computation of percentages, statistics, or tests for the one-way table; these values are based on the complete table.

MEASURES <(CL)>

requests measures of association and their asymptotic standard errors. This option provides the following measures: gamma, Kendall's tau-*b*, Stuart's tau-*c*, Somers' $D(C|R)$, Somers' $D(R|C)$, Pearson and Spearman correlation coefficients, lambda (symmetric and asymmetric), and uncertainty coefficients (symmetric and asymmetric). If you specify the CL option in parentheses after the MEASURES option, PROC FREQTAB provides confidence limits for the measures of association. For more information, see the section "Measures of Association" on page 482.

For 2×2 tables, the MEASURES option also provides the odds ratio, column 1 relative risk, column 2 relative risk, and their asymptotic Wald confidence limits. You can request the odds ratio and relative risks separately (without the other measures of association) by specifying the RELRISK option. You can request confidence limits for the odds ratio by specifying the OR(CL=) option.

You can use the TEST statement to request asymptotic tests for the following measures of association: gamma, Kendall's tau-*b*, Stuart's tau-*c*, Somers' $D(C|R)$, Somers' $D(R|C)$, and Pearson and Spearman correlation coefficients. You can use the EXACT statement to request exact confidence limits for the odds ratio, exact unconditional confidence limits for the relative risks, and exact tests for the following measures of association: Kendall's tau-*b*, Stuart's tau-*c*, Somers' $D(C|R)$ and $D(R|C)$, and Pearson and Spearman correlation coefficients. For more information, see the descriptions of the TEST and EXACT statements and the section "Exact Statistics" on page 547.

NOCOL

suppresses the display of column percentages in crosstabulation tables. A table cell's column percentage is computed as the cell frequency divided by the corresponding column frequency.

NOCUM

suppresses the display of cumulative frequencies and percentages in one-way frequency tables and in list-format crosstabulation tables (which you can request by specifying the LIST option).

NOFREQ

suppresses the display of table cell frequencies in crosstabulation tables. This option also suppresses row total frequencies.

In list-format crosstabulation tables (which you can request by specifying the LIST option), the NOFREQ option suppresses the display of frequencies and cumulative frequencies. To suppress display of frequencies in one-way frequency tables, you can specify the ONEWAY(NOFREQ) option.

NOPERCENT

suppresses the display of overall percentages in crosstabulation tables. These percentages include table cell, row, and column percentages of the total two-way table frequency. To suppress display of cell percentages of row or column totals, you can specify the NOROW or NOCOL option, respectively.

For one-way frequency tables and list-format crosstabulation tables, the NOPERCENT option suppresses the display of percentages and cumulative percentages.

NOPRINT

suppresses the display of frequency and crosstabulation tables but displays all requested tests and statistics. To suppress the display of all output, including tests and statistics, use the NOPRINT option in the PROC FREQTAB statement.

NOROW
suppresses the display of row percentages in crosstabulation tables. A table cell's row percentage is computed as the cell frequency divided by the corresponding row frequency.

NOSPARSE
suppresses the display of zero-frequency table cells in LIST and CROSSLIST tables. This option also excludes zero-frequency table cells from OUT= output data sets.

When you specify the ZEROS option in the WEIGHT statement, PROC FREQTAB includes observations that have weights of 0 in the analysis. By default, zero-frequency table cells are displayed in the LIST table and included in the OUT= data set. You can use the NOSPARSE option to exclude the zero-frequency table cells.

When you specify the CROSSLIST option, by default the CROSSLIST table displays all levels of the column variable within each level of the row variable (including any levels that have frequencies of 0). By default for multiway CROSSLIST tables, the CROSSLIST table displays all levels of the row variable within each stratum of the table (including any row levels that have frequencies of 0 in the stratum). To suppress variable levels in the CROSSLIST table that have frequencies of 0, you can specify the NOSPARSE option.

NOWARN
suppresses the log warning message for the validity of the asymptotic Pearson chi-square test. By default, PROC FREQTAB provides a validity warning for the asymptotic Pearson chi-square test when more than 20% of the table cells have expected frequencies that are less than 5. This warning message appears in the log if you specify the NOPRINT option in the PROC FREQTAB statement,

The NOWARN option is equivalent to the CHISQ(WARN=NOLOG) option. You can also use the CHISQ(WARN=) option to suppress the warning message in the display and to request a warning variable in the chi-square ODS output data set or in the OUTPUT data set.

ONEWAY (*options*)
controls the statistics to display in one-way frequency tables. For more information about the contents of one-way frequency tables, see the section "One-Way Frequency Tables" on page 554.

You can specify the following *options*:

CELLCHI2
displays cell chi-squares in the one-way frequency table. A cell chi-square is the table cell's contribution to the Pearson chi-square statistic. The cell chi-square is computed as $(frequency - expected)^2/expected$, where *frequency* is the table cell (level) frequency and *expected* is the expected frequency. For more information, see the section "Chi-Square Test for One-Way Tables" on page 477.

DEVIATION
displays deviations in the one-way frequency table. A deviation is the difference between the observed table cell (level) frequency and the expected frequency (*frequency – expected*), where the expected frequencies are the null hypothesis frequencies for the chi-square goodness-of-fit test.

By default, the expected frequency for each class level is the total frequency divided by the number of levels. Alternatively, you can provide null frequencies or null proportions by specifying the CHISQ(TESTF=) or CHISQ(TESTP=) option, respectively. For more information, see the section "Chi-Square Test for One-Way Tables" on page 477.

EXPECTED
displays expected frequencies in the one-way frequency table, where the expected frequencies are the null hypothesis frequencies for the chi-square goodness-of-fit test.

By default, the expected frequency for each class level is the total frequency divided by the number of levels. Alternatively, you can provide null frequencies or null proportions by specifying the CHISQ(TESTF=) or CHISQ(TESTP=) option, respectively. For more information, see the section "Chi-Square Test for One-Way Tables" on page 477.

When you specify the CHISQ(TESTF=) option, the one-way frequency table displays the specified null hypothesis frequencies by default, and the EXPECTED option has no effect.

FORMAT=*format-name*
specifies the format for frequencies, expected frequencies, and deviations in the one-way frequency table.

You can specify any standard SAS numeric format, or you can specify a numeric format that is defined by the FORMAT procedure. The format length must not exceed 24. By default, PROC FREQTAB uses the BEST8. format to display frequencies in the one-way frequency table.

To change display formats in any table that PROC FREQTAB produces, you can use PROC TEMPLATE. For more information, see the chapter "The TEMPLATE Procedure" in the *SAS Output Delivery System: User's Guide*.

FORMATP=*format-name*
specifies the format for percentages in the one-way frequency table.

You can specify any standard SAS numeric format, or you can specify a numeric format that is defined by the FORMAT procedure. The format length must not exceed 24. By default, PROC FREQTAB uses the format 6.2 to display percentages in the one-way frequency table.

To change display formats in any table that PROC FREQTAB produces, you can use PROC TEMPLATE. For more information, see the chapter "The TEMPLATE Procedure" in the *SAS Output Delivery System: User's Guide*.

MAXLEVELS=*n*
specifies the maximum number of variable levels to display in the one-way frequency table. The value of *n* must be a positive integer. PROC FREQTAB displays the first *n* variable levels, matching the order in which the levels appear in the one-way frequency table. (The ORDER= option controls the order of the variable levels. By default, ORDER=INTERNAL, which orders the variable levels by unformatted values.)

The MAXLEVELS= option also applies to one-way frequency plots, which you can request by specifying the PLOTS=FREQPLOT option when ODS Graphics is enabled.

If you specify the MISSPRINT option to display missing levels in frequency tables, MAXLEVELS=*n* displays the first *n* nonmissing levels.

The MAXLEVELS= option does not apply to the OUT= output data set, which includes all variable levels. The MAXLEVELS= option does not affect the computation of percentages, statistics, or tests for the one-way table; these values are based on the complete table.

NOCUM
suppresses the display of cumulative frequencies and percentages in one-way frequency tables.

NOFREQ
suppresses the display of frequencies and cumulative frequencies in one-way frequency tables.

NOPERCENT
suppresses the display of percentages and cumulative percentages in one-way frequency tables.

RESIDUAL
displays Pearson residuals in the one-way frequency table. The Pearson residual is the square root of the table cell's contribution to the Pearson chi-square statistic. The Pearson residual is computed as $(frequency - expected)/\sqrt{expected}$, where *frequency* is the table cell frequency (count) and *expected* is the expected table cell frequency.

OR < (CL=*type* | (*types*) >
ODDSRATIO < (CL=*type* | (*types*) >
requests the odds ratio and confidence limits for 2×2 tables. For more information, see the section "Odds Ratio" on page 516.

You can specify one or more *types* of confidence limits. When you specify only one confidence limit *type*, you can omit the parentheses around the request. PROC FREQTAB displays the confidence limits in the "Confidence Limits for the Odds Ratio" table.

Specifying the OR option without the CL= option is equivalent to specifying the RELRISK option, which produces the "Odds Ratio and Relative Risks" table. For more information, see the description of the RELRISK option. When you specify the OR(CL=) option, PROC FREQTAB does not produce the "Odds Ratio and Relative Risks" table unless you also specify the RELRISK or MEASURES option.

The ALPHA= option determines the confidence level; by default, ALPHA=0.05, which produces 95% confidence limits for the odds ratio.

You can specify the following *types*:

EXACT
displays exact confidence limits for the odds ratio in the "Confidence Limits for the Odds Ratio" table. (By default, PROC FREQTAB displays the exact confidence limits in a separate table.) You must also request computation of the exact confidence limits by specifying the OR option in the EXACT statement. For more information, see the subsection "Exact Confidence Limits" in the section "Confidence Limits for the Odds Ratio" on page 516.

LR
LIKELIHOODRATIO
requests likelihood ratio confidence limits for the odds ratio. For more information, see the subsection "Likelihood Ratio Confidence Limits" in the section "Confidence Limits for the Odds Ratio" on page 516.

MIDP
requests exact mid-*p* confidence limits for the odds ratio. For more information, see the subsection "Exact Mid-*p* Confidence Limits" in the section "Confidence Limits for the Odds Ratio" on page 516.

SCORE <(CORRECT=NO)>
requests score confidence limits for the odds ratio. For more information, see the subsection "Score Confidence Limits" in the section "Confidence Limits for the Odds Ratio" on page 516. If you specify CORRECT=NO, PROC FREQTAB provides the uncorrected form of the score confidence limits.

WALD
requests asymptotic Wald confidence limits, which are based on a log transformation of the odds ratio. For more information, see the subsection "Wald Confidence Limits" in the section "Confidence Limits for the Odds Ratio" on page 516.

WALDMODIFIED
requests Wald modified confidence limits for the odds ratio. For more information, see the subsection "Wald Modified Confidence Limits" in the section "Confidence Limits for the Odds Ratio" on page 516.

OUT=*data-table*
names an output data table that contains frequency or crosstabulation table counts and percentages. If more than one table request appears in the TABLES statement, the contents of the OUT= data table correspond to the last table request in the TABLES statement. The OUT= data table variable COUNT contains the frequencies and the variable PERCENT contains the percentages. For more information, see the section "Contents of the TABLES Statement Output Data Set" on page 552. You can specify the following options to include additional information in the OUT= data table: OUTCUM, OUTEXPECT, and OUTPCT.

OUTCUM
includes cumulative frequencies and cumulative percentages in the OUT= data set. The variable CUM_FREQ contains the cumulative frequencies, and the variable CUM_PCT contains the cumulative percentages. For more information, see the section "Contents of the TABLES Statement Output Data Set" on page 552. This option is available for one-way frequency tables and for tables that you display in list format (which you can request by specifying the LIST option).

OUTEXPECT
includes expected cell frequencies in the OUT= data set for crosstabulation tables. The variable EXPECTED contains the expected cell frequencies. For more information, see the section "Contents of the TABLES Statement Output Data Set" on page 552. This option has no effect for one-way tables or for tables that you display in list format (which you can request by specifying the LIST option).

OUTPCT
includes the following additional variables in the OUT= data set for crosstabulation tables:

PCT_COL	percentage of column frequency
PCT_ROW	percentage of row frequency
PCT_TABL	percentage of stratum (two-way table) frequency, for n-way tables where $n > 2$

For more information, see the section "Contents of the TABLES Statement Output Data Set" on page 552. This option has no effect for one-way tables or for tables that you display in list format (which you can request by specifying the LIST option).

PLCORR < (*options*) >

POLYCHORIC < (*options*) >

requests the polychoric correlation coefficient and its asymptotic standard error. For 2×2 tables, this statistic is more commonly known as the tetrachoric correlation coefficient and is labeled as such in the displayed output. For more information, see the section "Polychoric Correlation" on page 488.

If you also specify the CL or MEASURES(CL) option, PROC FREQTAB provides confidence limits for the polychoric correlation. If you specify the PLCORR option in the TEST statement, the procedure provides Wald and likelihood ratio tests for the polychoric correlation. The PLCORR option invokes the MEASURES option.

You can specify the following *options*:

ADJUST

replaces a 2×2 table cell frequency of 0 with 0.5 before computing the tetrachoric correlation (Brown and Benedetti 1977a, p. 353). To maintain the row and column marginal frequencies, adjacent cell frequencies are decreased by 0.5 and the opposite cell frequency is increased by 0.5.

This option is available for 2×2 tables and is applied only when a single cell frequency is 0. It has no effect when both off-diagonal cell frequencies are 0 (and therefore the correlation is 1) or when both diagonal cell frequencies are 1 (and therefore the correlation is –1).

CONVERGE=*value*

specifies the convergence criterion. The *value* must be a positive number. By default, CONVERGE=0.0001. Iterative computation of the polychoric correlation stops when the convergence measure falls below *value* or when the number of iterations exceeds the MAXITER= *number*, whichever happens first. For parameter values that are less than 0.01, PROC FREQTAB evaluates convergence by using the absolute difference instead of the relative difference. For more information, see the section "Polychoric Correlation" on page 488.

MAXITER=*number*

specifies the maximum *number* of iterations. The value of *number* must be a positive integer. By default, MAXITER=50. Iterative computation of the polychoric correlation stops when the number of iterations exceeds the maximum *number* or when the convergence measure falls below the CONVERGE= *value*, whichever happens first. For more information, see the section "Polychoric Correlation" on page 488.

PLOTS < (*global-plot-options*) > < =*plot-request* < (*plot-options*) > >

PLOTS < (*global-plot-options*) > < =(*plot-request* < (*plot-options*) > < . . . *plot-request* < (*plot-options*) > >) >

controls the plots that are produced through ODS Graphics. *Plot-requests* identify the plots, and *plot-options* control the appearance and content of the plots. You can specify *plot-options* in parentheses after a *plot-request*. A *global-plot-option* applies to all plots for which it is available unless it is altered by a specific *plot-option*. You can specify *global-plot-options* in parentheses after the PLOTS option.

When you specify only one *plot-request*, you can omit the parentheses around the request. For example:

```
plots=all
plots=freqplot
plots=(freqplot oddsratioplot)
plots(only)=(cumfreqplot deviationplot)
```

ODS Graphics must be enabled before plots can be requested. For example:

```
ods graphics on;
proc freqtab;
   tables treatment*response / chisq plots=freqplot;
   weight wt;
run;
ods graphics off;
```

If ODS Graphics is enabled but you do not specify the PLOTS= option, PROC FREQTAB produces all plots that are associated with the analyses that you request, with the exception of the frequency, cumulative frequency, and mosaic plots. To produce a frequency plot or cumulative frequency plot when ODS Graphics is enabled, you must specify the FREQPLOT or CUMFREQPLOT *plot-request*, respectively, in the PLOTS= option, or you must specify the PLOTS=ALL option. To produce a mosaic plot when ODS Graphics is enabled, you must specify the MOSAICPLOT *plot-request* in the PLOTS= option, or you must specify the PLOTS=ALL option.

PROC FREQTAB produces the remaining plots (listed in Table 11.10) by default when you request the corresponding TABLES statement options. You can suppress default plots and request specific plots by using the PLOTS(ONLY)= option; PLOTS(ONLY)=(*plot-requests*) produces only the plots that are specified as *plot-requests*. You can suppress all plots by specifying the PLOTS=NONE option. The PLOTS option has no effect when you specify the NOPRINT option in the PROC FREQTAB statement.

Plot Requests

Table 11.10 lists the available *plot-requests* together with their required TABLES statement options. Descriptions of the *plot-requests* follow the table in alphabetical order.

Table 11.10 Plot Requests

Plot Request	Description	Required TABLES Statement Option
AGREEPLOT	Agreement plot	AGREE ($r \times r$ table)
ALL	All plots	None
CUMFREQPLOT	Cumulative frequency plot	One-way table request
DEVIATIONPLOT	Deviation plot	CHISQ (one-way table)
FREQPLOT	Frequency plot	Any table request
KAPPAPLOT	Kappa plot	AGREE ($h \times r \times r$ table)
MOSAICPLOT	Mosaic plot	Two-way or multiway table request
NONE	No plots	None
ODDSRATIOPLOT	Odds ratio plot	MEASURES, OR, or RELRISK ($h \times 2 \times 2$ table)
RELRISKPLOT	Relative risk plot	MEASURES or RELRISK ($h \times 2 \times 2$ table)

Table 11.10 *continued*

Plot Request	Description	Required TABLES Statement Option
RISKDIFFPLOT	Risk difference plot	RISKDIFF ($h \times 2 \times 2$ table)
WTKAPPAPLOT	Weighted kappa plot	AGREE ($h \times r \times r$ table, $r > 2$)

You can specify the following *plot-requests*:

AGREEPLOT < (*plot-options*) >
requests an agreement plot (Bangdiwala and Bryan 1987), An agreement plot displays the strength of agreement in a two-way table, where the row and column variables represent two independent ratings of n subjects. For information about agreement plots, see Bangdiwala (1988), Bangdiwala et al. (2008), and Friendly (2000, Section 3.7.2).

To produce an agreement plot, you must also specify the AGREE option in the TABLES statement. Agreement statistics and plots are available for two-way square tables, where the number of rows equals the number of columns.

Table 11.11 lists the *plot-options* that are available for agreement plots. For descriptions of the *plot-options*, see the subsection "Plot Options" in this section.

Table 11.11 Plot Options for AGREEPLOT

Plot Option	Description	Values
LEGEND=	Legend	NO and YES*
PARTIAL=	Partial agreement	NO and YES*
SHOWSCALE=	Frequency scale	NO and YES*
STATS	Statistics	None

*Default

If you specify the STATS *plot-option*, the agreement plot displays the values of the kappa coefficient, the weighted kappa coefficient, the B_n measure (Bangdiwala and Bryan 1987), and the sample size. PROC FREQTAB stores these statistics in an ODS table named BnMeasure, which is not displayed. For more information, see the section "ODS Table Names" on page 565.

ALL
requests all plots that are associated with the specified analyses. Table 11.10 lists the available *plot-requests* and the corresponding analysis options. If you specify the PLOTS=ALL option, PROC FREQTAB produces the frequency, cumulative frequency, and mosaic plots that are associated with the tables that you request. (These plots are not produced by default when ODS Graphics is enabled.)

CUMFREQPLOT < (*plot-options*) >
requests a plot of cumulative frequencies. Cumulative frequency plots are available for one-way frequency tables.

To produce a cumulative frequency plot, you must specify the CUMFREQPLOT *plot-request* in the PLOTS= option, or you must specify the PLOTS=ALL option. PROC FREQTAB does not produce cumulative frequency plots by default when ODS Graphics is enabled.

Table 11.12 lists the *plot-options* that are available for cumulative frequency plots. For descriptions of the *plot-options*, see the subsection "Plot Options" in this section.

Table 11.12 Plot Options for CUMFREQPLOT

Plot Option	Description	Values
ORIENT=	Orientation	HORIZONTAL and VERTICAL*
SCALE=	Scale	FREQ* and PERCENT
TYPE=	Type	BARCHART* and DOTPLOT

*Default

DEVIATIONPLOT < (*plot-options*) >

requests a plot of relative deviations from expected frequencies. Deviation plots are available for chi-square analysis of one-way frequency tables. To produce a deviation plot, you must also specify the CHISQ option in the TABLES statement for a one-way frequency table.

Table 11.13 lists the *plot-options* that are available for deviation plots.

Table 11.13 Plot Options for DEVIATIONPLOT

Plot Option	Description	Values
NOSTAT	No statistic	None
ORIENT=	Orientation	HORIZONTAL and VERTICAL*
PLOTVAR=	Plot variable	CELLCHI2, DEVIATION*, and RESIDUAL
SCALE=	Scale	ABS, NONE*, and PERCENT
TYPE=	Type	BARCHART* and DOTPLOT

*Default

For descriptions of the NOSTAT, ORIENT=, and TYPE= *plot-options*, see the subsection "Plot Options" in this section.

You can specify the following *plot-options*:

PLOTVAR=CELLCHI2 | DEVIATION | RESIDUAL

specifies the deviation statistic to display in the deviation plot. By default, PLOTVAR=DEVIATION, which displays the relative deviations. For each class level, the relative deviation is computed as the deviation divided by the expected frequency under the null hypothesis ((*frequency* – *expected*)/*expected*).

PLOTVAR=CELLCHI2 displays the cell chi-square contributions, which are computed as $(frequency - expected)^2/expected$. For more information, see the ONEWAY(CELLCHI2) option.

PLOTVAR=RESIDUAL displays the Pearson residuals, which are computed as $(frequency - expected)/\sqrt{expected}$. For more information, see the ONEWAY(RESIDUAL) option.

SCALE=ABS | NONE | PERCENT

specifies the scale of the deviation statistic to display. By default, SCALE=NONE, which displays the unscaled deviations. SCALE=ABS displays the absolute values of the deviation statistics (relative deviations or residuals). SCALE=ABS has no effect for the cell chi-squares, which are nonnegative. SCALE=PERCENT displays the percentages of the total of the absolute deviations.

The SCALE=PERCENT *global-plot-option* invokes the DEVIATIONPLOT(SCALE=PERCENT) *plot-option*. You can alter this by specifying the individual DEVIATIONPLOT(SCALE=) *plot-option*.

FREQPLOT < (*plot-options*) >

requests a frequency plot. Frequency plots are available for frequency and crosstabulation tables. For multiway crosstabulation tables, PROC FREQTAB provides a two-way frequency plot for each stratum (two-way table).

To produce a frequency plot, you must specify the FREQPLOT *plot-request* in the PLOTS= option, or you must specify the PLOTS=ALL option. PROC FREQTAB does not produce frequency plots by default when ODS Graphics is enabled.

Table 11.14 lists the *plot-options* that are available for frequency plots. For descriptions of the *plot-options*, see the subsection "Plot Options" in this section.

Table 11.14 Plot Options for FREQPLOT

Plot Option	Description	Values
GROUPBY=**	Primary group	COLUMN* and ROW
NPANELPOS=**	Sections per panel	Number (4*)
ORIENT=	Orientation	HORIZONTAL and VERTICAL*
SCALE=	Scale	FREQ*, GROUPPERCENT**, LOG, PERCENT, and SQRT
TWOWAY=**	Two-way layout	CLUSTER, GROUPHORIZONTAL, GROUPVERTICAL*, and STACKED
TYPE=	Type	BARCHART* and DOTPLOT

*Default

**For two-way tables

You can specify the following *plot-options* for all frequency plots: ORIENT=, SCALE=, and TYPE=. You can specify the following *plot-options* for frequency plots of two-way (and multiway) tables: GROUPBY=, NPANELPOS=, and TWOWAY=. The NPANELPOS= *plot-option* is not available with the TWOWAY=CLUSTER or TWOWAY=STACKED layout, which is always displayed in a single panel.

By default, PROC FREQTAB displays frequency plots as bar charts. To display frequency plots as dot plots, specify TYPE=DOTPLOT. To plot percentages instead of frequencies, specify SCALE=PERCENT. For two-way tables, there are four frequency plot layouts available, which you can request by specifying the TWOWAY= *plot-option*. For more information, see the subsection "Plot Options" in this section.

By default, graph cells in a two-way layout are first grouped by column variable levels; row variable levels are then displayed within the column variable levels. To group first by row variable levels, specify GROUPBY=ROW.

KAPPAPLOT < (*plot-options*) >
requests a plot of kappa statistics along with confidence limits. Kappa plots are available for multiway square tables and display the kappa statistic (with confidence limits) for each two-way table (stratum). Kappa plots also display the overall kappa statistic unless you specify the COMMON=NO *plot-option*. To produce a kappa plot, you must specify the AGREE option in the TABLES statement to compute kappa statistics.

Table 11.15 lists the *plot-options* that are available for kappa plots. For descriptions of the *plot-options*, see the subsection "Plot Options" in this section.

Table 11.15 Plot Options for KAPPAPLOT and WTKAPPAPLOT

Plot Option	Description	Values
CLDISPLAY=	Error bar type	BAR, LINE, LINEARROW, SERIF*, and SERIFARROW
COMMON=	Overall kappa	NO and YES*
NPANELPOS=	Statistics per graphic	Number (all*)
ORDER=	Order of two-way levels	ASCENDING and DESCENDING
RANGE=	Range to display	Values and CLIP
STATS	Statistic values	None

*Default

MOSAICPLOT < (*plot-options*) >
requests a mosaic plot. Mosaic plots are available for two-way and multiway crosstabulation tables; for multiway tables, PROC FREQTAB provides a mosaic plot for each two-way table (stratum).

To produce a mosaic plot, you must specify the MOSAICPLOT *plot-request* in the PLOTS= option, or you must specify the PLOTS=ALL option. PROC FREQTAB does not produce mosaic plots by default when ODS Graphics is enabled.

Mosaic plots display tiles that correspond to the crosstabulation table cells. The areas of the tiles are proportional to the frequencies of the table cells. The column variable is displayed on the X axis, and the tile widths are proportional to the relative frequencies of the column variable levels. The row variable is displayed on the Y axis, and the tile heights are proportional to the relative frequencies of the row levels within column levels. For more information, see Friendly (2000).

By default, the colors of the tiles correspond to the row variable levels. If you specify the COLORSTAT= *plot-option*, the tiles are colored according to the values of the Pearson or standardized residuals.

You can specify the following *plot-options*:

COLORSTAT < =PEARSONRES | STDRES >
colors the mosaic plot tiles according to the values of residuals. If you specify COLORSTAT=PEARSONRES, the tiles are colored according to the Pearson residuals of the corresponding table cells. For more information, see the section "Pearson Chi-Square Test for Two-Way Tables" on page 477. If you specify COLORSTAT=STDRES, the tiles are colored according to the standardized residuals of the corresponding table cells. For more information, see the section "Standardized Residuals" on page 478. You can display the Pearson or standardized residuals in the CROSSLIST table by specifying the CROSSLIST(PEARSONRES) or CROSSLIST(STDRES) option, respectively.

SQUARE
produces a square mosaic plot, where the height of the Y axis equals the width of the X axis. In a square mosaic plot, the scale of the relative frequencies is the same on both axes. By default, PROC FREQTAB produces a rectangular mosaic plot.

NONE
suppresses all plots.

ODDSRATIOPLOT < (*plot-options*) >
requests a plot of odds ratios along with confidence limits. Odds ratio plots are available for multiway 2×2 tables and display the odds ratio (with confidence limits) for each 2×2 table (stratum). To produce an odds ratio plot, you must also specify the MEASURES, OR, or RELRISK option in the TABLES statement to compute the odds ratios.

Table 11.16 lists the *plot-options* that are available for odds ratio plots. For descriptions of the *plot-options*, see the subsection "Plot Options" in this section.

Table 11.16 Plot Options for ODDSRATIOPLOT, RELRISKPLOT, and RISKDIFFPLOT

Plot Option	Description	Values
CL=	Confidence limit type	Type
CLDISPLAY=	Error bar type	BAR, LINE, LINEARROW, SERIF*, and SERIFARROW
COLUMN=**	Risk column	1* and 2
COMMON=	Common value	NO and YES*
LOGBASE=***	Axis scale	2, E, and 10
NPANELPOS=	Statistics per graphic	Number (all*)
ORDER=	Order of two-way levels	ASCENDING and DESCENDING
RANGE=	Range to display	Values and CLIP
STATS	Statistic values	None

*Default

**Available for RELRISKPLOT and RISKDIFFPLOT

***Available for ODDSRATIOPLOT and RELRISKPLOT

You can specify one of the following confidence limit types for the odds ratio plot: exact (CL=EXACT), likelihood ratio (CL=LR), exact mid-p (CL=MIDP), score (CL=SCORE), Wald (CL=WALD), or Wald modified (CL=WALDMODIFIED). By default, the odds ratio plot displays

Wald confidence limits. For more information, see the descriptions of the CL= *plot-option* and the OR(CL=) option.

To display exact confidence limits in the odds ratio plot, you must also request their computation by specifying the OR option in the EXACT statement.

When CL=WALD or CL=EXACT, the odds ratio plot displays the common odds ratio by default when it is available. To compute the common odds ratio along with Wald confidence limits, specify the CMH option in the TABLES statement. To compute the common odds ratio along with exact confidence limits, specify the COMOR option in the EXACT statement. To suppress display of the common odds ratio, specify COMMON=NO.

RELRISKPLOT < (*plot-options*) >

requests a plot of relative risks along with confidence limits. Relative risk plots are available for multiway 2×2 tables and display the relative risk (with confidence limits) for each 2×2 table (stratum). To produce a relative risk plot, you must also specify the MEASURES or RELRISK option in the TABLES statement to compute relative risks.

Table 11.16 lists the *plot-options* that are available for relative risk plots. For descriptions of the *plot-options*, see the subsection "Plot Options" in this section.

You can specify one of the following confidence limit types for the relative risk plot: exact (CL=EXACT), likelihood ratio (CL=LR), score (CL=SCORE), Wald (CL=WALD), or Wald modified (CL=WALDMODIFIED). By default, the relative risk plot displays Wald confidence limits. For more information, see the descriptions of the CL= *plot-option* and the RELRISK(CL=) option.

To display exact confidence limits in the relative risk plot, you must also request their computation by specifying the RELRISK option in the EXACT statement. The risk column that you specify for the confidence limits must match the risk column that you specify for the plot.

The relative risk plot displays the common relative risk by default when you specify CL=WALD and the CMH option in the TABLES statement. To suppress display of the common relative risk, specify COMMON=NO.

In addition to the *plot-options* in Table 11.16, you can specify the following *plot-option*:

FOOTNOTE=NO

suppresses the footnote that identifies the column for which the relative risk is computed.

RISKDIFFPLOT < (*plot-options*) >

requests a plot of risk (proportion) differences along with confidence limits for multiway 2×2 tables. The risk difference plot displays the risk difference (with confidence limits) for each 2×2 table (stratum). Optionally, the plot also displays the common risk difference.

To produce a risk difference plot, you must also specify the RISKDIFF option in the TABLES statement to compute risk differences.

Table 11.16 lists the *plot-options* that are available for risk difference plots. For descriptions of the *plot-options*, see the subsection "Plot Options" in this section.

You can specify the confidence limit type for the stratum risk differences by using the CL= *plot-option*. You can specify one of the following confidence limit types: Agresti-Caffo (CL=AC),

exact (CL=EXACT), Hauck-Anderson (CL=HA), Miettinen-Nurminen (score) (CL=MN), Newcombe (CL=NEWCOMBE), and Wald (CL=WALD). By default, the plot displays Wald confidence limits for the stratum risk differences. For more information, see the descriptions of the CL= *plot-option* and the RISKDIFF(CL=) option.

To display exact confidence limits in the risk difference plot, you must also request their computation by specifying the RISKDIFF option in the EXACT statement. The risk column that you specify for the confidence limits must match the risk column that you specify for the plot.

By default, the risk difference plot displays the common risk difference when you specify the RISKDIFF(COMMON) or COMMONRISKDIFF option unless you specify the CL=EXACT *plot-option*. To suppress display of the common risk difference, specify COMMON=NO.

In addition to the *plot-options* in Table 11.16, you can specify the following *plot-options*:

CLNOTE=NO
suppresses the note that identifies the confidence limit type.

COMMON=*type*
specifies the *type* of confidence limits to display for the common risk difference.

The default common confidence limit *type* depends on the stratum confidence limit *type*. By default, COMMON=MN if CL=MN and COMMON=NEWCOMBE if CL=NEWCOMBE. Otherwise, COMMON=MH by default.

You can specify one of the following *types*:

K

KLINGENBERG
displays Klingenberg confidence limits. For more information, see the COMMONRISKDIFF(CL=K) option and the section "Klingenberg Confidence Limits" on page 512.

MH
displays Mantel-Haenszel confidence limits. For more information, see the COMMONRISKDIFF(CL=MH) option and the section "Mantel-Haenszel Confidence Limits and Test" on page 511.

MN
displays Miettinen-Nurminen confidence limits. For more information, see the COMMONRISKDIFF(CL=MN) option.

MNMH
displays confidence limits that are computed by using the Miettinen-Nurminen method with Mantel-Haenszel weights. For more information, see the COMMONRISKDIFF(CL=MNMH) option.

MR

MINRISK
displays minimum risk confidence limits. For more information, see the COMMONRISKDIFF(CL=MR) option and the section "Minimum Risk Confidence Limits and Test" on page 513.

NEWCOMBE
displays stratified Newcombe confidence limits that use Mantel-Haenszel weights to combine the stratum components. For more information, see the COMMONRISKDIFF(CL=NEWCOMBE) option and the section "Stratified Newcombe Confidence Limits" on page 515.

NEWCOMBEMR
displays stratified Newcombe confidence limits that use minimum risk weights to combine the stratum components. For more information, see the COMMONRISKDIFF(CL=NEWCOMBEMR) option and the section "Stratified Newcombe Confidence Limits" on page 515.

NONE
suppresses the common risk difference in the risk difference plot.

SCORE
displays summary score confidence limits. For more information, see the COMMONRISKDIFF(CL=SCORE) option and the section "Summary Score Confidence Limits" on page 514.

FOOTNOTE=NO
suppresses the footnote that identifies the column for which the risk difference is computed.

WTKAPPAPLOT <(*plot-options*)>
requests a plot of weighted kappa coefficients along with confidence limits. Weighted kappa plots are available for multiway square tables and display the weighted kappa coefficient (with confidence limits) for each two-way table (stratum). Weighted kappa plots also display the overall weighted kappa coefficient unless you specify the COMMON=NO *plot-option*.

To produce a weighted kappa plot, you must specify the AGREE option in the TABLES statement to compute weighted kappa coefficients, and the table dimension must be greater than 1.

Table 11.15 lists the *plot-options* that are available for weighted kappa plots. For descriptions of the *plot-options*, see the subsection "Plot Options" in this section.

Global Plot Options

A *global-plot-option* applies to all plots for which the option is available unless it is altered by an individual *plot-option*. You can specify *global-plot-options* in parentheses after the PLOTS option. For example:

```
plots(order=ascending stats)=(riskdiffplot oddsratioplot)
plots(only)=freqplot
```

The following *plot-options* are available as *global-plot-options*: CLDISPLAY=, COLUMN=, COMMON=, EXACT, LOGBASE=, NPANELPOS=, ORDER=, ORIENT=, RANGE=, SCALE=, STATS, and TYPE=. For descriptions of these *plot-options*, see the subsection "Plot Options" in this section.

In addition to these *plot-options*, you can specify the following *global-plot-option*:

ONLY
suppresses the default plots and requests only the plots that are specified as *plot-requests*.

Plot Options

You can specify the following *plot-options* in parentheses after a *plot-request*:

CL=*type*
specifies the *type* of confidence limits to display. You can specify the CL= *plot-option* when you specify any of the following *plot-requests*: ODDSRATIOPLOT, RELRISKPLOT, and RISKDIFFPLOT.

For odds ratio plots (ODDSRATIOPLOT), the available confidence limit types include the following: exact (CL=EXACT), likelihood ratio (CL=LR), exact mid-*p* (CL=MIDP), score (CL=SCORE), Wald (CL=WALD), and Wald modified (CL=WALDMODIFIED). For more information, see the description of the OR(CL=) option and the section "Confidence Limits for the Odds Ratio" on page 516. By default, CL=WALD. When you specify CL=EXACT to display exact confidence limits, you must also request computation of exact confidence limits by specifying the OR option in the EXACT statement.

For relative risk plots (RELRISKPLOT), the available confidence limit types include the following: exact (CL=EXACT), likelihood ratio (CL=LR), score (CL=SCORE), Wald (CL=WALD), and Wald modified (CL=WALDMODIFIED). For more information, see the description of the RELRISK(CL=) option and the section "Confidence Limits for the Relative Risk" on page 519. By default, CL=WALD. When you specify CL=EXACT to display exact confidence limits, you must also request computation of exact confidence limits by specifying the RELRISK option in the EXACT statement.

For risk difference plots (RISKDIFFPLOT), the available confidence limit types include the following: Agresti-Caffo (CL=AC), exact (CL=EXACT), Hauck-Anderson (CL=HA), Miettinen-Nurminen (score) (CL=MN), Newcombe (CL=NEWCOMBE), and Wald (CL=WALD). For more information, see the description of the RISKDIFF(CL=) option and the section "Confidence Limits for the Risk Difference" on page 502. By default, CL=WALD. When you specify CL=EXACT to display exact confidence limits in the plot, you must also request computation of exact confidence limits by specifying the RISKDIFF option in the EXACT statement.

CLDISPLAY=BAR < *width* **> | LINE | LINEARROW | SERIF | SERIFARROW**
controls the appearance of the confidence limit error bars. You can specify the CLDISPLAY= *plot-option* when you specify the following *plot-requests*: KAPPAPLOT, ODDSRATIOPLOT, RELRISKPLOT, RISKDIFFPLOT, and WTKAPPAPLOT.

By default, CLDISPLAY=SERIF, which displays the confidence limits as lines with serifs. CLDISPLAY=LINE displays the confidence limits as plain lines without serifs. The CLDISPLAY=SERIFARROW and CLDISPLAY=LINEARROW *plot-options* display arrowheads on any error bars that are clipped by the RANGE= *plot-option*; if an entire error bar is cut from the plot, the plot displays an arrowhead that points toward the statistic.

CLDISPLAY=BAR displays the confidence limits as bars. By default, the width of the bars equals the size of the marker for the estimate. You can control the width of the bars and the size of the marker by specifying the value of *width* as a percentage of the distance between bars, $0 < \textit{width} \leq 1$. The bar might disappear when the value of *width* is very small.

COLUMN=1 | 2

specifies the table column for which to compute the risks (proportions) for the relative risk plot (RELRISKPLOT) and the risk difference plot (RISKDIFFPLOT). If you specify COLUMN=1, the plot displays the column 1 relative risks or risk differences; if you specify COLUMN=2, the plot displays the column 2 relative risks or risk differences.

If you omit this option for the relative risk plot, the plot uses the table column that you specify for the relative risk statistics in the RELRISK(COLUMN=) option. If you omit both of these COLUMN= options, then by default, COLUMN=1 for the relative risk plot.

If you omit the COLUMN= option for the risk difference plot, the plot uses the table column that you specify for the risk difference statistics in the RISKDIFF(COLUMN=) option. If you omit both of these COLUMN= options, then by default, COLUMN=1 for the risk difference plot.

COMMON=NO | YES

controls the display of the common (overall) statistic in plots that display stratum (two-way table) statistics for multiway tables. You can specify the COMMON= *plot-option* when you specify the following *plot-requests*: KAPPAPLOT, ODDSRATIOPLOT, RELRISKPLOT, RISKDIFFPLOT, and WTKAPPAPLOT.

COMMON=NO suppresses display of the common statistic and its confidence limits. By default, COMMON=YES, which displays the common statistic and its confidence limits when these values are available. For more information, see the descriptions of the *plot-requests*.

EXACT

requests display of exact confidence limits instead of asymptotic confidence limits. You can specify the EXACT *plot-option* when you specify the following *plot-requests*: ODDSRATIOPLOT, RELRISKPLOT, and RISKDIFFPLOT. The EXACT *plot-option* is equivalent to the CL=EXACT *plot-option*.

When you specify the EXACT *plot-option*, you must also request computation of exact confidence limits by specifying the appropriate *statistic-option* in the EXACT statement.

GROUPBY=COLUMN | ROW

specifies the primary grouping for two-way frequency plots, which you can request by specifying the FREQPLOT *plot=request*. By default, GROUPBY=COLUMN, which groups graph cells first by column variable and displays row variable levels within column variable levels. You can specify GROUPBY=ROW to group first by row variable. In two-way and multiway table requests, the column variable is the last variable specified and forms the columns of the crosstabulation table. The row variable is the next-to-last variable specified and forms the rows of the table.

By default for a bar chart that is displayed in the TWOWAY=STACKED layout, bars correspond to the column variable levels, and row levels are displayed (stacked) within each column bar. By default for a bar chart that is displayed in the TWOWAY=CLUSTER layout, bars are first grouped by column variable levels, and row levels are displayed as adjacent bars within each column-level group. You can reverse the default row and column variable grouping by specifying GROUPBY=ROW.

LOGBASE=2 | E | 10
applies to the odds ratio plot (ODDSRATIOPLOT) and the relative risk plot (RELRISKPLOT). This *plot-option* displays the odds ratio or relative risk axis on the log scale that you specify.

LEGEND=NO | YES
applies to the agreement plot (AGREEPLOT). LEGEND=NO suppresses the legend that identifies the areas of exact and partial agreement. By default, LEGEND=YES.

NOSTAT
applies to the deviation plot (DEVIATIONPLOT). NOSTAT suppresses the chi-square *p*-value that deviation plot displays by default.

NPANELPOS=*n*
divides the plot into multiple panels that display at most |*n*| statistics or sections.

If *n* is positive, the number of statistics or sections per panel is balanced; if *n* is negative, the number of statistics per panel is not balanced. For example, suppose you want to display 21 odds ratios. NPANELPOS=20 displays two panels, the first with 11 odds ratios and the second with 10 odds ratios; NPANELPOS=–20 displays 20 odds ratios in the first panel but only 1 odds ratio in the second panel. This *plot-option* is available for all plots except mosaic plots and one-way weighted frequency plots.

For two-way frequency plots (FREQPLOT), NPANELPOS=*n* requests that panels display at most |*n*| sections, where sections correspond to row or column variable levels, depending on the type of plot and the grouping. By default, *n*=4 and each panel includes at most four sections. This *plot-option* applies to two-way plots that are displayed in the TWOWAY=GROUPVERTICAL or TWOWAY=GROUPHORIZONTAL layout. The NPANELPOS= *plot-option* does not apply to the TWOWAY=CLUSTER and TWOWAY=STACKED layouts, which are always displayed in a single panel.

For plots that display statistics along with confidence limits, NPANELPOS=*n* requests that panels display at most |*n*| statistics. By default, *n*=0 and all statistics are displayed in a single panel. This *plot-option* applies to the following plots: KAPPAPLOT, ODDSRATIOPLOT, RELRISKPLOT, RISKDIFFPLOT, and WTKAPPAPLOT.

ORDER=ASCENDING | DESCENDING
displays the two-way table (strata) statistics in order of the statistic value. You can specify the ORDER= *plot-option* when you specify the following *plot-requests*: KAPPAPLOT, ODDSRATIOPLOT, RELRISKPLOT, RISKDIFFPLOT, and WTKAPPAPLOT.

If you specify ORDER=ASCENDING or ORDER=DESCENDING, the plot displays the statistics in ascending or descending order, respectively. By default, the order of the statistics in the plot matches the order that the two-way table strata appear in the multiway table display.

ORIENT=HORIZONTAL | VERTICAL
controls the orientation of the plot. You can specify the ORIENT= *plot-option* when you specify the following *plot-requests*: CUMFREQPLOT, DEVIATIONPLOT, and FREQPLOT.

ORIENT=HORIZONTAL places the variable levels on the Y axis and the frequencies, percentages, or statistic values on the X axis. ORIENT=VERTICAL places the variable levels on the X axis. The default orientation is ORIENT=VERTICAL for bar charts (TYPE=BARCHART) and ORIENT=HORIZONTAL for dot plots (TYPE=DOTPLOT).

PARTIAL=NO | YES

controls the display of partial agreement in the agreement plot (AGREEPLOT). PARTIAL=NO suppresses the display of partial agreement. When you specify PARTIAL=NO, the agreement plot displays only exact agreement. Exact agreement includes the diagonal cells of the square table, where the row and column variable levels are the same. Partial agreement includes the adjacent off-diagonal table cells, where the row and column values are within one level of exact agreement. By default, PARTIAL=YES.

RANGE=(< *min*> < **,** *max*> **)| CLIP**

specifies the range of values to display. You can specify the RANGE= *plot-option* when you specify the following *plot-requests*: KAPPAPLOT, ODDSRATIOPLOT, RELRISKPLOT, RISKDIFFPLOT, and WTKAPPAPLOT.

If you specify RANGE=CLIP, the confidence limits are clipped and the display range is determined by the minimum and maximum values of the statistics. By default, the display range includes all confidence limits.

SCALE=FREQ | GROUPPERCENT | LOG | PERCENT | SQRT

specifies the scale of the frequencies to display. This *plot-option* is available for frequency plots (FREQPLOT) and cumulative frequency plots (CUMFREQPLOT).

By default, SCALE=FREQ, which displays unscaled frequencies. SCALE=PERCENT displays percentages (relative frequencies) of the total frequency. SCALE=LOG displays log (base 10) frequencies. SCALE=SQRT displays square roots of the frequencies, producing a plot known as a *rootogram*.

SCALE=GROUPPERCENT is available for two-way frequency plots. This option displays the row or column percentages instead of the overall percentages (of the table frequency). By default (or when you specify the GROUPBY=COLUMN *plot-option*), SCALE=GROUPPERCENT displays the column percentages. If you specify the GROUPBY=ROW *plot-option*, the primary grouping of graph cells is by row variable level and the plot displays row percentages. For more information, see the description of the GROUPBY= *plot-option*.

SHOWSCALE=NO | YES

controls the display of the cumulative frequency scale on the right side of the agreement plot (AGREEPLOT). SHOWSCALE=NO suppresses the display of the scale. By default, SHOWSCALE=YES.

STATS

displays statistic values in the plot. For the following *plot-requests*, the STATS *plot-option* displays the statistics and their confidence limits on the right side of the plot: KAPPAPLOT, ODDSRATIOPLOT, RELRISKPLOT, RISKDIFFPLOT, and WTKAPPAPLOT.

For the agreement plot (AGREEPLOT), the STATS *plot-option* displays the values of the kappa statistic, the weighted kappa statistic, the B_n measure (Bangdiwala and Bryan 1987), and the sample size. PROC FREQTAB stores these statistics in an ODS table named BnMeasure, which is not displayed. For more information, see the section "ODS Table Names" on page 565.

If you do not request the STATS *plot-option*, these plots do not display the statistic values.

TWOWAY=CLUSTER | GROUPHORIZONTAL | GROUPVERTICAL | STACKED

specifies the layout for two-way frequency plots.

All TWOWAY= layouts are available for bar charts (TYPE=BARCHART). All TWOWAY= layouts except TWOWAY=CLUSTER are available for dot plots (TYPE=DOTPLOT). The ORIENT= and GROUPBY= *plot-options* are available for all TWOWAY= layouts.

The default two-way layout is TWOWAY=GROUPVERTICAL, which produces a grouped plot that has a vertical common baseline. By default for bar charts (TYPE=BARCHART, ORIENT=VERTICAL), the X axis displays column variable levels, and the Y axis displays frequencies. The plot includes a vertical (Y-axis) block for each row variable level. The relative positions of the graph cells in this plot layout are the same as the relative positions of the table cells in the crosstabulation table. You can reverse the default row and column grouping by specifying the GROUPBY=ROW *plot-option*.

The TWOWAY=GROUPHORIZONTAL layout produces a grouped plot that has a horizontal common baseline. By default (GROUPBY=COLUMN), the plot displays a block on the X axis for each column variable level. Within each column-level block, the plot displays row variable levels.

The TWOWAY=STACKED layout produces stacked displays of frequencies. By default (GROUPBY=COLUMN) in a stacked bar chart, the bars correspond to column variable levels, and row levels are stacked within each column level. By default in a stacked dot plot, the dotted lines correspond to column levels, and cell frequencies are plotted as data dots on the corresponding column line. The dot color identifies the row level.

The TWOWAY=CLUSTER layout, which is available only for bar charts, displays groups of adjacent bars. By default, the primary grouping is by column variable level, and row levels are displayed within each column level.

You can reverse the default row and column grouping in any layout by specifying the GROUPBY=ROW *plot-option*. By default, GROUPBY=COLUMN, which groups first by column variable.

TYPE=BARCHART | DOTPLOT

specifies the plot type (format) of the frequency (FREQPLOT), cumulative frequency (CUMFREQPLOT), and deviation plots (DEVIATIONPLOT). TYPE=BARCHART produces a bar chart and TYPE=DOTPLOT produces a dot plot. By default, TYPE=BARCHART.

PRINTKWTS

displays the agreement weights that PROC FREQTAB uses to compute the weighted kappa coefficient. Agreement weights reflect the relative agreement between pairs of variable levels. By default, PROC FREQTAB uses the Cicchetti-Allison form of agreement weights. If you specify the AGREE(WT=FC) option, the procedure uses the Fleiss-Cohen form of agreement weights. For more information, see the section "Weighted Kappa Coefficient" on page 532.

This option has no effect unless you also specify the AGREE option to compute the weighted kappa coefficient. The PRINTKWTS option is equivalent to the AGREE(PRINTKWTS) option.

RELRISK <(*relrisk-options*)>

requests relative risk measures and their confidence limits for 2 × 2 tables. These measures include the odds ratio, the column 1 relative risk, and the column 2 relative risk. For more information, see the section "Odds Ratio and Relative Risks" on page 516. By default, PROC FREQTAB displays the relative risk measures and their asymptotic Wald confidence limits in the "Odds Ratio and Relative Risks" table. You can also obtain this table by specifying the MEASURES option, which produces other measures of association in addition to the relative risks.

You can specify *relrisk-options* in parentheses after the RELRISK option to request tests and additional confidence limits for the column 1 or column 2 relative risk. Table 11.17 summarizes the *relrisk-options*.

When you request tests or additional confidence limit types for the relative risk, PROC FREQTAB does not display the "Odds Ratio and Relative Risks" table unless you also specify the PRINTALL *relrisk-option*.

Table 11.17 RELRISK (Relative Risk) Options

Option	Description
COLUMN=1 | 2	Specifies the risk column
PRINTALL	Displays "Odds Ratio and Relative Risks" table
Request Confidence Limits	
CL=EXACT	Displays exact confidence limits
CL=LR	Requests likelihood ratio confidence limits
CL=SCORE	Requests score confidence limits
CL=WALD	Requests Wald confidence limits
CL=WALDMODIFIED	Requests Wald modified confidence limits
Request Tests	
EQUAL(NULL=)	Requests an equality test
EQUIV | EQUIVALENCE	Requests an equivalence test
MARGIN=	Specifies the test margin
METHOD=	Specifies the test method
NONINF | NONINFERIORITY	Requests a noninferiority test
SUP | SUPERIORITY	Requests a superiority test

You can specify the following *relrisk-options*:

CL=*type* | (*types*)

specifies confidence limit types for the relative risk. You can specify one or more *types* of confidence limits. When you specify only one *type*, you can omit the parentheses around the request. When you specify the CL= *relrisk-option*, PROC FREQTAB displays the confidence limits in the "Confidence Limits for the Relative Risk" table.

The ALPHA= option determines the level of the confidence limits that the CL= *relrisk-option* provides. By default, ALPHA=0.05, which produces 95% confidence limits for the relative risk.

You can specify the following *types*:

EXACT
displays exact unconditional confidence limits for the relative risk in the "Confidence Limits for the Relative Risk" table. (By default, PROC FREQTAB displays the exact confidence limits in a separate table.) You must also request computation of the exact confidence limits by specifying the RELRISK option in the EXACT statement. For more information, see the subsection "Exact Unconditional Confidence Limits" in the section "Confidence Limits for the Relative Risk" on page 519.

LR
LIKELIHOOD RATIO
requests likelihood ratio confidence limits for the relative risk. For more information, see the subsection "Likelihood Ratio Confidence Limits" in the section "Confidence Limits for the Relative Risk" on page 519.

SCORE < (CORRECT=NO) >
requests score confidence limits for the relative risk. For more information, see the subsection "Score Confidence Limits" in the section "Confidence Limits for the Relative Risk" on page 519. If you specify CORRECT=NO, PROC FREQTAB provides the uncorrected form of the confidence limits.

WALD
requests asymptotic Wald confidence limits, which are based on a log transformation of the relative risk. For more information, see the subsection "Wald Confidence Limits" in the section "Confidence Limits for the Relative Risk" on page 519.

WALDMODIFIED
requests Wald modified confidence limits for the relative risk. For more information, see the subsection "Wald Modified Confidence Limits" in the section "Confidence Limits for the Relative Risk" on page 519.

COLUMN=1 | 2
specifies the table column for which to compute the relative risk confidence limits (which you request by specifying the CL= *relrisk-option*) and the relative risk tests (EQUAL, EQUIV, NONINF, and SUP).

This option has no effect on the "Odds Ratio and Relative Risks" table, which displays both column 1 and column 2 relative risks.

By default, COLUMN=1 unless you specify the column for the relative risk plot by using the RELRISKPLOT(COLUMN=) option. When you specify the column for the plot (and omit the RELRISK(COLUMN=) option), PROC FREQTAB uses the same column for both the plot and the relative risk statistics.

EQUAL < (NULL=*value*)>
requests an equality test for the relative risk. For more information, see the subsection "Equality Test" in the section "Relative Risk Tests" on page 522.

You can specify the null hypothesis *value* of the relative risk in the NULL= option. The null *value* must be a positive number. By default, NULL=1. You can specify the equality test method in the METHOD= *relrisk-option*. By default, PROC FREQTAB produces a Wald test (METHOD=WALD).

EQUIV

EQUIVALENCE

requests an equivalence test for the relative risk. For more information, see the subsection "Equivalence Test" in the section "Relative Risk Tests" on page 522. You can specify the test method in the METHOD= *relrisk-option*, and you can specify the test margins in the MARGIN= *relrisk-option*. By default, METHOD=WALD and MARGIN=(0.8,1.25).

MARGIN=*value* | **(***lower***,** *upper***)**

specifies the margin for the noninferiority, superiority, and equivalence tests, which you request by specifying the NONINF, SUP, and EQUIV *relrisk-options*, respectively. By default, MARGIN=0.8 for noninferiority tests, MARGIN=1.25 for superiority tests, and MARGIN=(0.8,1.25) for equivalence tests.

For noninferiority and superiority tests, specify a single *value* in the MARGIN= option. The *value* must be a positive number. For a noninferiority test, the *value* should be less than 1; for a superiority test, the *value* should be greater than 1.

For an equivalence test, you can specify a single MARGIN= *value*, or you can specify both *lower* and *upper* values. All values must be positive numbers. If you specify a single *value*, PROC FREQTAB uses *value* as the lower margin and the inverse of *value* as the upper margin. If you specify both *lower* and *upper* values, the value of *lower* must be less than the value of *upper*.

METHOD=*method*

specifies the method to be used for the equality, equivalence, noninferiority, and superiority tests, which you request by specifying the EQUAL, EQUIV, NONINF, and SUP *relrisk-options*, respectively. By default, METHOD=WALD.

You can specify one of the following *methods*:

FM

SCORE

requests Farrington-Manning (score) tests for the equality, equivalence, noninferiority, and superiority analyses of the relative risk. For more information, see the subsection "Farrington-Manning (Score) Test" in the section "Relative Risk Tests" on page 522.

LR

LIKELIHOODRATIO

requests likelihood ratio tests for the equality, equivalence, noninferiority, and superiority analyses of the relative risk. For more information, see the subsection "Likelihood Ratio Test" in the section "Relative Risk Tests" on page 522.

WALD

requests Wald tests for the equality, equivalence, noninferiority, and superiority analyses of the relative risk. For more information, see the subsection "Wald Test" in the section "Relative Risk Tests" on page 522.

WALDMODIFIED
requests Wald modified tests for the equality, equivalence, noninferiority, and superiority analyses of the relative risk. For more information, see the subsection "Wald Modified Test" in the section "Relative Risk Tests" on page 522.

NONINF
NONINFERIORITY
requests a noninferiority test for the relative risk. For more information, see the subsection "Noninferiority Test" in the section "Relative Risk Tests" on page 522. You can specify the test method in the METHOD= *relrisk-option*, and you can specify the margin in the MARGIN= *relrisk-option*. By default, METHOD=WALD and MARGIN=0.8.

PRINTALL
displays the "Odds Ratio and Relative Risks" table when you request tests or additional confidence limits by specifying *relrisk-options*. By default, PROC FREQTAB does not display this table when you request tests or additional confidence limits for the relative risk.

SUP
SUPERIORITY
requests a superiority test for the relative risk. For more information, see the subsection "Superiority Test" in the section "Relative Risk Tests" on page 522. You can specify the test method in the METHOD= *relrisk-option*, and you can specify the margin in the MARGIN= *relrisk-option*. By default, METHOD=WALD and MARGIN=1.25.

RISKDIFF < (*riskdiff-options*) >
requests risks (binomial proportions) and risk differences for 2 × 2 tables. By default, this option provides the row 1 risk, row 2 risk, total (overall) risk, and risk difference (row 1 – row 2), together with their asymptotic standard errors and Wald confidence limits; by default, this option also provides exact (Clopper-Pearson) confidence limits for the row 1, row 2, and total risks. You can request exact unconditional confidence limits for the risk difference by specifying the RISKDIFF option in the EXACT statement. PROC FREQTAB displays these results in the column 1 and column 2 "Risk Estimates" tables (which you can suppress by specifying the NORISKS *riskdiff-option*).

You can specify *riskdiff-options* in parentheses after the RISKDIFF option to request tests and additional confidence limits for the risk difference, in addition to estimates of the common risk difference for multiway 2 × 2 tables. Table 11.18 summarizes the *riskdiff-options*.

The CL= *riskdiff-option* requests confidence limits for the risk difference. Available confidence limit types include Agresti-Caffo, exact unconditional, Hauck-Anderson, Miettinen-Nurminen (score), Newcombe, and Wald. Continuity-corrected Newcombe and Wald confidence limits are also available. You can request more than one type of confidence limits in the same analysis. PROC FREQTAB displays the confidence limits in the "Confidence Limits for the Risk Difference" table.

The CL=EXACT *riskdiff-option* displays exact unconditional confidence limits in the "Confidence Limits for the Risk Difference" table. When you specify CL=EXACT, you must also request computation of the exact confidence limits by specifying the RISKDIFF option in the EXACT statement.

The EQUAL, EQUIV, NONINF, and SUP *riskdiff-options* request tests of equality, equivalence, noninferiority, and superiority, respectively, for the risk difference. Available test methods include Farrington-Manning (score), Hauck-Anderson, and Wald. Newcombe (hybrid-score) confidence limits are available for the equivalence, noninferiority, and superiority analyses.

As part of the noninferiority, superiority, and equivalence analyses, PROC FREQTAB provides null-based equivalence limits that have a confidence coefficient of $100(1-2\alpha)\%$ (Schuirmann 1999). The ALPHA= option determines the confidence level; by default, ALPHA=0.05, which produces 90% equivalence limits for these analyses. For more information, see the sections "Noninferiority Tests" on page 507 and "Equivalence Test" on page 510.

Table 11.18 RISKDIFF (Proportion Difference) Options

Option	Description
COLUMN=1 | 2	Specifies the risk column
COMMON	Requests common risk difference
CORRECT	Requests continuity correction
NORISKS	Suppresses default risk tables
Request Confidence Limits	
CL=AC	Requests Agresti-Caffo confidence limits
CL=EXACT	Displays exact confidence limits
CL=HA	Requests Hauck-Anderson confidence limits
CL=MN | SCORE	Requests Miettinen-Nurminen confidence limits
CL=NEWCOMBE	Requests Newcombe confidence limits
CL=WALD	Requests Wald confidence limits
Request Tests	
EQUAL(NULL=)	Requests an equality test
EQUIV | EQUIVALENCE	Requests an equivalence test
MARGIN=	Specifies the test margin
METHOD=	Specifies the test method
NONINF | NONINFERIORITY	Requests a noninferiority test
SUP | SUPERIORITY	Requests a superiority test
VAR=SAMPLE | NULL	Specifies the test variance

You can specify the following *riskdiff-options*:

CL=*type* **|** **(***types***)**
requests confidence limits for the risk difference. You can specify one or more *types* of confidence limits. When you specify only one *type*, you can omit the parentheses around the request. PROC FREQTAB displays the confidence limits in the "Confidence Limits for the Risk Difference" table.

The ALPHA= option determines the level of the confidence limits. By default, ALPHA=0.05, which produces 95% confidence limits for the risk difference.

You can specify the CL= *riskdiff-option* with or without requests for risk difference tests. The confidence limits that CL= produces do not depend on the tests that you request and do not use the value of the test margin (which you can specify in the MARGIN= *riskdiff-option*).

You can specify the following *types*:

AC

AGRESTICAFFO

requests Agresti-Caffo confidence limits for the risk difference. For more information, see the subsection "Agresti-Caffo Confidence Limits" in the section "Confidence Limits for the Risk Difference" on page 502.

EXACT

displays exact unconditional confidence limits for the risk difference in the "Confidence Limits for the Risk Difference" table. You must also request computation of the exact confidence limits by specifying the RISKDIFF option in the EXACT statement.

By default, PROC FREQTAB computes the exact confidence limits by inverting two separate one-sided exact tests that are based on the score statistic. For more information, see the RISKDIFF option in the EXACT statement and the subsection "Exact Unconditional Confidence Limits" in the section "Confidence Limits for the Risk Difference" on page 502.

By default, PROC FREQTAB also displays these exact confidence limits in the "Risk Estimates" table. You can suppress this table by specifying the NORISKS *riskdiff-option*.

HA

requests Hauck-Anderson confidence limits for the risk difference. For more information, see the subsection "Hauck-Anderson Confidence Limits" in the section "Confidence Limits for the Risk Difference" on page 502.

MN < (CORRECT=NO | MEE) >

SCORE < (CORRECT=NO | MEE) >

requests Miettinen-Nurminen (score) confidence limits for the risk difference. For more information, see the subsection "Miettinen-Nurminen (Score) Confidence Limits" in the section "Confidence Limits for the Risk Difference" on page 502. By default, the Miettinen-Nurminen confidence limits include a bias correction factor (Miettinen and Nurminen 1985; Newcombe and Nurminen 2011). If you specify CL=MN(CORRECT=NO), PROC FREQTAB provides the uncorrected form of the confidence limits (Mee 1984).

NEWCOMBE < (CORRECT) >

requests Newcombe hybrid-score confidence limits for the risk difference. If you specify CL=NEWCOMBE(CORRECT) or the CORRECT *riskdiff-option*, the Newcombe confidence limits include a continuity correction. For more information, see the subsection "Newcombe Confidence Limits" in the section "Confidence Limits for the Risk Difference" on page 502.

WALD < (CORRECT) >

requests Wald confidence limits for the risk difference. If you specify CL=WALD(CORRECT) or the CORRECT *riskdiff-option*, the Wald confidence limits include a continuity correction. For more information, see the subsection "Wald Confidence Limits" in the section "Confidence Limits for the Risk Difference" on page 502.

COLUMN=1 | 2 | BOTH

specifies the table column for which to compute the risk difference confidence limits (which you request by specifying the CL= *riskdiff-option*), the risk difference tests (EQUAL, EQUIV, NONINF, and SUP), and the common risk difference statistics (COMMON). COLUMN=BOTH does not apply to the common risk difference statistics.

By default, COLUMN=1 unless you specify the column for the risk difference plot by using the RISKDIFFPLOT(COLUMN=) option. When you specify the column for the plot (and omit the RISKDIFF(COLUMN=) option), PROC FREQTAB uses the same column for both the plot and the risk difference statistics.

This option has no effect on the "Risk Estimates" table, which is produced for both column 1 and column 2. You can suppress the "Risk Estimates" table by specifying the NORISKS *riskdiff-option*.

COMMON

requests estimates of the common (overall) risk difference for multiway 2×2 tables. This option provides Mantel-Haenszel and summary score estimates for the common risk difference, together with their confidence limits. If you specify the RISKDIFF(CL=NEWCOMBE) option, the RISKDIFF(COMMON) option also provides Newcombe confidence limits for the common risk difference. For more information, see the section "Common Risk Difference" on page 511.

You can use the COMMONRISKDIFF option to request additional confidence limits and tests for the common risk difference.

If you do not specify the COLUMN= *riskdiff-option*, PROC FREQTAB provides the common risk difference for column 1 by default. If you specify COLUMN=2, PROC FREQTAB provides the common risk difference for column 2. COLUMN=BOTH does not apply to the common risk difference.

CORRECT

includes a continuity correction in the Wald confidence limits, Wald tests, and Newcombe confidence limits. For more information, see the section "Risks and Risk Differences" on page 501.

EQUAL < (NULL=*value*)>

requests an equality test for the risk difference. For more information, see the section "Equality Tests" on page 507.

You can specify the null hypothesis *value* of the risk difference in the NULL= option. By default, NULL=0. You can specify the null *value* in proportion form as a number between –1 and 1, or you can specify the null *value* in percentage form as a number between –100 and 100. When the *value* is between –100 and –1 or between 1 and 100, PROC FREQTAB converts the number to a proportion. PROC FREQTAB treats the values –1 and 1 as percentages.

You can specify the equality test method by using the METHOD= *riskdiff-option*. By default, PROC FREQTAB produces a Wald test (METHOD=WALD). By default, PROC FREQTAB uses the sample variance to compute the Wald test statistic; if you specify the VAR=NULL *riskdiff-option*, PROC FREQTAB uses the null (test-based) variance.

EQUIV

EQUIVALENCE

requests an equivalence test for the risk difference. For more information, see the section "Equivalence Test" on page 510. You can specify the test method in the METHOD= *riskdiff-option*, and you can specify the margins in the MARGIN= *riskdiff-option*. By default, METHOD=WALD and MARGIN=0.2.

MARGIN=*value* | **(***lower***,** *upper***)**

specifies the margin for the noninferiority, superiority, and equivalence tests, which you request by specifying the NONINF, SUP, and EQUIV *riskdiff-options*, respectively. By default, MARGIN=0.2.

For noninferiority and superiority tests, specify a single *value* in the MARGIN= option. The *value* must be a positive number. You can specify *value* as a number between 0 and 1. Or you can specify *value* in percentage form as a number between 1 and 100, and PROC FREQTAB converts that number to a proportion. PROC FREQTAB treats the value 1 as 1%.

For an equivalence test, you can specify a single MARGIN= *value*, or you can specify both *lower* and *upper* values. If you specify a single *value*, it must be a positive number, as described previously. If you specify a single *value* for an equivalence test, PROC FREQTAB uses –*value* as the lower margin and *value* as the upper margin for the test. If you specify both *lower* and *upper* values for an equivalence test, you can specify them in proportion form as numbers between –1 and 1. Or you can specify them in percentage form as numbers between –100 and 100, and PROC FREQTAB converts the numbers to proportions. The value of *lower* must be less than the value of *upper*.

METHOD=*method*

specifies the method to be used for the equality, equivalence, noninferiority, and superiority tests, which you request by specifying the EQUAL, EQUIV, NONINF, and SUP *riskdiff-options*, respectively. By default, METHOD=WALD.

You can specify the following *methods*:

FM

SCORE

requests Farrington-Manning (score) tests for the equality, equivalence, noninferiority, and superiority analyses. For more information, see the subsection "Farrington-Manning (Score) Test" in the section "Noninferiority Tests" on page 507.

HA

requests Hauck-Anderson tests for the equality, equivalence, noninferiority, and superiority analyses. For more information, see the subsection "Hauck-Anderson Test" in the section "Noninferiority Tests" on page 507.

NEWCOMBE

requests Newcombe (hybrid-score) confidence limits for the equivalence, noninferiority, and superiority analyses. If you specify the CORRECT *riskdiff-option*, the Newcombe confidence limits include a continuity correction. For more information, see the subsection "Newcombe Noninferiority Analysis" in the section "Noninferiority Tests" on page 507.

WALD
requests Wald tests for the equality, equivalence, noninferiority, and superiority analyses. By default, PROC FREQTAB uses the sample variance to compute these Wald test statistics; if you specify the VAR=NULL *riskdiff-option*, PROC FREQTAB uses the null (test-based) variance. If you specify the CORRECT *riskdiff-option*, the Wald tests and confidence limits include a continuity correction. For more information, see the subsection "Wald Test" in the section "Noninferiority Tests" on page 507.

NONINF

NONINFERIORITY
requests a noninferiority test for the risk difference. For more information, see the section "Noninferiority Tests" on page 507. You can specify the test method in the METHOD= *riskdiff-option*, and you can specify the margin in the MARGIN= *riskdiff-option*. By default, METHOD=WALD and MARGIN=0.2.

NORISKS
suppresses display of the "Risk Estimates" tables, which the RISKDIFF option produces by default for column 1 and column 2. The "Risk Estimates" tables contain the risks and risk differences, together with their asymptotic standard errors, Wald confidence limits, and exact confidence limits.

SUP

SUPERIORITY
requests a superiority test for the risk difference. For more information, see the section "Superiority Test" on page 509. You can specify the test method in the METHOD= *riskdiff-option*, and you can specify the margin in the MARGIN= *riskdiff-option*. By default, METHOD=WALD and MARGIN=0.2.

VAR=NULL | SAMPLE
specifies the type of variance to use in the Wald tests of equality, equivalence, noninferiority, and superiority. If you specify VAR=SAMPLE, PROC FREQTAB uses the sample variance. If you specify VAR=NULL, PROC FREQTAB uses a test-based variance that is computed by using the null hypothesis value of the risk difference. For more information, see the sections "Equality Tests" on page 507 and "Noninferiority Tests" on page 507. By default, VAR=SAMPLE.

SCORES=*type*
specifies the type of row and column scores that PROC FREQTAB uses to compute the following statistics: Mantel-Haenszel chi-square, Pearson correlation, Cochran-Armitage test for trend, weighted kappa coefficient, and Cochran-Mantel-Haenszel statistics. The value of *type* can be one of the following:

- **MODRIDIT**
- **RANK**
- **RIDIT**
- **TABLE**

See the section "Scores" on page 475 for descriptions of these score types.

If you do not specify the SCORES= option, PROC FREQTAB uses SCORES=TABLE by default. For character variables, the row and column TABLE scores are the row and column numbers. That is, the TABLE score is 1 for row 1, 2 for row 2, and so on. For numeric variables, the row and column TABLE scores equal the variable values. For more information, see the section "Scores" on page 475. Using MODRIDIT, RANK, or RIDIT scores yields nonparametric analyses.

You can use the SCOROUT option to display the row and column scores.

SCOROUT

displays the row and column scores that PROC FREQTAB uses to compute score-based tests and statistics. You can specify the score type by using the SCORES= option. For more information, see the section "Scores" on page 475.

The scores are computed and displayed only when PROC FREQTAB computes statistics for two-way tables. You can use ODS to store the scores in an output data set. See the section "ODS Table Names" on page 565 for more information.

SENSPEC <(*options*)>

requests estimates of sensitivity, specificity, accuracy, positive predictive value, and negative predictive value for 2×2 tables. The "Sensitivity and Specificity" table provides these estimates together with their standard errors and confidence limits. For more information, see the section "Sensitivity and Specificity" on page 500.

You can specify the confidence level in the ALPHA= option. By default, ALPHA=0.05, which produces 95% confidence limits.

You can specify the following *options*:

NOTE<=YES | NO>

controls the footnote that identifies the reference level (true positives) in the "Sensitivity and Specificity" table. NOTE=NO suppresses this footnote, and NOTE<=YES> displays it. By default, PROC FREQTAB displays the footnote only when you specify the reference level by using the REFCELL=, REFCOLUMN=, and/or REFROW= options.

REFCELL=*number*

specifies the 2×2 table cell to use as the true-positive classification (positive row variable level and positive column variable level) in the SENSPEC computations. For more information, see the section "Sensitivity and Specificity" on page 500.

You can specify the true-positive cell *number* as 1, 2, 3, or 4; these numbers correspond to the cells of the 2×2 crosstabulation table. By default, REFCELL=1, which uses table cell (1,1) as the true-positive cell (and the corresponding row 1 and column 1 as the positive row and column levels, respectively). REFCELL=2, 3, and 4 use table cells (1,2), (2,1), and (2,2), respectively, as the true-positive cells.

You cannot specify this option together with the REFROW= or REFCOLUMN= option. The true-positive cell *number* determines the corresponding positive row and column levels.

REFCOLUMN=*number* | '*level-value*'
specifies the column level to use as the positive column level in the SENSPEC computations. For more information, see the section "Sensitivity and Specificity" on page 500.

You can specify the positive column level *number* as 1 or 2, which is the order in which the level appears in the 2×2 crosstabulation table. Or you can specify the positive column *level-value*, which is the formatted value of the level. The *level-value* must be enclosed in single quotes. By default, REFCOLUMN=1.

REFROW=*number* | '*level-value*'
specifies the row level to use as the positive row level in the SENSPEC computations. For more information, see the section "Sensitivity and Specificity" on page 500.

You can specify the positive row level *number* as 1 or 2, which is the order in which the level appears in the 2×2 crosstabulation table. Or you can specify the positive row *level-value*, which is the formatted value of the level. The *level-value* must be enclosed in single quotes. By default, REFROW=1.

TOTPCT
displays percentages of the total multiway table frequency in multiway crosstabulation tables (n-way tables where $n > 2$). By default, crosstabulation tables display percentages of the two-way table (stratum) frequency but do not display percentages of the total multiway table frequency. For more information, see the section "Two-Way and Multiway Tables" on page 556.

By default, tables in list format (which you can request by specifying the LIST option) display the percentages of the total multiway table frequency. The variable PERCENT in the OUT= output data set also provides the percentages of the total multiway table frequency.

TREND
requests the Cochran-Armitage test for trend. The table must be $2 \times C$ or $R \times 2$ to compute the trend test. For more information, see the section "Cochran-Armitage Test for Trend" on page 526. To request exact p-values for the trend test, specify the TREND option in the EXACT statement. See the section "Exact Statistics" on page 547 for more information.

TEST Statement

TEST *test-options* **;**

The TEST statement requests asymptotic tests for measures of association and measures of agreement. The *test-options* identify which tests to compute. Table 11.19 lists the available *test-options*, together with their corresponding TABLES statement options. Descriptions of the *test-options* follow the table in alphabetical order.

For each measure of association or agreement that you request in the TEST statement, PROC FREQTAB provides an asymptotic test that the measure is 0. The procedure displays the asymptotic standard error under the null hypothesis, the test statistic, and the one-sided and two-sided p-values. PROC FREQTAB also provides confidence limits for the measure. The ALPHA= option in the TABLES statement determines the confidence level; by default, ALPHA=0.05, which provides 95% confidence limits. For more information, see the sections "Asymptotic Tests" on page 482 and "Confidence Limits" on page 482. For information about

the individual measures, see the sections "Measures of Association" on page 482 and "Tests and Measures of Agreement" on page 529.

You can also request exact tests for selected measures of association and agreement by using the EXACT statement. For more information, see the section "Exact Statistics" on page 547.

Using the TEST Statement with the TABLES Statement

You must use a TABLES statement with the TEST statement. If you use only one TABLES statement, you do not need to specify the same options in both the TABLES and TEST statements; when you specify an option in the TEST statement, PROC FREQTAB automatically invokes the corresponding TABLES statement option. However, when you use the TEST statement with multiple TABLES statements, you must specify options in the TABLES statements to request statistics; PROC FREQTAB then provides asymptotic tests for those statistics that you specify in the TEST statement.

Test Options

Table 11.19 summarizes the *test-options* that you can specify in the TEST statement.

Table 11.19 TEST Statement Options

Test Option	Asymptotic Tests	Required TABLES Statement Option
AGREE	Simple and weighted kappa coefficients	AGREE
GAMMA	Gamma	ALL or MEASURES
KAPPA	Simple kappa coefficient	AGREE
KENTB | TAUB	Kendall's tau-b	ALL or MEASURES
MEASURES	Gamma, Kendall's tau-b, Stuart's tau-c, Somers' $D(C\|R)$, Somers' $D(R\|C)$, Pearson and Spearman correlations	ALL or MEASURES
PCORR	Pearson correlation coefficient	ALL or MEASURES
PLCORR	Polychoric correlation	PLCORR
SCORR	Spearman correlation coefficient	ALL or MEASURES
SMDCR	Somers' $D(C\|R)$	ALL or MEASURES
SMDRC	Somers' $D(R\|C)$	ALL or MEASURES
STUTC | TAUC	Stuart's tau-c	ALL or MEASURES
WTKAPPA | WTKAP	Weighted kappa coefficient	AGREE

You can specify the following *test-options*:

AGREE

requests asymptotic tests for the simple kappa coefficient and the weighted kappa coefficient. For more information, see the sections "Simple Kappa Coefficient" on page 530 and "Weighted Kappa Coefficient" on page 532.

By default, these tests are based on null values of 0; you can specify nonzero null values for the simple kappa and weighted kappa tests by using the AGREE(NULLKAPPA=) and AGREE(NULLWTKAPPA=) options, respectively, in the TABLES statement.

The AGREE option in the TABLES statement provides estimates, standard errors, and confidence limits for kappa coefficients. You can request exact tests for kappa coefficients by using the EXACT statement.

Kappa coefficients are defined only for square tables, where the number of rows equals the number of columns. Kappa coefficients are not computed for tables that are not square. For 2×2 tables, the weighted kappa coefficient is identical to the simple kappa coefficient, and PROC FREQTAB presents only the simple kappa coefficient.

GAMMA

requests an asymptotic test for the gamma statistic. For more information, see the section "Gamma" on page 483. The MEASURES option in the TABLES statement provides the gamma statistic and its asymptotic standard error.

KAPPA

requests an asymptotic test for the simple kappa coefficient. For more information, see the section "Simple Kappa Coefficient" on page 530.

By default, the null value of kappa for this test is 0; you can specify a nonzero null value by using the AGREE(NULLKAPPA=) option in the TABLES statement.

The AGREE option in the TABLES statement provides the kappa statistic, its standard error, and its confidence limits. You can request an exact test for the simple kappa coefficient by specifying the KAPPA option in the EXACT statement.

Kappa coefficients are defined only for square tables, where the number of rows equals the number of columns. PROC FREQTAB does not compute kappa coefficients for tables that are not square.

KENTB

TAUB

requests an asymptotic test for Kendall's tau-*b*. For more information, see the section "Kendall's Tau-b" on page 484.

The MEASURES option in the TABLES statement provides Kendall's tau-*b* and its standard error. You can request an exact test for Kendall's tau-*b* by specifying the KENTB option in the EXACT statement.

MEASURES

requests asymptotic tests for the following measures of association: gamma, Kendall's tau-*b*, Pearson correlation coefficient, Somers' $D(C|R)$, Somers' $D(R|C)$, Spearman correlation coefficient, and Stuart's tau-*c*. For more information, see the section "Measures of Association" on page 482.

The MEASURES option in the TABLES statement provides measures of association and their asymptotic standard errors. You can request exact tests for selected measures by using the EXACT statement.

PCORR

requests an asymptotic test for the Pearson correlation coefficient. For more information, see the section "Pearson Correlation Coefficient" on page 485.

The MEASURES option in the TABLES statement provides the Pearson correlation and its standard error. You can request an exact test for the Pearson correlation by specifying the PCORR option in the EXACT statement.

PLCORR

requests Wald and likelihood ratio tests for the polychoric correlation coefficient. For more information, see the section "Polychoric Correlation" on page 488.

The PLCORR option in the TABLES statement provides the polychoric correlation and its standard error.

SCORR

requests an asymptotic test for the Spearman correlation coefficient. For more information, see the section "Spearman Rank Correlation Coefficient" on page 486.

The MEASURES option in the TABLES statement provides the Spearman correlation and its standard error. You can request an exact test for the Spearman correlation by specifying the SCORR option in the EXACT statement.

SMDCR

requests an asymptotic test for Somers' $D(C|R)$. For more information, see the section "Somers' D" on page 485.

The MEASURES option in the TABLES statement provides Somers' $D(C|R)$ and its standard error. You can request an exact test for Somers' $D(C|R)$ by specifying the SMDCR option in the EXACT statement.

SMDRC

requests an asymptotic test for Somers' $D(R|C)$. For more information, see the section "Somers' D" on page 485.

The MEASURES option in the TABLES statement provides Somers' $D(R|C)$ and its standard error. You can request an exact test for Somers' $D(R|C)$ by specifying the SMDRC option in the EXACT statement.

STUTC

TAUC

requests an asymptotic test for Stuart's tau-c. For more information, see the section "Stuart's Tau-c" on page 484.

The MEASURES option in the TABLES statement provides Stuart's tau-c and its standard error. You can request an exact test for Stuart's tau-c by specifying the STUTC option in the EXACT statement.

WTKAPPA

WTKAP

requests an asymptotic test for the weighted kappa coefficient. For more information, see the section "Weighted Kappa Coefficient" on page 532.

By default, the null value of weighted kappa for this test is 0; you can specify a nonzero null value by using the AGREE(NULLWTKAPPA=) option in the TABLES statement.

The AGREE option in the TABLES statement provides the weighted kappa coefficient, its standard error, and confidence limits. You can request an exact test for the weighted kappa by specifying the WTKAPPA option in the EXACT statement.

Kappa coefficients are defined only for square tables, where the number of rows equals the number of columns. PROC FREQTAB does not compute kappa coefficients for tables that are not square. For

2×2 tables, the weighted kappa coefficient is identical to the simple kappa coefficient, and PROC FREQTAB presents only the simple kappa coefficient.

WEIGHT Statement

WEIGHT *variable* < / *option* > **;**

The WEIGHT statement names a numeric variable that provides a weight for each observation in the input data table. The WEIGHT statement is most commonly used to input cell count data. See the section "Inputting Frequency Counts" on page 472 for more information. If you use a WEIGHT statement, PROC FREQTAB assumes that an observation represents *n* observations, where *n* is the value of *variable*. The value of the WEIGHT variable is not required to be an integer.

If the value of the WEIGHT variable is missing, PROC FREQTAB does not use that observation in the analysis. If the value of the WEIGHT variable is 0, PROC FREQTAB ignores the observation unless you specify the ZEROS option, which includes observations that have weights of 0. If you do not specify a WEIGHT statement, PROC FREQTAB assigns a weight of 1 to each observation. The sum of the WEIGHT variable values represents the total number of observations.

If any value of the WEIGHT variable is negative, PROC FREQTAB displays the frequencies computed from the weighted values but does not compute percentages and statistics. If you create an output data set by using the OUT= option in the TABLES statement, PROC FREQTAB assigns missing values to the PERCENT variable. PROC FREQTAB also assigns missing values to the variables that the OUTEXPECT and OUTPCT options provide. If any value of the WEIGHT variable is negative, you cannot create an output data set by using the OUTPUT statement because statistics are not computed when there are negative weights.

You can specify the following *option*:

ZEROS

includes observations that have weights of 0. By default, PROC FREQTAB ignores observations that have weights of 0.

If you specify the ZEROS option, frequency and crosstabulation tables display levels that contain only zero-weight observations. If you do not specify the ZEROS option, PROC FREQTAB does not process observations that have weights of 0 and therefore does not display levels that contain only zero-weight observations.

When you specify the ZEROS option, PROC FREQTAB includes zero-weight levels in chi-square tests and binomial computations for one-way tables. This makes it possible to compute binomial tests and estimates for a reference level that contains no observations with positive weights.

For two-way tables, the ZEROS option enables computation of kappa statistics when there are levels that contain no observations with positive weights. For more information, see the section "Tables with Zero-Weight Rows or Columns" on page 536.

Even when you specify the ZEROS option, PROC FREQTAB does not compute CHISQ or MEASURES statistics for two-way tables that contain a zero-weight row or column because most of these statistics are undefined in this case.

By default, the ZEROS option includes zero-weight table cells in the LIST table and the OUT= data set. To suppress zero-weight cells, you can specify the NOSPARSE option in the TABLES statement.

Details: FREQTAB Procedure

Inputting Frequency Counts

PROC FREQTAB can use either raw data or cell count data to produce frequency and crosstabulation tables. *Raw data*, also known as case-record data, report the data as one record for each subject or sample member. *Cell count data* report the data as a table, listing all possible combinations of data values along with the frequency counts. This way of presenting data often appears in published results.

The following DATA step statements store raw data in a data table:

```
data mylib.Raw;
   input Subject $ R C @@;
   datalines;
01 1 1  02 1 1  03 1 1  04 1 1  05 1 1
06 1 2  07 1 2  08 1 2  09 2 1  10 2 1
11 2 1  12 2 1  13 2 2  14 2 2  14 2 2
;
```

You can store the same data as cell counts by using the following DATA step statements:

```
data mylib.CellCounts;
   input R C Count @@;
   datalines;
1 1 5   1 2 3
2 1 4   2 2 3
;
```

The variable R contains the values for the rows, and the variable C contains the values for the columns. The variable Count contains the cell count for each row and column combination.

Both the mylib.Raw data table and the mylib.CellCounts data table produce identical frequency counts, two-way tables, and statistics. When you use the mylib.CellCounts data table, you must include a WEIGHT statement to specify that the variable Count contains cell counts. For example, the following PROC FREQTAB statements create a two-way crosstabulation table by using the mylib.CellCounts data table:

```
proc freqtab data=mylib.CellCounts;
   tables R*C;
   weight Count;
run;
```

Missing Values

If an observation has a missing value for the WEIGHT variable, PROC FREQTAB excludes that observation from the analysis.

If an observation has a missing value for a TABLES statement variable, by default PROC FREQTAB does not include that observation in the frequency or crosstabulation table (or in the frequency total, percentages, and statistics). PROC FREQTAB displays the frequency of missing values after the frequency or crosstabulation table.

PROC FREQTAB includes the frequency of missing values in the output data sets. The OUT= data set that you specify in the TABLES statement includes an observation that contains the frequency of missing values. The OUT= data set that you specify in the OUTPUT statement includes the frequency of missing values in the variable NMISS when you specify the NMISS option in the OUTPUT statement.

The following options change the way that PROC FREQTAB handles missing values of TABLES statement variables:

MISSPRINT displays missing value frequencies in frequency tables but does not include them in computing percentages or statistics. This option is available for one-way frequency tables and for tables in list format (which you can request by specifying the LIST option in the TABLES statement).

MISSING treats missing values as a valid nonmissing level for all TABLES statement variables. This option displays missing levels in frequency and crosstabulation tables and includes them in computing percentages and statistics.

This example shows the three ways that PROC FREQTAB can handle missing values of TABLES statement variables. The following DATA step statements create a data table that contains a missing value for the variable A:

```
data mylib.One;
   input A Freq;
   datalines;
1 2
2 2
. 2
;
```

The following PROC FREQTAB statements request a one-way frequency table for the variable A. The first request does not specify a missing value option. The second request specifies the MISSPRINT option in the PROC FREQTAB statement. The third request specifies the MISSING option in the PROC FREQTAB statement.

```
proc freqtab data=mylib.One;
   tables A;
   weight Freq;
   title 'Default';
run;
proc freqtab data=mylib.One missprint;
   tables A;
   weight Freq;
   title 'MISSPRINT Option';
```

```
run;
proc freqtab data=mylib.One missing;
   tables A;
   weight Freq;
   title 'MISSING Option';
run;
```

Figure 11.9 displays the frequency tables produced by this example. The first table shows PROC FREQTAB's default behavior for handling missing values. The frequency table does not include the missing level of the TABLES statement variable A. PROC FREQTAB displays the frequency of missing values after the table.

The second table, for which the MISSPRINT option is specified, displays the missing level of A but does not include its frequency in computing the total frequency and percentages.

The third table, for which the MISSING option is specified, treats the missing level as a valid nonmissing level. The frequency table displays the missing level of A and includes its frequency in computing the total frequency and percentages.

Figure 11.9 Missing Values in Frequency Tables

Default

The FREQTAB Procedure

A	Frequency	Percent	Cumulative Frequency	Cumulative Percent
1	2	50.00	2	50.00
2	2	50.00	4	100.00

Frequency Missing = 2

MISSPRINT Option

The FREQTAB Procedure

A	Frequency	Percent	Cumulative Frequency	Cumulative Percent
.	2	.	.	.
1	2	50.00	2	50.00
2	2	50.00	4	100.00

Frequency Missing = 2

MISSING Option

The FREQTAB Procedure

A	Frequency	Percent	Cumulative Frequency	Cumulative Percent
.	2	33.33	2	33.33
1	2	33.33	4	66.67
2	2	33.33	6	100.00

When a combination of variable levels in a two-way table is missing, PROC FREQTAB assigns 0 to the frequency count of the corresponding table cell. The default crosstabulation table and the CROSSLIST crosstabulation table display table cells that have frequencies of 0.

By default, the LIST crosstabulation table and the OUT= output data set do not include table cells that have frequencies of 0. If you specify the ZEROS option in the WEIGHT statement to include observations that have weights of 0, the LIST table and the OUT= output data set include zero-frequency table cells by default unless you specify the NOSPARSE option in the TABLES statement.

Statistical Computations

Definitions and Notation

A two-way table represents the crosstabulation of row variable X and column variable Y. Let the table row values or levels be denoted by $X_i, i = 1, 2, \ldots, R$, and the column values by $Y_j, j = 1, 2, \ldots, C$. Let n_{ij} denote the frequency of the table cell in the *i*th row and *j*th column and define the following notation:

$$n_{i\cdot} = \sum_j n_{ij} \quad \text{(row totals)}$$

$$n_{\cdot j} = \sum_i n_{ij} \quad \text{(column totals)}$$

$$n = \sum_i \sum_j n_{ij} \quad \text{(overall total)}$$

$$p_{ij} = n_{ij}/n \quad \text{(cell percentages)}$$

$$p_{i\cdot} = n_{i\cdot}/n \quad \text{(row percentages of total)}$$

$$p_{\cdot j} = n_{\cdot j}/n \quad \text{(column percentages of total)}$$

$$R_i = \text{score for row } i$$

$$C_j = \text{score for column } j$$

$$\bar{R} = \sum_i n_{i\cdot} R_i / n \quad \text{(average row score)}$$

$$\bar{C} = \sum_j n_{\cdot j} C_j / n \quad \text{(average column score)}$$

$$A_{ij} = \sum_{k>i} \sum_{l>j} n_{kl} + \sum_{k<i} \sum_{l<j} n_{kl}$$

$$D_{ij} = \sum_{k>i} \sum_{l<j} n_{kl} + \sum_{k<i} \sum_{l>j} n_{kl}$$

$$P = \sum_i \sum_j n_{ij} A_{ij} \quad \text{(twice the number of concordances)}$$

$$Q = \sum_i \sum_j n_{ij} D_{ij} \quad \text{(twice the number of discordances)}$$

Scores

PROC FREQTAB uses scores of the variable values to compute the Mantel-Haenszel chi-square, Pearson correlation, Cochran-Armitage test for trend, weighted kappa coefficient, and Cochran-Mantel-Haenszel statistics. The SCORES= option in the TABLES statement specifies the score type that PROC FREQTAB

uses. The available score types are TABLE, RANK, RIDIT, and MODRIDIT scores. The default score type is TABLE. Using MODRIDIT, RANK, or RIDIT scores yields nonparametric analyses.

For numeric variables, table scores are the values of the row and column levels. If the row or column variable is formatted, then the table score is the internal numeric value corresponding to that level. If two or more numeric values are classified into the same formatted level, then the internal numeric value for that level is the smallest of these values. For character variables, table scores are defined as the row numbers and column numbers (that is, 1 for the first row, 2 for the second row, and so on).

Rank scores, which you request with the SCORES=RANK option, are defined as

$$R_i^1 = \sum_{k<i} n_{k\cdot} + (n_{i\cdot} + 1)/2 \qquad i = 1, 2, \ldots, R$$
$$C_j^1 = \sum_{l<j} n_{\cdot l} + (n_{\cdot j} + 1)/2 \qquad j = 1, 2, \ldots, C$$

where R_i^1 is the rank score of row i, and C_j^1 is the rank score of column j. Note that rank scores yield midranks for tied values.

Ridit scores, which you request with the SCORES=RIDIT option, are defined as rank scores standardized by the sample size (Bross 1958; Mack and Skillings 1980). Ridit scores are derived from the rank scores as

$$R_i^2 = R_i^1/n \qquad i = 1, 2, \ldots, R$$
$$C_j^2 = C_j^1/n \qquad j = 1, 2, \ldots, C$$

Modified ridit scores (SCORES=MODRIDIT) represent the expected values of the order statistics of the uniform distribution on (0,1) (Van Elteren 1960; Lehmann and D'Abrera 2006). Modified ridit scores are derived from rank scores as

$$R_i^3 = R_i^1/(n+1) \qquad i = 1, 2, \ldots, R$$
$$C_j^3 = C_j^1/(n+1) \qquad j = 1, 2, \ldots, C$$

Chi-Square Tests and Statistics

The CHISQ option provides chi-square tests of homogeneity or independence and measures of association that are based on the chi-square statistic. When you specify the CHISQ option in the TABLES statement, PROC FREQTAB computes the following chi-square tests for each two-way table: Pearson chi-square, likelihood ratio chi-square, and Mantel-Haenszel chi-square tests. PROC FREQTAB provides the following measures of association that are based on the Pearson chi-square statistic: phi coefficient, contingency coefficient, and Cramér's V. For 2×2 tables, the CHISQ option also provides Fisher's exact test and the continuity-adjusted chi-square statistic. You can request Fisher's exact test for general $R \times C$ tables by specifying the FISHER option in the TABLES or EXACT statement.

If you specify the CHISQ option for one-way tables, PROC FREQTAB provides a one-way Pearson chi-square goodness-of-fit test. If you specify the CHISQ(LRCHI) option for one-way tables, PROC FREQTAB also provides a one-way likelihood ratio chi-square test. The other tests and statistics that the CHISQ option produces are available only for two-way tables.

For two-way tables, the null hypothesis for the chi-square tests is no association between the row variable and the column variable. When the sample size n is large, the test statistics have asymptotic chi-square

distributions under the null hypothesis. When the sample size is not large, or when the data set is sparse or heavily tied, exact tests might be more appropriate than asymptotic tests. PROC FREQTAB provides exact *p*-values for the Pearson chi-square, likelihood ratio chi-square, and Mantel-Haenszel chi-square tests, in addition to Fisher's exact test. For one-way tables, PROC FREQTAB provides exact *p*-values for the Pearson and likelihood ratio chi-square goodness-of-fit tests. You can request these exact tests by specifying the corresponding options in the EXACT statement. See the section "Exact Statistics" on page 547 for more information.

The Mantel-Haenszel chi-square statistic is appropriate only when both variables lie on an ordinal scale. The other chi-square tests and statistics in this section are appropriate for either nominal or ordinal variables. The following sections give the formulas that PROC FREQTAB uses to compute the chi-square tests and statistics. For more information about these statistics, see Agresti (2007) and Stokes, Davis, and Koch (2012), and the other references cited.

Chi-Square Test for One-Way Tables

For one-way frequency tables, the CHISQ option in the TABLES statement provides a chi-square goodness-of-fit test. Let C denote the number of classes, or levels, in the one-way table. Let f_i denote the frequency of class i (or the number of observations in class i) for $i = 1, 2, \ldots, C$. Then PROC FREQTAB computes the one-way chi-square statistic as

$$Q_P = \sum_{i=1}^{C} (f_i - e_i)^2 / e_i$$

where e_i is the expected frequency for class i under the null hypothesis.

In the test for equal proportions, which is the default for the CHISQ option, the null hypothesis specifies equal proportions of the total sample size for each class. Under this null hypothesis, the expected frequency for each class equals the total sample size divided by the number of classes,

$$e_i = n \,/\, C \quad \text{for} \quad i = 1, 2, \ldots, C$$

In the test for specified frequencies, which PROC FREQTAB computes when you input null hypothesis frequencies by using the TESTF= option, the expected frequencies are the TESTF= values that you specify. In the test for specified proportions, which PROC FREQTAB computes when you input null hypothesis proportions by using the TESTP= option, the expected frequencies are determined from the specified TESTP= proportions p_i as

$$e_i = p_i \times n \quad \text{for} \quad i = 1, 2, \ldots, C$$

Under the null hypothesis (of equal proportions, specified frequencies, or specified proportions), Q_P has an asymptotic chi-square distribution with C–1 degrees of freedom.

In addition to the asymptotic test, you can request an exact one-way chi-square test by specifying the CHISQ option in the EXACT statement. See the section "Exact Statistics" on page 547 for more information.

Pearson Chi-Square Test for Two-Way Tables

The Pearson chi-square for two-way tables involves the differences between the observed and expected frequencies, where the expected frequencies are computed under the null hypothesis of independence. The Pearson chi-square statistic is computed as

$$Q_P = \sum_{i} \sum_{j} (n_{ij} - e_{ij})^2 / e_{ij}$$

where n_{ij} is the observed frequency in table cell (i, j) and e_{ij} is the expected frequency in table cell (i, j). The expected frequency is computed under the null hypothesis that the row and column variables are independent,

$$e_{ij} = (n_{i\cdot} \times n_{\cdot j}) \; / \; n$$

When the row and column variables are independent, Q_P has an asymptotic chi-square distribution with $(R-1)(C-1)$ degrees of freedom. For large values of Q_P, this test rejects the null hypothesis in favor of the alternative hypothesis of general association.

In addition to the asymptotic test, you can request an exact Pearson chi-square test by specifying the PCHI or CHISQ option in the EXACT statement. See the section "Exact Statistics" on page 547 for more information.

For 2×2 tables, the Pearson chi-square is also appropriate for testing the equality of two binomial proportions. For $R \times 2$ and $2 \times C$ tables, the Pearson chi-square tests the homogeneity of proportions. For more information, see Fienberg (1980).

Standardized Residuals

When you specify the CROSSLIST(STDRES) option in the TABLES statement for two-way or multiway tables, PROC FREQTAB displays the standardized residuals in the CROSSLIST table.

The standardized residual of a crosstabulation table cell is the ratio of (*frequency* – *expected*) to its standard error, where *frequency* is the table cell frequency and *expected* is the estimated expected cell frequency. The expected frequency is computed under the null hypothesis that the row and column variables are independent. See the section "Pearson Chi-Square Test for Two-Way Tables" on page 477 for more information.

PROC FREQTAB computes the standardized residual of table cell (i, j) as

$$(n_{ij} - e_{ij}) \; / \; \sqrt{e_{ij}(1 - p_{i\cdot})(1 - p_{\cdot j})}$$

where n_{ij} is the observed frequency of table cell (i, j), e_{ij} is the expected frequency of the table cell, $p_{i\cdot}$ is the proportion in row i $(n_{i\cdot}/n)$, and $p_{\cdot j}$ is the proportion in column j $(n_{\cdot j}/n)$. The expected frequency of table cell (i, j) is computed as

$$e_{ij} = (n_{i\cdot} \times n_{\cdot j}) \; / \; n$$

Under the null hypothesis of independence, each standardized residual has an asymptotic standard normal distribution. See section 2.4.5 of Agresti (2007) for more information.

Likelihood Ratio Chi-Square Test for One-Way Tables

For one-way frequency tables, the CHISQ(LRCHI) option in the TABLES statement provides a likelihood ratio chi-square goodness-of-fit test. By default, the likelihood ratio test is based on the null hypothesis of equal proportions in the C classes (levels) of the one-way table. If you specify null hypothesis proportions or frequencies by using the CHISQ(TESTP=) or CHISQ(TESTF=) option, respectively, the likelihood ratio test is based on the null hypothesis values that you specify.

PROC FREQTAB computes the one-way likelihood ratio test as

$$G^2 = 2 \sum_{i=1}^{C} f_i \log(f_i / e_i)$$

where f_i is the observed frequency of class i, and e_i is the expected frequency of class i under the null hypothesis.

For the null hypothesis of equal proportions, the expected frequency of each class is the total sample size divided by the number of classes,

$$e_i = n \,/\, C \quad \text{for} \quad i = 1, 2, \ldots, C$$

If you provide null hypothesis frequencies by specifying the CHISQ(TESTF=) option in the TABLES statement, the expected frequencies are the TESTF= values that you specify. If you provide null hypothesis proportions by specifying the CHISQ(TESTP=) option in the TABLES statement, PROC FREQTAB computes the expected frequencies as

$$e_i = p_i \times n \quad \text{for} \quad i = 1, 2, \ldots, C$$

where the proportions p_i are the TESTP= values that you specify.

Under the null hypothesis (of equal proportions, specified frequencies, or specified proportions), the likelihood ratio statistic G^2 has an asymptotic chi-square distribution with C–1 degrees of freedom.

In addition to the asymptotic test, you can request an exact one-way likelihood ratio chi-square test by specifying the LRCHI option in the EXACT statement. See the section "Exact Statistics" on page 547 for more information.

Likelihood Ratio Chi-Square Test

The likelihood ratio chi-square involves the ratios between the observed and expected frequencies. The likelihood ratio chi-square statistic is computed as

$$G^2 = 2 \sum_i \sum_j n_{ij} \log\left(n_{ij}/e_{ij}\right)$$

where n_{ij} is the observed frequency in table cell (i, j) and e_{ij} is the expected frequency for table cell (i, j).

When the row and column variables are independent, G^2 has an asymptotic chi-square distribution with $(R–1)(C–1)$ degrees of freedom.

In addition to the asymptotic test, you can request an exact likelihood ratio chi-square test by specifying the LRCHI or CHISQ option in the EXACT statement. See the section "Exact Statistics" on page 547 for more information.

Continuity-Adjusted Chi-Square Test

The continuity-adjusted chi-square for 2×2 tables is similar to the Pearson chi-square, but it is adjusted for the continuity of the chi-square distribution. The continuity-adjusted chi-square is most useful for small sample sizes. The use of the continuity adjustment is somewhat controversial; this chi-square test is more conservative (and more like Fisher's exact test) when the sample size is small. As the sample size increases, the continuity-adjusted chi-square becomes more like the Pearson chi-square.

The continuity-adjusted chi-square statistic is computed as

$$Q_C = \sum_i \sum_j \left(\max(0, |n_{ij} - e_{ij}| - 0.5) \right)^2 \,/\, e_{ij}$$

Under the null hypothesis of independence, Q_C has an asymptotic chi-square distribution with $(R–1)(C–1)$ degrees of freedom.

Mantel-Haenszel Chi-Square Test

The Mantel-Haenszel chi-square statistic tests the alternative hypothesis that there is a linear association between the row variable and the column variable. Both variables must lie on an ordinal scale. The Mantel-Haenszel chi-square statistic is computed as

$$Q_{\mathrm{MH}} = (n-1)r^2$$

where r is the Pearson correlation between the row variable and the column variable. For a description of the Pearson correlation, see the "Pearson Correlation Coefficient" on page 485. The Pearson correlation and thus the Mantel-Haenszel chi-square statistic use the scores that you specify in the SCORES= option in the TABLES statement. See Mantel and Haenszel (1959) and Landis, Heyman, and Koch (1978) for more information.

Under the null hypothesis of no association, Q_{MH} has an asymptotic chi-square distribution with 1 degree of freedom.

In addition to the asymptotic test, you can request an exact Mantel-Haenszel chi-square test by specifying the MHCHI or CHISQ option in the EXACT statement. See the section "Exact Statistics" on page 547 for more information.

Fisher's Exact Test

Fisher's exact test is another test of association between the row and column variables. This test assumes that the row and column totals are fixed and uses the hypergeometric distribution to compute probabilities of possible tables conditional on the observed row and column totals. Fisher's exact test does not depend on any large-sample distribution assumptions, and so it is appropriate even for small sample sizes and for sparse tables.

2 × 2 Tables For 2 × 2 tables, PROC FREQTAB gives the following information for Fisher's exact test: table probability, two-sided p-value, left-sided p-value, and right-sided p-value. The table probability is the hypergeometric probability of the observed table, and is in fact the value of the test statistic for Fisher's exact test.

Where p is the hypergeometric probability of a specific table with the observed row and column totals, Fisher's exact p-values are computed by summing probabilities p over defined sets of tables,

$$\mathrm{Prob} = \sum_{A} p$$

The two-sided p-value is the sum of all possible table probabilities (conditional on the observed row and column totals) that are less than or equal to the observed table probability. For the two-sided p-value, the set A includes all possible tables with hypergeometric probabilities less than or equal to the probability of the observed table. A small two-sided p-value supports the alternative hypothesis of association between the row and column variables.

For 2 × 2 tables, one-sided p-values for Fisher's exact test are defined in terms of the frequency of the cell in the first row and first column of the table, the (1,1) cell. Denoting the observed (1,1) cell frequency by n_{11}, the left-sided p-value for Fisher's exact test is the probability that the (1,1) cell frequency is less than or equal to n_{11}. For the left-sided p-value, the set A includes those tables with a (1,1) cell frequency less than or equal to n_{11}. A small left-sided p-value supports the alternative hypothesis that the probability of an observation being in the first cell is actually less than expected under the null hypothesis of independent row and column variables.

Similarly, for a right-sided alternative hypothesis, A is the set of tables where the frequency of the (1,1) cell is greater than or equal to that in the observed table. A small right-sided p-value supports the alternative that the probability of the first cell is actually greater than that expected under the null hypothesis.

Because the (1,1) cell frequency completely determines the 2×2 table when the marginal row and column sums are fixed, these one-sided alternatives can be stated equivalently in terms of other cell probabilities or ratios of cell probabilities. The left-sided alternative is equivalent to an odds ratio less than 1, where the odds ratio is $(n_{11}n_{22}/n_{12}n_{21})$. The left-sided alternative is also equivalent to the column 1 risk for row 1 being less than the column 1 risk for row 2, $p_{1|1} < p_{1|2}$. Similarly, the right-sided alternative is equivalent to the column 1 risk for row 1 being greater than the column 1 risk for row 2, $p_{1|1} > p_{1|2}$. For more information, see Agresti (2007).

R × C Tables Fisher's exact test was extended to general $R \times C$ tables by Freeman and Halton (1951), and this test is also known as the Freeman-Halton test. For $R \times C$ tables, the two-sided p-value definition is the same as for 2×2 tables. The set A contains all tables with p less than or equal to the probability of the observed table. A small p-value supports the alternative hypothesis of association between the row and column variables. For $R \times C$ tables, Fisher's exact test is inherently two-sided. The alternative hypothesis is defined only in terms of general, and not linear, association. Therefore, Fisher's exact test does not have right-sided or left-sided p-values for general $R \times C$ tables.

For $R \times C$ tables, PROC FREQTAB computes Fisher's exact test by using the network algorithm of Mehta and Patel (1983), which provides a faster and more efficient solution than direct enumeration. See the section "Exact Statistics" on page 547 for more details.

Phi Coefficient

The phi coefficient is a measure of association derived from the Pearson chi-square. The range of the phi coefficient is $-1 \leq \phi \leq 1$ for 2×2 tables. For tables larger than 2×2, the range is $0 \leq \phi \leq \min(\sqrt{R-1}, \sqrt{C-1})$ (Liebetrau 1983). The phi coefficient is computed as

$$\phi = (n_{11}n_{22} - n_{12}n_{21}) \,/\, \sqrt{n_{1\cdot}n_{2\cdot}n_{\cdot 1}n_{\cdot 2}} \quad \text{for } 2 \times 2 \text{ tables}$$

$$\phi = \sqrt{Q_P/n} \quad \text{otherwise}$$

See Fleiss, Levin, and Paik (2003, pp. 98–99) for more information.

Contingency Coefficient

The contingency coefficient is a measure of association derived from the Pearson chi-square. The range of the contingency coefficient is $0 \leq P \leq \sqrt{(m-1)/m}$, where $m = \min(R, C)$ (Liebetrau 1983). The contingency coefficient is computed as

$$P = \sqrt{Q_P \,/\, (Q_P + n)}$$

See Kendall and Stuart (1979, pp. 587–588) for more information.

Cramér's V

Cramér's V is a measure of association derived from the Pearson chi-square. It is designed so that the attainable upper bound is always 1. The range of Cramér's V is $-1 \leq V \leq 1$ for 2×2 tables; for tables larger than 2×2, the range is $0 \leq V \leq 1$. Cramér's V is computed as

$$V = \phi \quad \text{for } 2 \times 2 \text{ tables}$$

$$V = \sqrt{\frac{Q_P/n}{\min(R-1, C-1)}} \quad \text{otherwise}$$

See Kendall and Stuart (1979, p. 588) for more information.

Measures of Association

When you specify the MEASURES option in the TABLES statement, PROC FREQTAB computes several statistics that describe the association between the row and column variables of the contingency table. The following are measures of ordinal association that consider whether the column variable Y tends to increase as the row variable X increases: gamma, Kendall's tau-*b*, Stuart's tau-*c*, and Somers' *D*. These measures are appropriate for ordinal variables, and they classify pairs of observations as *concordant* or *discordant*. A pair is concordant if the observation with the larger value of X also has the larger value of Y. A pair is discordant if the observation with the larger value of X has the smaller value of Y. See Agresti (2007) and the other references cited for the individual measures of association.

The Pearson correlation coefficient and the Spearman rank correlation coefficient are also appropriate for ordinal variables. The Pearson correlation describes the strength of the linear association between the row and column variables, and it is computed by using the row and column scores specified by the SCORES= option in the TABLES statement. The Spearman correlation is computed with rank scores. The polychoric correlation (requested by the PLCORR option) also requires ordinal variables and assumes that the variables have an underlying bivariate normal distribution. The following measures of association do not require ordinal variables and are appropriate for nominal variables: lambda asymmetric, lambda symmetric, and the uncertainty coefficients.

PROC FREQTAB computes estimates of the measures according to the formulas given in the following sections. For each measure, PROC FREQTAB computes an asymptotic standard error (ASE), which is the square root of the asymptotic variance denoted by Var in the following sections.

Confidence Limits

If you specify the CL option in the TABLES statement, PROC FREQTAB computes asymptotic confidence limits for all MEASURES statistics. The confidence coefficient is determined according to the value of the ALPHA= option, which, by default, is 0.05 and produces 95% confidence limits.

The confidence limits are computed as

$$\text{Est} \pm (z_{\alpha/2} \times \text{ASE})$$

where Est is the estimate of the measure, $z_{\alpha/2}$ is the $100(1-\alpha/2)$th percentile of the standard normal distribution, and ASE is the asymptotic standard error of the estimate.

Asymptotic Tests

For each measure that you specify in the TEST statement, PROC FREQTAB computes an asymptotic test of the null hypothesis that the measure is 0. Asymptotic tests are available for the following measures of association: gamma, Kendall's tau-*b*, Stuart's tau-*c*, Somers' $D(C|R)$, Somers' $D(R|C)$, the Pearson correlation coefficient, and the Spearman rank correlation coefficient. To compute an asymptotic test, PROC FREQTAB uses a standardized test statistic *z*, which has an asymptotic standard normal distribution under the null hypothesis. The test statistic is computed as

$$z = \text{Est} \; / \; \sqrt{\text{Var}_0(\text{Est})}$$

where Est is the estimate of the measure and $\mathrm{Var}_0(\mathrm{Est})$ is the variance of the estimate under the null hypothesis. Formulas for $\mathrm{Var}_0(\mathrm{Est})$ for the individual measures of association are given in the following sections.

Note that the ratio of Est to $\sqrt{\mathrm{Var}_0(\mathrm{Est})}$ is the same for the following measures: gamma, Kendall's tau-*b*, Stuart's tau-*c*, Somers' $D(C|R)$, and Somers' $D(R|C)$. Therefore, the tests for these measures are identical. For example, the *p*-values for the test of $H_0\colon \mathrm{gamma} = 0$ equal the *p*-values for the test of $H_0\colon \mathrm{tau} - b = 0$.

PROC FREQTAB computes one-sided and two-sided *p*-values for each of these tests. When the test statistic z is greater than its null hypothesis expected value of 0, PROC FREQTAB displays the right-sided *p*-value, which is the probability of a larger value of the statistic occurring under the null hypothesis. A small right-sided *p*-value supports the alternative hypothesis that the true value of the measure is greater than 0. When the test statistic is less than or equal to 0, PROC FREQTAB displays the left-sided *p*-value, which is the probability of a smaller value of the statistic occurring under the null hypothesis. A small left-sided *p*-value supports the alternative hypothesis that the true value of the measure is less than 0. The one-sided *p*-value P_1 can be expressed as

$$P_1 = \begin{cases} \mathrm{Prob}(Z > z) & \text{if } z > 0 \\ \mathrm{Prob}(Z < z) & \text{if } z \leq 0 \end{cases}$$

where Z has a standard normal distribution. The two-sided *p*-value P_2 is computed as

$$P_2 = \mathrm{Prob}(|Z| > |z|)$$

Exact Tests

Exact tests are available for the following measures of association: Kendall's tau-*b*, Stuart's tau-*c*, Somers' $D(C|R)$ and $(R|C)$, the Pearson correlation coefficient, and the Spearman rank correlation coefficient. If you request an exact test for a measure of association in the EXACT statement, PROC FREQTAB computes the exact test of the hypothesis that the measure is 0. For more information, see the section "Exact Statistics" on page 547.

Gamma

The gamma (Γ) statistic is based only on the number of concordant and discordant pairs of observations. It ignores tied pairs (that is, pairs of observations that have equal values of X or equal values of Y). Gamma is appropriate only when both variables lie on an ordinal scale. The range of gamma is $-1 \leq \Gamma \leq 1$. If the row and column variables are independent, gamma tends to be close to 0. Gamma is computed as

$$G = (P - Q) \,/\, (P + Q)$$

and the asymptotic variance is

$$\mathrm{Var}(G) = \frac{16}{(P+Q)^4} \sum_i \sum_j n_{ij} (Q A_{ij} - P D_{ij})^2$$

For 2×2 tables, gamma is equivalent to Yule's Q. See Goodman and Kruskal (1979) and Agresti (2002) for more information.

The variance under the null hypothesis that gamma equals 0 is computed as

$$\mathrm{Var}_0(G) = \frac{4}{(P+Q)^2} \left(\sum_i \sum_j n_{ij} (A_{ij} - D_{ij})^2 - (P - Q)^2 / n \right)$$

For more information, see Brown and Benedetti (1977b).

Kendall's Tau-b

Kendall's tau-b (τ_b) is similar to gamma except that tau-b uses a correction for ties. Tau-b is appropriate only when both variables lie on an ordinal scale. The range of tau-b is $-1 \leq \tau_b \leq 1$. Kendall's tau-b is computed as

$$t_b = (P - Q) \,/\, \sqrt{w_r w_c}$$

and the asymptotic variance is

$$\mathrm{Var}(t_b) = \frac{1}{w^4}\left(\sum_i \sum_j n_{ij}(2wd_{ij} + t_b v_{ij})^2 - n^3 t_b^2 (w_r + w_c)^2\right)$$

where

$$\begin{aligned} w &= \sqrt{w_r w_c} \\ w_r &= n^2 - \sum_i n_{i.}^2 \\ w_c &= n^2 - \sum_j n_{.j}^2 \\ d_{ij} &= A_{ij} - D_{ij} \\ v_{ij} &= n_{i.} w_c + n_{.j} w_r \end{aligned}$$

See Kendall (1955) for more information.

The variance under the null hypothesis that tau-b equals 0 is computed as

$$\mathrm{Var}_0(t_b) = \frac{4}{w_r w_c}\left(\sum_i \sum_j n_{ij}(A_{ij} - D_{ij})^2 - (P - Q)^2/n\right)$$

For more information, see Brown and Benedetti (1977b).

PROC FREQTAB also provides an exact test for the Kendall's tau-b. You can request this test by specifying the KENTB option in the EXACT statement. See the section "Exact Statistics" on page 547 for more information.

Stuart's Tau-c

Stuart's tau-c (τ_c) makes an adjustment for table size in addition to a correction for ties. Tau-c is appropriate only when both variables lie on an ordinal scale. The range of tau-c is $-1 \leq \tau_c \leq 1$. Stuart's tau-c is computed as

$$t_c = m(P - Q) \,/\, n^2(m - 1)$$

and the asymptotic variance is

$$\mathrm{Var}(t_c) = \frac{4m^2}{(m-1)^2 n^4}\left(\sum_i \sum_j n_{ij} d_{ij}^2 - (P - Q)^2/n\right)$$

where $m = \min(R, C)$ and $d_{ij} = A_{ij} - D_{ij}$. The variance under the null hypothesis that tau-c equals 0 is the same as the asymptotic variance

$$\mathrm{Var}_0(t_c) = \mathrm{Var}(t_c)$$

For more information, see Brown and Benedetti (1977b).

PROC FREQTAB also provides an exact test for the Stuart's tau-c. You can request this test by specifying the STUTC option in the EXACT statement. See the section "Exact Statistics" on page 547 for more information.

Somers' D

Somers' $D(C|R)$ and Somers' $D(R|C)$ are asymmetric modifications of tau-b. $C|R$ indicates that the row variable X is regarded as the independent variable and the column variable Y is regarded as dependent. Similarly, $R|C$ indicates that the column variable Y is regarded as the independent variable and the row variable X is regarded as dependent. Somers' D differs from tau-b in that it uses a correction only for pairs that are tied on the independent variable. Somers' D is appropriate only when both variables lie on an ordinal scale. The range of Somers' D is $-1 \leq D \leq 1$. Somers' $D(C|R)$ is computed as

$$D(C|R) = (P - Q) \,/\, w_r$$

and its asymptotic variance is

$$\mathrm{Var}(D(C|R)) = \frac{4}{w_r^4} \sum_i \sum_j n_{ij} \left(w_r d_{ij} - (P - Q)(n - n_{i.}) \right)^2$$

where $d_{ij} = A_{ij} - D_{ij}$ and

$$w_r = n^2 - \sum_i n_{i.}^2$$

For more information, see Somers (1962); Goodman and Kruskal (1979); Liebetrau (1983).

The variance under the null hypothesis that $D(C|R)$ equals 0 is computed as

$$\mathrm{Var}_0(D(C|R)) = \frac{4}{w_r^2} \left(\sum_i \sum_j n_{ij} (A_{ij} - D_{ij})^2 - (P - Q)^2 / n \right)$$

For more information, see Brown and Benedetti (1977b).

Formulas for Somers' $D(R|C)$ are obtained by interchanging the indices.

PROC FREQTAB also provides exact tests for Somers' $D(C|R)$ and $(R|C)$. You can request these tests by specifying the SMDCR and SMDCR options in the EXACT statement. See the section "Exact Statistics" on page 547 for more information.

Pearson Correlation Coefficient

The Pearson correlation coefficient (ρ) is computed by using the scores specified in the SCORES= option. This measure is appropriate only when both variables lie on an ordinal scale. The range of the Pearson correlation is $-1 \leq \rho \leq 1$. The Pearson correlation coefficient is computed as

$$r = v/w = s_{rc} / \sqrt{s_r s_c}$$

and its asymptotic variance is

$$\mathrm{Var}(r) = \frac{1}{w^4} \sum_i \sum_j n_{ij} \left(w(R_i - \bar{R})(C_j - \bar{C}) - \frac{b_{ij} v}{2w} \right)^2$$

where R_i and C_j are the row and column scores and

$$\begin{aligned} s_r &= \sum_i \sum_j n_{ij} (R_i - \bar{R})^2 \\ s_c &= \sum_i \sum_j n_{ij} (C_j - \bar{C})^2 \\ s_{rc} &= \sum_i \sum_j n_{ij} (R_i - \bar{R})(C_j - \bar{C}) \end{aligned}$$

$$\begin{aligned} b_{ij} &= (R_i - \bar{R})^2 s_c + (C_j - \bar{C})^2 s_r \\ v &= s_{rc} \\ w &= \sqrt{s_r s_c} \end{aligned}$$

For more information, see Snedecor and Cochran (1989).

The SCORES= option in the TABLES statement determines the type of row and column scores used to compute the Pearson correlation (and other score-based statistics). The default is SCORES=TABLE. See the section "Scores" on page 475 for details about the available score types and how they are computed.

The variance under the null hypothesis that the correlation equals 0 is computed as

$$\mathrm{Var}_0(r) = \left(\sum_i \sum_j n_{ij} (R_i - \bar{R})^2 (C_j - \bar{C})^2 - ss_{rc}^2 / n \right) \; / \; ss_r \, ss_c$$

This expression for the variance is derived for multinomial sampling in a contingency table framework, and it differs from the form obtained under the assumption that both variables are continuous and normally distributed. For more information, see Brown and Benedetti (1977b).

PROC FREQTAB also provides an exact test for the Pearson correlation coefficient. You can request this test by specifying the PCORR option in the EXACT statement. See the section "Exact Statistics" on page 547 for more information.

Spearman Rank Correlation Coefficient

The Spearman correlation coefficient (ρ_s) is computed by using rank scores, which are defined in the section "Scores" on page 475. This measure is appropriate only when both variables lie on an ordinal scale. The range of the Spearman correlation is $-1 \leq \rho_s \leq 1$. The Spearman correlation coefficient is computed as

$$r_s = v \; / \; w$$

and its asymptotic variance is

$$\mathrm{Var}(r_s) = \frac{1}{n^2 w^4} \sum_i \sum_j n_{ij} (z_{ij} - \bar{z})^2$$

where R_i and C_j are the row and column rank scores and

$$\begin{aligned}
v &= \sum_i \sum_j n_{ij} R(i) C(j) \\
w &= \frac{1}{12} \sqrt{FG} \\
F &= n^3 - \sum_i n_{i\cdot}^3 \\
G &= n^3 - \sum_j n_{\cdot j}^3 \\
R(i) &= R_i - n/2 \\
C(j) &= C_j - n/2 \\
\bar{z} &= \frac{1}{n} \sum_i \sum_j n_{ij} z_{ij} \\
z_{ij} &= w v_{ij} - v w_{ij}
\end{aligned}$$

$$\begin{aligned}
v_{ij} &= n \left(R(i) C(j) + \frac{1}{2} \sum_l n_{il} C(l) + \frac{1}{2} \sum_k n_{kj} R(k) + \right. \\
&\qquad \left. \sum_l \sum_{k>i} n_{kl} C(l) + \sum_k \sum_{l>j} n_{kl} R(k) \right) \\
w_{ij} &= \frac{-n}{96w} \left(F n_{\cdot j}^2 + G n_{i\cdot}^2 \right)
\end{aligned}$$

For more information, see Snedecor and Cochran (1989).

The variance under the null hypothesis that the correlation equals 0 is computed as

$$\mathrm{Var}_0(r_s) = \frac{1}{n^2 w^2} \sum_i \sum_j n_{ij} (v_{ij} - \bar{v})^2$$

where

$$\bar{v} = \sum_i \sum_j n_{ij} v_{ij} / n$$

This expression for the variance is derived for multinomial sampling in a contingency table framework, and it differs from the form obtained under the assumption that both variables are continuous and normally distributed. For more information, see Brown and Benedetti (1977b).

PROC FREQTAB also provides an exact test for the Spearman correlation coefficient. You can request this test by specifying the SCORR option in the EXACT statement. For more information, see the section "Exact Statistics" on page 547.

Polychoric Correlation

When you specify the PLCORR option in the TABLES statement, PROC FREQTAB computes the polychoric correlation and its standard error. The polychoric correlation is based on the assumption that the two ordinal, categorical variables of the frequency table have an underlying bivariate normal distribution. The polychoric correlation coefficient is the maximum likelihood estimate of the product-moment correlation between the underlying normal variables. The range of the polychoric correlation is from –1 to 1. For 2×2 tables, the polychoric correlation is also known as the tetrachoric correlation (and it is labeled as such in the displayed output). See Drasgow (1986) for an overview of polychoric correlation coefficient.

Olsson (1979) gives the likelihood equations and the asymptotic standard errors for estimating the polychoric correlation. The underlying continuous variables relate to the observed crosstabulation table through thresholds, which define a range of numeric values that correspond to each categorical (table) level. PROC FREQTAB uses Olsson's maximum likelihood method for simultaneous estimation of the polychoric correlation and the thresholds. (Olsson also presents a two-step method that estimates the thresholds first.)

PROC FREQTAB iteratively solves the likelihood equations by using a Newton-Raphson algorithm. The initial estimates of the thresholds are computed from the inverse of the normal distribution function at the cumulative marginal proportions of the table. Iterative computation of the polychoric correlation stops when the convergence measure falls below the convergence criterion or when the maximum number of iterations is reached, whichever occurs first. For parameter values that are less than 0.01, the procedure evaluates convergence by using the absolute difference instead of the relative difference. The PLCORR(CONVERGE=) option specifies the convergence criterion, which is 0.0001 by default. The PLCORR(MAXITER=) option specifies the maximum number of iterations, which is 20 by default.

If you specify the CL option in the TABLES statement, PROC FREQTAB provides confidence limits for the polychoric correlation. The confidence limits are computed as

$$\hat{\rho} \pm (z_{\alpha/2} \times \mathrm{se}(\hat{\rho}))$$

where $\hat{\rho}$ is the estimate of the polychoric correlation, $z_{\alpha/2}$ is the $100(1-\alpha/2)$th percentile of the standard normal distribution, and $\mathrm{se}(\hat{\rho})$ is the standard error of the polychoric correlation estimate.

If you specify the PLCORR option in the TEST statement, PROC FREQTAB provides Wald and likelihood ratio tests of the null hypothesis that the polychoric correlation is 0. The Wald test statistic is computed as

$$z = \hat{\rho} \,/\, \mathrm{se}(\hat{\rho})$$

which has a standard normal distribution under the null hypothesis. PROC FREQTAB computes one-sided and two-sided p-values for the Wald test. When the test statistic z is greater than its null expected value of 0, PROC FREQTAB displays the right-sided p-value. When the test statistic is less than or equal to 0, PROC FREQTAB displays the left-sided p-value.

The likelihood ratio statistic for the polychoric correlation is computed as

$$G^2 = -2 \log(L_0/L_1)$$

where L_0 is the value of the likelihood function (Olsson 1979) when the polychoric correlation is 0, and L_1 is the value of the likelihood function at the maximum (where all parameters are replaced with their maximum likelihood estimates). Under the null hypothesis, the likelihood ratio statistic has an asymptotic chi-square distribution with 1 degree of freedom.

Lambda (Asymmetric)

Asymmetric lambda, $\lambda(C|R)$, is interpreted as the probable improvement in predicting the column variable Y given knowledge of the row variable X. The range of asymmetric lambda is $0 \leq \lambda(C|R) \leq 1$. Asymmetric lambda $(C|R)$ is computed as

$$\lambda(C|R) = \frac{\sum_i r_i - r}{n - r}$$

and its asymptotic variance is

$$\mathrm{Var}(\lambda(C|R)) = \frac{n - \sum_i r_i}{(n-r)^3} \left(\sum_i r_i + r - 2 \sum_i (r_i \mid l_i = l) \right)$$

where

$$\begin{aligned} r_i &= \max_j(n_{ij}) \\ r &= \max_j(n_{\cdot j}) \\ c_j &= \max_i(n_{ij}) \\ c &= \max_i(n_{i\cdot}) \end{aligned}$$

The values of l_i and l are determined as follows. Denote by l_i the unique value of j such that $r_i = n_{ij}$, and let l be the unique value of j such that $r = n_{\cdot j}$. Because of the uniqueness assumptions, ties in the frequencies or in the marginal totals must be broken in an arbitrary but consistent manner. In case of ties, l is defined as the smallest value of j such that $r = n_{\cdot j}$.

For those columns containing a cell (i, j) for which $n_{ij} = r_i = c_j$, cs_j records the row in which c_j is assumed to occur. Initially cs_j is set equal to –1 for all j. Beginning with i=1, if there is at least one value j such that $n_{ij} = r_i = c_j$, and if $cs_j = -1$, l_i is defined to be the smallest such value of j, and cs_j is set equal to i. Otherwise, if $n_{il} = r_i$, l_i is defined to be equal to l. If neither condition is true, l_i is taken to be the smallest value of j such that $n_{ij} = r_i$.

The formulas for lambda asymmetric $(R|C)$ can be obtained by interchanging the indices.

For more information, see Goodman and Kruskal (1979).

Lambda (Symmetric)

The nondirectional lambda is the average of the two asymmetric lambdas, $\lambda(C|R)$ and $\lambda(R|C)$. Its range is $0 \leq \lambda \leq 1$. Lambda symmetric is computed as

$$\lambda = \frac{\sum_i r_i + \sum_j c_j - r - c}{2n - r - c} = \frac{w - v}{w}$$

and its asymptotic variance is computed as

$$\mathrm{Var}(\lambda) = \frac{1}{w^4}\Big(wvy - 2w^2\big(n - \sum_i \sum_j (n_{ij} \mid j = l_i, i = k_j)\big) - 2v^2(n - n_{kl})\Big)$$

where

$$\begin{aligned}
r_i &= \max_j(n_{ij}) \\
r &= \max_j(n_{\cdot j}) \\
c_j &= \max_i(n_{ij}) \\
c &= \max_i(n_{i\cdot}) \\
w &= 2n - r - c \\
v &= 2n - \sum_i r_i - \sum_j c_j \\
x &= \sum_i (r_i \mid l_i = l) + \sum_j (c_j \mid k_j = k) + r_k + c_l \\
y &= 8n - w - v - 2x
\end{aligned}$$

The definitions of l_i and l are given in the previous section. The values k_j and k are defined in a similar way for lambda asymmetric $(R|C)$.

For more information, see Goodman and Kruskal (1979).

Uncertainty Coefficients (Asymmetric)

The uncertainty coefficient $U(C|R)$ measures the proportion of uncertainty (entropy) in the column variable Y that is explained by the row variable X. Its range is $0 \leq U(C|R) \leq 1$. The uncertainty coefficient is computed as

$$U(C|R) = (H(X) + H(Y) - H(XY)) \;/\; H(Y) = v/w$$

and its asymptotic variance is

$$\mathrm{Var}(U(C|R)) = \frac{1}{n^2 w^4} \sum_i \sum_j n_{ij}\Big(H(Y)\log\Big(\frac{n_{ij}}{n_{i\cdot}}\Big) + (H(X) - H(XY))\log\Big(\frac{n_{\cdot j}}{n}\Big)\Big)^2$$

where

$$\begin{aligned} v &= H(X) + H(Y) - H(XY) \\ w &= H(Y) \\ H(X) &= -\sum_i \left(\frac{n_{i\cdot}}{n}\right) \log\left(\frac{n_{i\cdot}}{n}\right) \\ H(Y) &= -\sum_j \left(\frac{n_{\cdot j}}{n}\right) \log\left(\frac{n_{\cdot j}}{n}\right) \\ H(XY) &= -\sum_i \sum_j \left(\frac{n_{ij}}{n}\right) \ln\left(\frac{n_{ij}}{n}\right) \end{aligned}$$

The formulas for the uncertainty coefficient $U(R|C)$ can be obtained by interchanging the indices.

For more information, see Theil (1972, pp. 115–120) and Goodman and Kruskal (1979).

Uncertainty Coefficient (Symmetric)

The uncertainty coefficient U is the symmetric version of the two asymmetric uncertainty coefficients. Its range is $0 \leq U \leq 1$. The uncertainty coefficient is computed as

$$U = 2\left(H(X) + H(Y) - H(XY)\right) \;/\; \left(H(X) + H(Y)\right)$$

and its asymptotic variance is

$$\mathrm{Var}(U) = 4 \sum_i \sum_j \frac{n_{ij}\left(H(XY)\log\left(\frac{n_{i\cdot}n_{\cdot j}}{n^2}\right) - \left(H(X)+H(Y)\right)\log\left(\frac{n_{ij}}{n}\right)\right)^2}{n^2\left(H(X)+H(Y)\right)^4}$$

where $H(X)$, $H(Y)$, and $H(XY)$ are defined in the previous section. For more information, see Goodman and Kruskal (1979).

Binomial Proportion

If you specify the BINOMIAL option in the TABLES statement, PROC FREQTAB computes the binomial proportion for one-way tables. By default, this is the proportion of observations in the first variable level that appears in the output. (You can use the LEVEL= option to specify a different level for the proportion.) The binomial proportion is computed as

$$\hat{p} = n_1 \;/\; n$$

where n_1 is the frequency of the first (or designated) level and n is the total frequency of the one-way table. The standard error of the binomial proportion is computed as

$$\mathrm{se}(\hat{p}) = \sqrt{\hat{p}\;(1-\hat{p}) \;/\; n}$$

Binomial Confidence Limits

PROC FREQTAB provides Wald and exact (Clopper-Pearson) confidence limits for the binomial proportion. You can also request the following binomial confidence limit types by specifying the BINOMIAL(CL=) option: Agresti-Coull, Blaker, Jeffreys, exact mid-p, likelihood ratio, logit, and Wilson (score). For more information, see Brown, Cai, and DasGupta (2001), Agresti and Coull (1998), and Newcombe (1998b), in addition to the references cited for each confidence limit type.

Wald Confidence Limits Wald asymptotic confidence limits are based on the normal approximation to the binomial distribution. PROC FREQTAB computes the Wald confidence limits for the binomial proportion as

$$\hat{p} \pm (z_{\alpha/2} \times \mathrm{se}(\hat{p}))$$

where $z_{\alpha/2}$ is the $100(1-\alpha/2)$th percentile of the standard normal distribution. The confidence level α is determined by the ALPHA= option; by default, ALPHA=0.05, which produces 95% confidence limits.

If you specify CL=WALD(CORRECT) or the CORRECT *binomial-option*, PROC FREQTAB includes a continuity correction of $1/2n$ in the Wald asymptotic confidence limits. The purpose of this correction is to adjust for the difference between the normal approximation and the discrete binomial distribution. See Fleiss, Levin, and Paik (2003) for more information. The continuity-corrected Wald confidence limits for the binomial proportion are computed as

$$\hat{p} \pm (z_{\alpha/2} \times \mathrm{se}(\hat{p}) + (1/2n))$$

Exact (Clopper-Pearson) Confidence Limits Exact (Clopper-Pearson) confidence limits for the binomial proportion are constructed by inverting the equal-tailed test based on the binomial distribution. This method is attributed to Clopper and Pearson (1934). The exact confidence limits P_L and P_U satisfy the following equations, for $n_1 = 1, 2, \ldots n-1$:

$$\sum_{x=n_1}^{n} \binom{n}{x} P_L{}^x (1-P_L)^{n-x} = \alpha/2$$

$$\sum_{x=0}^{n_1} \binom{n}{x} P_U{}^x (1-P_U)^{n-x} = \alpha/2$$

The lower confidence limit is 0 when $n_1 = 0$, and the upper confidence limit is 1 when $n_1 = n$.

PROC FREQTAB computes the exact (Clopper-Pearson) confidence limits by using the F distribution as

$$P_L = \left(1 + \frac{n - n_1 + 1}{n_1 \, F(\, \alpha/2,\, 2n_1,\, 2(n-n_1+1)\,)} \right)^{-1}$$

$$P_U = \left(1 + \frac{n - n_1}{(n_1+1) \, F(\, 1-\alpha/2,\, 2(n_1+1),\, 2(n-n_1)\,)} \right)^{-1}$$

where $F(\alpha/2, b, c)$ is the $(\alpha/2)$th percentile of the F distribution with b and c degrees of freedom. See Leemis and Trivedi (1996) for a derivation of this expression. Also see Collett (1991) for more information about exact binomial confidence limits.

Because this is a discrete problem, the confidence coefficient (coverage probability) of the exact (Clopper-Pearson) interval is not exactly $(1-\alpha)$ but is at least $(1-\alpha)$. Thus, this confidence interval is conservative. Unless the sample size is large, the actual coverage probability can be much larger than the target value. For more information about the performance of these confidence limits, see Agresti and Coull (1998), Brown, Cai, and DasGupta (2001), and Leemis and Trivedi (1996).

Agresti-Coull Confidence Limits If you specify the CL=AGRESTICOULL *binomial-option*, PROC FREQTAB computes Agresti-Coull confidence limits for the binomial proportion as

$$\tilde{p} \pm (z_{\alpha/2} \times \sqrt{\tilde{p}\,(1-\tilde{p})\,/\,\tilde{n}})$$

where

$$\begin{aligned} \tilde{n}_1 &= n_1 + z^2_{\alpha/2}/2 \\ \tilde{n} &= n + z^2_{\alpha/2} \\ \tilde{p} &= \tilde{n}_1\,/\,\tilde{n} \end{aligned}$$

The Agresti-Coull confidence interval has the same general form as the standard Wald interval but uses $\tilde{p}$ in place of $\hat{p}$. For $\alpha = 0.05$, the value of $z_{\alpha/2}$ is close to 2, and this interval is the "add 2 successes and 2 failures" adjusted Wald interval of Agresti and Coull (1998).

Blaker Confidence Limits If you specify the CL=BLAKER *binomial-option*, PROC FREQTAB computes Blaker confidence limits for the binomial proportion, which are constructed by inverting the two-sided exact Blaker test (Blaker 2000). The $100(1-\alpha)\%$ Blaker confidence interval consists of all values of the proportion p_0 for which the test statistic $B(p_0, n_1)$ falls in the acceptance region,

$$\{p_0 : B(p_0, n_1) > \alpha\}$$

where

$$B(p_0, n_1) = \text{Prob}(\, \gamma(p_0, X) \le \gamma(p_0, n_1) \mid p_0 \,)$$

$$\gamma(p_0, n_1) = \min(\, \text{Prob}(\, X \ge n_1 \mid p_0 \,),\ \text{Prob}(\, X \le n_1 \mid p_0 \,) \,)$$

and X is a binomial random variable. For more information, see Blaker (2000).

Jeffreys Confidence Limits If you specify the CL=JEFFREYS *binomial-option*, PROC FREQTAB computes Jeffreys confidence limits for the binomial proportion as

$$\big(\, \beta(\alpha/2,\ n_1 + 1/2,\ n - n_1 + 1/2),\ \ \beta(1-\alpha/2,\ n_1 + 1/2,\ n - n_1 + 1/2) \,\big)$$

where $\beta(\alpha, b, c)$ is the αth percentile of the beta distribution with shape parameters b and c. The lower confidence limit is set to 0 when $n_1 = 0$, and the upper confidence limit is set to 1 when $n_1 = n$. This is an equal-tailed interval based on the noninformative Jeffreys prior for a binomial proportion. For more information, see Brown, Cai, and DasGupta (2001). For information about using beta priors for inference on the binomial proportion, see Berger (1985).

When you specify CL=JEFFREYS(MODIFY), the procedure provides modified Jeffreys confidence limits (Brown, Cai, and DasGupta 2001, p. 113) to improve the coverage of the confidence limits for proportions near 0 or 1. When $n_1 = 0$, the modified version replaces the upper confidence limit with $p_l = 1 - (\alpha/2)^{1/n}$. When $n_1 = n$, the modified version replaces the lower confidence limit with $1 - p_l$. The modified version also sets the lower confidence limit to 0 when $n_1 = 1$ and sets the upper confidence limit to 1 when $n_1 = n - 1$. Otherwise (when $1 < n_1 < n - 1$), the Jeffreys confidence limits are not modified.

Likelihood Ratio Confidence Limits If you specify the CL=LIKELIHOODRATIO *binomial-option*, PROC FREQTAB computes likelihood ratio confidence limits for the binomial proportion by inverting the likelihood ratio test. The likelihood ratio test statistic for the null hypothesis that the proportion equals p_0 can be expressed as

$$L(p_0) \;=\; -2\,(n_1 \log(\hat{p}/p_0) \;+\; (n-n_1)\log((1-\hat{p})/(1-p_0)))$$

The $100(1-\alpha)\%$ likelihood ratio confidence interval consists of all values of p_0 for which the test statistic $L(p_0)$ falls in the acceptance region,

$$\{p_0 : L(p_0) < \chi^2_{1,\alpha}\}$$

where $\chi^2_{1,\alpha}$ is the $100(1-\alpha)$th percentile of the chi-square distribution with 1 degree of freedom. PROC FREQTAB finds the confidence limits by iterative computation. For more information, see Fleiss, Levin, and Paik (2003), Brown, Cai, and DasGupta (2001), Agresti (2013), and Newcombe (1998b).

Logit Confidence Limits If you specify the CL=LOGIT *binomial-option*, PROC FREQTAB computes logit confidence limits for the binomial proportion, which are based on the logit transformation $Y = \log(\hat{p}/(1-\hat{p}))$. Approximate confidence limits for Y are computed as

$$Y_L \;=\; \log(\hat{p}/(1-\hat{p})) \;-\; z_{\alpha/2}\sqrt{n/(n_1(n-n_1))}$$

$$Y_U \;=\; \log(\hat{p}/(1-\hat{p})) \;+\; z_{\alpha/2}\sqrt{n/(n_1(n-n_1))}$$

The confidence limits for Y are inverted to produce $100(1-\alpha)\%$ logit confidence limits P_L and P_U for the binomial proportion p as

$$P_L \;=\; \exp(Y_L/(1+\exp(Y_L))$$

$$P_U \;=\; \exp(Y_U/(1+\exp(Y_U))$$

For more information, see Brown, Cai, and DasGupta (2001) and Korn and Graubard (1998).

Mid-*p* Confidence Limits If you specify the CL=MIDP *binomial-option*, PROC FREQTAB computes exact mid-p confidence limits for the binomial proportion by inverting two one-sided binomial tests that include mid-p tail areas. The mid-p approach replaces the probability of the observed frequency with half of that probability in the Clopper-Pearson sum, which is described in the section "Exact (Clopper-Pearson) Confidence Limits" on page 492. The exact mid-p confidence limits P_L and P_U are the solutions to the equations

$$\sum_{x=n_1+1}^{n}\binom{n}{x}P_L^{\,x}(1-P_L)^{\,n-x} \;+\; \frac{1}{2}\binom{n}{n_1}P_L^{\,n_1}(1-P_L)^{\,n-n_1} \;=\; \alpha/2$$

$$\sum_{x=0}^{n_1-1}\binom{n}{x}P_U^{\,x}(1-P_U)^{\,n-x} \;+\; \frac{1}{2}\binom{n}{n_1}P_U^{\,n_1}(1-P_U)^{\,n-n_1} \;=\; \alpha/2$$

For more information, see Agresti and Gottard (2007), Agresti (2013), Newcombe (1998b), and Brown, Cai, and DasGupta (2001).

Wilson (Score) Confidence Limits If you specify the CL=WILSON *binomial-option*, PROC FREQTAB computes Wilson confidence limits for the binomial proportion. These are also known as score confidence limits (Wilson 1927). The confidence limits are based on inverting the normal test that uses the null proportion in the variance (the score test). Wilson confidence limits are the roots of

$$|p - \hat{p}| = z_{\alpha/2}\sqrt{p(1-p)/n}$$

and are computed as

$$\left(\hat{p} + z^2_{\alpha/2}/2n \pm z_{\alpha/2}\sqrt{\left(\hat{p}(1-\hat{p}) + z^2_{\alpha/2}/4n\right)/n}\right) / \left(1 + z^2_{\alpha/2}/n\right)$$

When you specify CL=WILSON(CORRECT) or the CORRECT *binomial-option*, PROC FREQTAB provides continuity-corrected Wilson confidence limits, which are computed as the roots of

$$|p - \hat{p}| - 1/2n = z_{\alpha/2}\sqrt{p(1-p)/n}$$

The Wilson interval has been shown to have better performance than the Wald interval and the exact (Clopper-Pearson) interval. For more information, see Agresti and Coull (1998), Brown, Cai, and DasGupta (2001), and Newcombe (1998b).

When you specify CL=WILSON(ADAPT), PROC FREQTAB provides adapted Wilson confidence limits (Agresti and Coull 1998, p. 125) to improve the coverage of the confidence limits for proportions near 0 or 1. When $n_1 = 1$, the adapted version replaces the lower confidence limit with $(-\log(1-\alpha)/n)$. When $n_1 = (n-1)$, the adapted version replaces the upper confidence limit with $(1 + \log(1-\alpha)/n)$.

When you specify CL=WILSON(MODIFY), PROC FREQTAB provides modified Wilson confidence limits (Brown, Cai, and DasGupta 2001, pp. 112–113) to improve the coverage of the confidence limits for proportions near 0 or 1. The modified version replaces the lower confidence limit when $n_1 = 1, \ldots, n_1^*$, where $n_1^* = 2$ for $n \leq 50$ and $n_1^* = 3$ otherwise. The modified lower confidence limit is computed as $\chi^2(2n_1, \alpha)/2n$, where $\chi^2(2n_1, \alpha)$ is the 100αth percentile of the chi-square distribution with $2n_1$ degrees of freedom. Similarly, the modified version replaces the upper confidence limit when $n_1 = n - n_1^*, \ldots, n-1$. The modified upper confidence limit is computed as $1 - \chi^2(2(n-n_1), \alpha)/2n$.

Binomial Tests

The BINOMIAL option provides an asymptotic equality test for the binomial proportion by default. You can also specify *binomial-options* to request tests of noninferiority, superiority, and equivalence for the binomial proportion. If you specify the BINOMIAL option in the EXACT statement, PROC FREQTAB also computes exact p-values for the tests that you request with the *binomial-options*.

Equality Test PROC FREQTAB computes an asymptotic test of the hypothesis that the binomial proportion equals p_0, where you can specify the value of p_0 with the P= *binomial-option*. If you do not specify a null value with P=, PROC FREQTAB uses $p_0 = 0.5$ by default. The binomial test statistic is computed as

$$z = (\hat{p} - p_0)/\text{se}$$

By default, the standard error is based on the null hypothesis proportion as

$$\text{se} = \sqrt{p_0(1-p_0)/n}$$

If you specify the VAR=SAMPLE *binomial-option*, the standard error is computed from the sample proportion as

$$\text{se} = \sqrt{\hat{p}(1-\hat{p})/n}$$

If you specify the CORRECT *binomial-option*, PROC FREQTAB includes a continuity correction in the asymptotic test statistic, towards adjusting for the difference between the normal approximation and the discrete binomial distribution. For more information, see Fleiss, Levin, and Paik (2003). The continuity correction of $(1/2n)$ is subtracted from the numerator of the test statistic if $(\hat{p} - p_0)$ is positive; otherwise, the continuity correction is added to the numerator.

PROC FREQTAB computes one-sided and two-sided p-values for this test. When the test statistic z is greater than 0 (its expected value under the null hypothesis), PROC FREQTAB computes the right-sided p-value, which is the probability of a larger value of the statistic occurring under the null hypothesis. A small right-sided p-value supports the alternative hypothesis that the true value of the proportion is greater than p_0. When the test statistic is less than or equal to 0, PROC FREQTAB computes the left-sided p-value, which is the probability of a smaller value of the statistic occurring under the null hypothesis. A small left-sided p-value supports the alternative hypothesis that the true value of the proportion is less than p_0. The one-sided p-value P_1 can be expressed as

$$P_1 = \begin{cases} \text{Prob}(Z > z) & \text{if } z > 0 \\ \text{Prob}(Z < z) & \text{if } z \leq 0 \end{cases}$$

where Z has a standard normal distribution. The two-sided p-value is computed as $P_2 = 2 \times P_1$.

If you specify the BINOMIAL option in the EXACT statement, PROC FREQTAB also computes an exact test of the null hypothesis H_0: $p = p_0$. To compute the exact test, PROC FREQTAB uses the binomial probability function,

$$\text{Prob}(X = x \mid p_0) = \binom{n}{x} {p_0}^x \, (1-p_0)^{(n-x)} \quad \text{for } x = 0, 1, 2, \ldots, n$$

where the variable X has a binomial distribution with parameters n and p_0. To compute the left-sided p-value, $\text{Prob}(X \leq n_1)$, PROC FREQTAB sums the binomial probabilities over x from 0 to n_1. To compute the right-sided p-value, $\text{Prob}(X \geq n_1)$, PROC FREQTAB sums the binomial probabilities over x from n_1 to n. The exact one-sided p-value is the minimum of the left-sided and right-sided p-values,

$$P_1 = \min\left(\text{Prob}(X \leq n_1 \mid p_0),\ \text{Prob}(X \geq n_1 \mid p_0) \right)$$

and the exact two-sided p-value is computed as $P_2 = 2 \times P_1$.

Noninferiority Test If you specify the NONINF *binomial-option*, PROC FREQTAB provides a noninferiority test for the binomial proportion. The null hypothesis for the noninferiority test is

$$H_0: p - p_0 \leq -\delta$$

versus the alternative

$$H_a: p - p_0 > -\delta$$

where δ is the noninferiority margin and p_0 is the null proportion. Rejection of the null hypothesis indicates that the binomial proportion is not inferior to the null value. See Chow, Shao, and Wang (2003) for more information.

You can specify the value of δ with the MARGIN= *binomial-option*, and you can specify p_0 with the P= *binomial-option*. By default, $\delta = 0.2$ and $p_0 = 0.5$.

PROC FREQTAB provides an asymptotic Wald test for noninferiority. The test statistic is computed as

$$z = (\hat{p} - p_0^*) \,/\, \mathrm{se}$$

where p_0^* is the noninferiority limit,

$$p_0^* = p_0 - \delta$$

By default, the standard error is computed from the sample proportion as

$$\mathrm{se} = \sqrt{\hat{p}(1-\hat{p})/n}$$

If you specify the VAR=NULL *binomial-option*, the standard error is based on the noninferiority limit (determined by the null proportion and the margin) as

$$\mathrm{se} = \sqrt{p_0^*(1-p_0^*)/n}$$

If you specify the CORRECT *binomial-option*, PROC FREQTAB includes a continuity correction in the asymptotic test statistic z. The continuity correction of $(1/2n)$ is subtracted from the numerator of the test statistic if $(\hat{p} - p_0^*)$ is positive; otherwise, the continuity correction is added to the numerator.

The p-value for the noninferiority test is

$$P_z = \mathrm{Prob}(Z > z)$$

where Z has a standard normal distribution.

As part of the noninferiority analysis, PROC FREQTAB provides asymptotic Wald confidence limits for the binomial proportion. These confidence limits are computed as described in the section "Wald Confidence Limits" on page 492 but use the same standard error (VAR=NULL or VAR=SAMPLE) as the noninferiority test statistic z. The confidence coefficient is $100(1-2\alpha)\%$ (Schuirmann 1999). By default, if you do not specify the ALPHA= option, the noninferiority confidence limits are 90% confidence limits. You can compare the confidence limits to the noninferiority limit, $p_0^* = p_0 - \delta$.

If you specify the BINOMIAL option in the EXACT statement, PROC FREQTAB provides an exact noninferiority test for the binomial proportion. The exact p-value is computed by using the binomial probability function with parameters p_0^* and n,

$$P_x = \sum_{k=n_1}^{k=n} \binom{n}{k} (p_0^*)^{\,k}\, (1-p_0^*)^{\,(n-k)}$$

For more information, see Chow, Shao, and Wang (2003, p. 116). If you request exact binomial statistics, PROC FREQTAB also includes exact (Clopper-Pearson) confidence limits for the binomial proportion in the equivalence analysis display. For more information, see the section "Exact (Clopper-Pearson) Confidence Limits" on page 492.

Superiority Test If you specify the SUP *binomial-option*, PROC FREQTAB provides a superiority test for the binomial proportion. The null hypothesis for the superiority test is

$$H_0: p - p_0 \leq \delta$$

versus the alternative

$$H_a: p - p_0 > \delta$$

where δ is the superiority margin and p_0 is the null proportion. Rejection of the null hypothesis indicates that the binomial proportion is superior to the null value. You can specify the value of δ with the MARGIN= *binomial-option*, and you can specify the value of p_0 with the P= *binomial-option*. By default, $\delta = 0.2$ and $p_0 = 0.5$.

The superiority analysis is identical to the noninferiority analysis but uses a positive value of the margin δ in the null hypothesis. The superiority limit equals $p_0 + \delta$. The superiority computations follow those in the section "Noninferiority Test" on page 496 but replace $-\delta$ with δ. See Chow, Shao, and Wang (2003) for more information.

Equivalence Test If you specify the EQUIV *binomial-option*, PROC FREQTAB provides an equivalence test for the binomial proportion. The null hypothesis for the equivalence test is

$$H_0: p - p_0 \leq \delta_L \quad \text{or} \quad p - p_0 \geq \delta_U$$

versus the alternative

$$H_a: \delta_L < p - p_0 < \delta_U$$

where δ_L is the lower margin, δ_U is the upper margin, and p_0 is the null proportion. Rejection of the null hypothesis indicates that the binomial proportion is equivalent to the null value. See Chow, Shao, and Wang (2003) for more information.

You can specify the value of the margins δ_L and δ_U with the MARGIN= *binomial-option*. If you do not specify MARGIN=, PROC FREQTAB uses lower and upper margins of –0.2 and 0.2 by default. If you specify a single margin value δ, PROC FREQTAB uses lower and upper margins of $-\delta$ and δ. You can specify the null proportion p_0 with the P= *binomial-option*. By default, $p_0 = 0.5$.

PROC FREQTAB computes two one-sided tests (TOST) for equivalence analysis (Schuirmann 1987). The TOST approach includes a right-sided test for the lower margin and a left-sided test for the upper margin. The overall p-value is taken to be the larger of the two p-values from the lower and upper tests.

For the lower margin, the asymptotic Wald test statistic is computed as

$$z_L = (\hat{p} - p_L^*) \,/\, \text{se}$$

where the lower equivalence limit is

$$p_L^* = p_0 + \delta_L$$

By default, the standard error is computed from the sample proportion as

$$\text{se} = \sqrt{\hat{p}(1-\hat{p})/n}$$

If you specify the VAR=NULL *binomial-option*, the standard error is based on the lower equivalence limit (determined by the null proportion and the lower margin) as

$$\text{se} = \sqrt{p_L^*(1 - p_L^*)/n}$$

If you specify the CORRECT *binomial-option*, PROC FREQTAB includes a continuity correction in the asymptotic test statistic z_L. The continuity correction of $(1/2n)$ is subtracted from the numerator of the test statistic $(\hat{p} - p_L^*)$ if the numerator is positive; otherwise, the continuity correction is added to the numerator.

The p-value for the lower margin test is

$$P_{z,L} = \text{Prob}(Z > z_L)$$

The asymptotic test for the upper margin is computed similarly. The Wald test statistic is

$$z_U = (\hat{p} - p_U^*) \,/\, \text{se}$$

where the upper equivalence limit is

$$p_U^* = p_0 + \delta_U$$

By default, the standard error is computed from the sample proportion. If you specify the VAR=NULL *binomial-option*, the standard error is based on the upper equivalence limit as

$$\text{se} = \sqrt{p_U^*(1 - p_U^*)/n}$$

If you specify the CORRECT *binomial-option*, PROC FREQTAB includes a continuity correction of $(1/2n)$ in the asymptotic test statistic z_U.

The p-value for the upper margin test is

$$P_{z,U} = \text{Prob}(Z < z_U)$$

Based on the two one-sided tests (TOST), the overall p-value for the test of equivalence equals the larger p-value from the lower and upper margin tests, which can be expressed as

$$P_z = \max(P_{z,L}\,,\ P_{z,U})$$

As part of the equivalence analysis, PROC FREQTAB provides asymptotic Wald confidence limits for the binomial proportion. These confidence limits are computed as described in the section "Wald Confidence Limits" on page 492, but use the same standard error (VAR=NULL or VAR=SAMPLE) as the equivalence test statistics and have a confidence coefficient of $100(1 - 2\alpha)\%$ (Schuirmann 1999). By default, if you do not specify the ALPHA= option, the equivalence confidence limits are 90% limits. If you specify VAR=NULL, separate standard errors are computed for the lower and upper margin tests, each based on the null proportion and the corresponding (lower or upper) margin. The confidence limits are computed by using the maximum of these two standard errors. You can compare the confidence limits to the equivalence limits, $(p_0 + \delta_L,\ p_0 + \delta_U)$.

If you specify the BINOMIAL option in the EXACT statement, PROC FREQTAB also provides an exact equivalence test by using two one-sided exact tests (TOST). The procedure computes lower and upper margin exact tests by using the binomial probability function as described in the section "Noninferiority Test" on page 496. The overall exact p-value for the equivalence test is taken to be the larger p-value from the lower and upper margin exact tests. If you request exact statistics, PROC FREQTAB also includes exact (Clopper-Pearson) confidence limits in the equivalence analysis display. The confidence coefficient is $100(1 - 2\alpha)\%$ (Schuirmann 1999). For more information, see the section "Exact (Clopper-Pearson) Confidence Limits" on page 492.

Sensitivity and Specificity

The SENSPEC option in the TABLES statement provides estimates of sensitivity, specificity, positive predictive value, and negative predictive value for 2×2 tables. These measures are conditional (row and column) proportions in the 2×2 crosstabulation table. In sensitivity analysis, the row variable might represent a positive or negative diagnostic test result, and the column variable might represent the presence or absence of a condition. For more information, see Fleiss, Levin, and Paik (2003). The SENSPEC option also provides an estimate of accuracy, which is the proportion of correct classifications.

By default, the SENSPEC computations use table cell (1,1) as the reference cell that represents the true positives (positive row value and positive column value). You can specify a different reference cell by using the SENSPEC REFCELL= suboption or the REFCOLUMN= and REFROW= suboptions. For example, REFCELL=4 specifies table cell (2,2) as the true-positive cell (and row 2 and column 2 as the positive row and column levels, respectively). The notation in this section assumes that table cell (1,1) is the reference cell.

Sensitivity is computed as the column proportion for table cell (1,1), which is the ratio of the frequency in table cell (1,1) to the total frequency in column 1 of the 2×2 table. Sensitivity is denoted by

$$\mathrm{SN} = n_{11} \,/\, n_{\cdot 1}$$

Specificity is computed as the column proportion for table cell (2,2), which is the ratio of the frequency in table cell (2,2) to the total frequency in column 2. Specificity is denoted by

$$\mathrm{SP} = n_{22} \,/\, n_{\cdot 2}$$

The positive predictive value is the row proportion for table cell (1,1), which is computed as

$$\mathrm{PPV} = n_{11} \,/\, n_{1\cdot}$$

The negative predictive value is the row proportion for table cell (2,2), which is computed as

$$\mathrm{NPV} = n_{22} \,/\, n_{2\cdot}$$

Accuracy (correct classification rate) is the overall proportion in table cells (1,1) and (2,2), which is computed as

$$\mathrm{A} = (n_{11} + n_{22}) \,/\, n$$

The "Sensitivity and Specificity" table provides the estimates together with their standard errors and Wald confidence limits. PROC FREQTAB computes the standard errors and Wald confidence limits for these proportions as described in the section "Risks and Risk Differences" on page 501. The value of the confidence coefficient α is determined by the ALPHA= option; by default, ALPHA=0.05, which produces 95% confidence limits.

Risks and Risk Differences

The RISKDIFF option in the TABLES statement provides estimates of risks (binomial proportions) and risk differences for 2×2 tables. This analysis might be appropriate when comparing the proportion of some characteristic for two groups, where row 1 and row 2 correspond to the two groups, and the columns correspond to two possible characteristics or outcomes. For example, the row variable might be a treatment or dose, and the column variable might be the response. For more information, see Collett (1991); Fleiss, Levin, and Paik (2003); Stokes, Davis, and Koch (2012).

Let the frequencies of the 2×2 table be represented as follows:

	Column 1	Column 2	Total
Row 1	n_{11}	n_{12}	$n_{1\cdot}$
Row 2	n_{21}	n_{22}	$n_{2\cdot}$
Total	$n_{\cdot 1}$	$n_{\cdot 2}$	n

By default when you specify the RISKDIFF option, PROC FREQTAB provides estimates of the row 1 risk (proportion), the row 2 risk, the overall risk, and the risk difference for column 1 and for column 2 of the 2×2 table. The risk difference is defined as the row 1 risk minus the row 2 risk. The risks are binomial proportions of their rows (row 1, row 2, or overall), and the computation of their standard errors and Wald confidence limits follow the binomial proportion computations, which are described in the section "Binomial Proportion" on page 491.

The column 1 risk for row 1 is the proportion of row 1 observations classified in column 1,

$$\hat{p}_1 = n_{11} \,/\, n_{1\cdot}$$

which estimates the conditional probability of the column 1 response, given the first level of the row variable. The column 1 risk for row 2 is the proportion of row 2 observations classified in column 1,

$$\hat{p}_2 = n_{21} \,/\, n_{2\cdot}$$

The overall column 1 risk is the proportion of all observations classified in column 1,

$$\hat{p} = n_{\cdot 1} \,/\, n$$

The column 1 risk difference compares the risks for the two rows, and it is computed as the column 1 risk for row 1 minus the column 1 risk for row 2,

$$\hat{d} = \hat{p}_1 - \hat{p}_2$$

The standard error of the column 1 risk for row i is computed as

$$\mathrm{se}(\hat{p}_i) = \sqrt{\hat{p}_i \,(1 - \hat{p}_i) \,/\, n_{i\cdot}}$$

The standard error of the overall column 1 risk is computed as

$$\mathrm{se}(\hat{p}) = \sqrt{\hat{p} \,(1 - \hat{p}) \,/\, n}$$

Where the two rows represent independent binomial samples, the standard error of the column 1 risk difference is computed as

$$\mathrm{se}(\hat{d}) = \sqrt{\hat{p}_1(1 - \hat{p}_1)/n_{1\cdot} \;+\; \hat{p}_2(1 - \hat{p}_2)/n_{2\cdot}}$$

The computations are similar for the column 2 risks and risk difference.

Confidence Limits

By default, the RISKDIFF option provides Wald asymptotic confidence limits for the risks (row 1, row 2, and overall) and the risk difference. By default, the RISKDIFF option also provides exact (Clopper-Pearson) confidence limits for the risks. You can suppress the display of this information by specifying the NORISKS *riskdiff-option*. You can specify *riskdiff-options* to request tests and other types of confidence limits for the risk difference. For more information, see the sections "Confidence Limits for the Risk Difference" on page 502 and "Risk Difference Tests" on page 506.

The risks are equivalent to the binomial proportions of their corresponding rows. This section describes the Wald confidence limits that are provided by default when you specify the RISKDIFF option. The BINOMIAL option provides additional confidence limit types and tests for risks (binomial proportions). For more information, see the sections "Binomial Confidence Limits" on page 492 and "Binomial Tests" on page 495.

The Wald confidence limits are based on the normal approximation to the binomial distribution. PROC FREQTAB computes the Wald confidence limits for the risks and risk differences as

$$\text{Est} \pm \left(z_{\alpha/2} \times \text{se}(\text{Est}) \right)$$

where Est is the estimate, $z_{\alpha/2}$ is the $100(1-\alpha/2)$th percentile of the standard normal distribution, and se(Est) is the standard error of the estimate. The confidence level α is determined by the value of the ALPHA= option; by default, ALPHA=0.05, which produces 95% confidence limits.

If you specify the CORRECT *riskdiff-option*, PROC FREQTAB includes continuity corrections in the Wald confidence limits for the risks and risk differences. The purpose of a continuity correction is to adjust for the difference between the normal approximation and the binomial distribution, which is discrete. See Fleiss, Levin, and Paik (2003) for more information. The continuity-corrected Wald confidence limits are computed as

$$\text{Est} \pm \left(z_{\alpha/2} \times \text{se}(\text{Est}) + c \right)$$

where c is the continuity correction. For the row 1 risk, $c = (1/2n_{1\cdot})$; for the row 2 risk, $c = (1/2n_{2\cdot})$; for the overall risk, $c = (1/2n)$; and for the risk difference, $c = ((1/n_{1\cdot} + 1/n_{2\cdot})/2)$. The column 1 and column 2 risks use the same continuity correction.

By default when you specify the RISKDIFF option, PROC FREQTAB also provides exact (Clopper-Pearson) confidence limits for the column 1, column 2, and overall risks. These confidence limits are constructed by inverting the equal-tailed test that is based on the binomial distribution. For more information, see the section "Exact (Clopper-Pearson) Confidence Limits" on page 492.

Confidence Limits for the Risk Difference PROC FREQTAB provides the following confidence limit types for the risk difference: Agresti-Caffo, exact unconditional, Hauck-Anderson, Miettinen-Nurminen (score), Newcombe (hybrid-score), and Wald confidence limits. Continuity-corrected forms of Newcombe and Wald confidence limits are also available.

The confidence coefficient for the confidence limits produced by the CL= *riskdiff-option* is $100(1-\alpha)\%$, where the value of α is determined by the ALPHA= option. By default, ALPHA=0.05, which produces 95% confidence limits. This differs from the test-based confidence limits that are provided with the equivalence, noninferiority, and superiority tests, which have a confidence coefficient of $100(1-2\alpha)\%$ (Schuirmann 1999). For more information, see the section "Risk Difference Tests" on page 506.

Agresti-Caffo Confidence Limits

Agresti-Caffo confidence limits for the risk difference are computed as

$$\tilde{d} \pm (z_{\alpha/2} \times \mathrm{se}(\tilde{d}))$$

where $\tilde{d} = \tilde{p}_1 - \tilde{p}_2$, $\tilde{p}_i = (n_{i1} + 1)/(n_{i\cdot} + 2)$,

$$\mathrm{se}(\tilde{d}) = \sqrt{\tilde{p}_1(1-\tilde{p}_2)/(n_{1\cdot}+2) + \tilde{p}_2(1-\tilde{p}_2)/(n_{2\cdot}+2)}$$

and $z_{\alpha/2}$ is the $100(1-\alpha/2)$th percentile of the standard normal distribution.

The Agresti-Caffo interval adjusts the Wald interval for the risk difference by adding a pseudo-observation of each type (success and failure) to each sample. See Agresti and Caffo (2000) and Agresti and Coull (1998) for more information.

Hauck-Anderson Confidence Limits

Hauck-Anderson confidence limits for the risk difference are computed as

$$\hat{d} \pm (c + z_{\alpha/2} \times \mathrm{se}(\hat{d}))$$

where $\hat{d} = \hat{p}_1 - \hat{p}_2$ and $z_{\alpha/2}$ is the $100(1-\alpha/2)$th percentile of the standard normal distribution. The standard error is computed from the sample proportions as

$$\mathrm{se}(\hat{d}) = \sqrt{\hat{p}_1(1-\hat{p}_1)/(n_{1\cdot}-1) + \hat{p}_2(1-\hat{p}_2)/(n_{2\cdot}-1)}$$

The Hauck-Anderson continuity correction c is computed as

$$c = 1 \,/\, \left(2 \min(n_{1\cdot}, n_{2\cdot})\right)$$

For more information, see Hauck and Anderson (1986).

The subsection "Hauck-Anderson Test" in the section "Noninferiority Tests" on page 507 describes the corresponding noninferiority test.

Miettinen-Nurminen (Score) Confidence Limits

Miettinen-Nurminen (score) confidence limits for the risk difference (Miettinen and Nurminen 1985) are computed by inverting score tests for the risk difference. A score-based test statistic for the null hypothesis that the risk difference equals δ can be expressed as

$$T(\delta) = (\hat{d} - \delta)/\sqrt{\widetilde{\mathrm{Var}}(\delta)}$$

where $\hat{d}$ is the observed value of the risk difference $(\hat{p}_1 - \hat{p}_2)$,

$$\widetilde{\mathrm{Var}}(\delta) = (n/(n-1)) \; (\tilde{p}_1(\delta)(1-\tilde{p}_1(\delta))/n_1 + \tilde{p}_2(\delta)(1-\tilde{p}_2(\delta))/n_2)$$

and $\tilde{p}_1(\delta)$ and $\tilde{p}_2(\delta)$ are the maximum likelihood estimates of the row 1 and row 2 risks (proportions) under the restriction that the risk difference is δ. For more information, see Miettinen and Nurminen (1985, pp. 215–216) and Miettinen (1985, chapter 12).

The $100(1-\alpha)\%$ confidence interval for the risk difference consists of all values of δ for which the score test statistic $T(\delta)$ falls in the acceptance region,

$$\{\delta : T(\delta) < z_{\alpha/2}\}$$

where $z_{\alpha/2}$ is the $100(1-\alpha/2)$th percentile of the standard normal distribution. PROC FREQTAB finds the confidence limits by iterative computation, which stops when the iteration increment falls below the convergence criterion or when the maximum number of iterations is reached, whichever occurs first. By default, the convergence criterion is 0.00000001 and the maximum number of iterations is 100.

By default, the Miettinen-Nurminen confidence limits include the bias correction factor $n/(n-1)$ in the computation of $\widetilde{\mathrm{Var}}(\delta)$ (Miettinen and Nurminen 1985, p. 216). For more information, see Newcombe and Nurminen (2011). If you specify the CL=MN(CORRECT=NO) *riskdiff-option*, PROC FREQTAB does not include the bias correction factor in this computation (Mee 1984). See also Agresti (2002, p. 77). The uncorrected confidence limits are labeled as "Miettinen-Nurminen-Mee" confidence limits in the displayed output.

The maximum likelihood estimates of p_1 and p_2, subject to the constraint that the risk difference is δ, are computed as

$$\tilde{p}_1 = 2u\cos(w) - b/3a \quad \text{and} \quad \tilde{p}_2 = \tilde{p}_1 - \delta$$

where

$$\begin{aligned}
w &= (\pi + \cos^{-1}(v/u^3))/3 \\
v &= b^3/(3a)^3 - bc/6a^2 + d/2a \\
u &= \mathrm{sign}(v)\sqrt{b^2/(3a)^2 - c/3a} \\
a &= 1+\theta \\
b &= -(1+\theta+\hat{p}_1+\theta\hat{p}_2+\delta(\theta+2)) \\
c &= \delta^2 + \delta(2\hat{p}_1+\theta+1) + \hat{p}_1 + \theta\hat{p}_2 \\
d &= -\hat{p}_1\delta(1+\delta) \\
\theta &= n_{2\cdot}/n_{1\cdot}
\end{aligned}$$

For more information, see Farrington and Manning (1990, p. 1453).

Newcombe Confidence Limits

Newcombe (hybrid-score) confidence limits for the risk difference are constructed from the Wilson score confidence limits for each of the two individual proportions. The confidence limits for the individual proportions are used in the standard error terms of the Wald confidence limits for the proportion difference. See Newcombe (1998a) and Barker et al. (2001) for more information.

Wilson score confidence limits for p_1 and p_2 are the roots of

$$|p_i - \hat{p}_i| = z_{\alpha/2}\sqrt{p_i(1-p_i)/n_{i\cdot}}$$

for $i = 1, 2$. The confidence limits are computed as

$$\left(\hat{p}_i + z^2_{\alpha/2}/2n_{i\cdot} \pm z_{\alpha/2}\sqrt{\left(\hat{p}_i(1-\hat{p}_i) + z^2_{\alpha/2}/4n_{i\cdot}\right)/n_{i\cdot}}\right) / \left(1 + z^2_{\alpha/2}/n_{i\cdot}\right)$$

For more information, see the section "Wilson (Score) Confidence Limits" on page 495.

Denote the lower and upper Wilson score confidence limits for p_1 as L_1 and U_1, and denote the lower and upper confidence limits for p_2 as L_2 and U_2. The Newcombe confidence limits for the proportion difference ($d = p_1 - p_2$) are computed as

$$d_L = (\hat{p}_1 - \hat{p}_2) - \sqrt{(\hat{p}_1 - L_1)^2 + (U_2 - \hat{p}_2)^2}$$

$$d_U = (\hat{p}_1 - \hat{p}_2) + \sqrt{(U_1 - \hat{p}_1)^2 + (\hat{p}_2 - L_2)^2}$$

If you specify the CORRECT *riskdiff-option*, PROC FREQTAB provides continuity-corrected Newcombe confidence limits. By including a continuity correction of $1/2n_{i\cdot}$, the Wilson score confidence limits for the individual proportions are computed as the roots of

$$|p_i - \hat{p}_i| - 1/2n_{i\cdot} = z_{\alpha/2}\sqrt{p_i(1-p_i)/n_{i\cdot}}$$

The continuity-corrected confidence limits for the individual proportions are then used to compute the proportion difference confidence limits d_L and d_U.

Wald Confidence Limits

Wald confidence limits for the risk difference are computed as

$$\hat{d} \pm (z_{\alpha/2} \times \mathrm{se}(\hat{d}))$$

where $\hat{d} = \hat{p}_1 - \hat{p}_2$, $z_{\alpha/2}$ is the $100(1-\alpha/2)$th percentile of the standard normal distribution. and the standard error is computed from the sample proportions as

$$\mathrm{se}(\hat{d}) = \sqrt{\hat{p}_1(1-\hat{p}_1)/n_{1\cdot} + \hat{p}_2(1-\hat{p}_2)/n_{2\cdot}}$$

If you specify the CORRECT *riskdiff-option*, the Wald confidence limits include a continuity correction cc,

$$\hat{d} \pm (c + z_{\alpha/2} \times \mathrm{se}(\hat{d}))$$

where $c = (1/n_{1\cdot} + 1/n_{2\cdot})/2$.

The subsection "Wald Test" in the section "Noninferiority Tests" on page 507 describes the corresponding noninferiority test.

Exact Unconditional Confidence Limits

If you specify the RISKDIFF option in the EXACT statement, PROC FREQTAB provides exact unconditional confidence limits for the risk difference ($d = p_1 - p_2$). The exact unconditional approach fixes the row margins of the 2×2 table and eliminates the nuisance parameter p_2 by using the maximum p-value (worst-case scenario) over all possible values of p_2 (Santner and Snell 1980). The conditional approach, which is described in the section "Exact Statistics" on page 547, does not apply to the risk difference because of the nuisance parameter (Agresti 1992).

By default, PROC FREQTAB computes the confidence limits by the tail method, which inverts two separate one-sided exact tests of the risk difference, where the tests are based on the score statistic (Chan and Zhang 1999). The size of each one-sided exact test is at most $\alpha/2$, and the confidence coefficient is at least

$(1-\alpha)$. If you specify the RISKDIFF(METHOD=NOSCORE) option in the EXACT statement, PROC FREQTAB computes the confidence limits by inverting two separate one-sided exact tests that are based on the unstandardized risk difference. If you specify the RISKDIFF(METHOD=SCORE2) option in the EXACT statement, PROC FREQTAB computes the confidence limits by inverting a single two-sided exact test that is based on the score statistic (Agresti and Min 2001).

The score statistic is a less discrete statistic than the unstandardized risk difference and produces less conservative confidence limits (Agresti and Min 2001). For more information, see Santner et al. (2007). The section "Miettinen-Nurminen (Score) Confidence Limits" describe computation of the risk difference score statistic. For more information, see Miettinen and Nurminen (1985) and Farrington and Manning (1990).

PROC FREQTAB computes the exact unconditional confidence limits as follows. The risk difference is defined as the difference between the row 1 and row 2 risks (proportions), $d = p_1 - p_2$, and n_1 and n_2 denote the row totals of the 2×2 table. The joint probability function for the table can be expressed in terms of the table cell frequencies, the risk difference, and the nuisance parameter p_2 as

$$f(n_{11}, n_{21}; n_1, n_2, d, p_2) = \binom{n_1}{n_{11}} (d + p_2)^{n_{11}} (1 - d - p_2)^{n_1 - n_{11}} \times \binom{n_2}{n_{21}} p_2^{n_{21}} (1 - p_2)^{n_2 - n_{21}}$$

For the tail method (which inverts two separate one-sided exact tests), the $100(1-\alpha/2)\%$ confidence limits for the risk difference are computed as

$$\begin{aligned} d_L &= \sup\,(d_* : P_U(d_*) > \alpha/2) \\ d_U &= \inf\,(d_* : P_L(d_*) > \alpha/2) \end{aligned}$$

where

$$\begin{aligned} P_U(d_*) &= \sup_{p_2} \Big(\sum_{A, T(a) \geq t_0} f(n_{11}, n_{21}; n_1, n_2, d_*, p_2) \Big) \\ P_L(d_*) &= \sup_{p_2} \Big(\sum_{A, T(a) \leq t_0} f(n_{11}, n_{21}; n_1, n_2, d_*, p_2) \Big) \end{aligned}$$

The set A includes all 2×2 tables in which the row sums are n_1 and n_2, $T(a)$ denotes the value of the test statistic for table a in A, and t_0 is the value of the test statistic for the observed table. The test statistic is either the score statistic (by default) or the unstandardized risk difference. To compute $P_U(d_*)$, the sum includes probabilities of those tables for which $(T(a) \geq t_0)$. For a fixed value of d_*, $P_U(d_*)$ is defined as the maximum sum over all possible values of p_2.

The two-sided score method evaluates the p-values $P_U(d_*)$ and $P_L(d_*)$ by comparing $|T(a)|$ to $|t_0|$. To compute the confidence limits d_L and d_u, the two-sided method compares the p-values to α. For more information, see Agresti and Min (2001) and Santner et al. (2007).

Risk Difference Tests

PROC FREQTAB provides tests of equality, noninferiority, superiority, and equivalence for the risk (proportion) difference. The following analysis methods are available: Wald (with and without continuity correction), Hauck-Anderson, Farrington-Manning (score), and Newcombe (with and without continuity correction). You can specify the method by using the METHOD= *riskdiff-option*; by default, PROC FREQTAB provides Wald tests.

Equality Tests The equality test for the risk difference tests the null hypothesis that the risk difference equals the null value. You can specify a null value by using the EQUAL(NULL=) *riskdiff-option*; by default, the null value is 0. This test can be expressed as $H_0: d = d_0$ versus the alternative $H_a: d \neq d_0$, where $d = p_1 - p_2$ denotes the risk difference (for column 1 or column 2) and d_0 denotes the null value.

The test statistic is computed as

$$z = (\hat{d} - d_0)/\mathrm{se}(\hat{d})$$

where the standard error $\mathrm{se}(\hat{d})$ is computed by using the method that you specify. Available methods for the equality test include Wald (with and without continuity correction), Hauck-Anderson, and Farrington-Manning (score). For a description of the standard error computation, see the subsections "Wald Test," "Hauck-Anderson Test," and "Farrington-Manning (Score) Test," respectively, in the section "Noninferiority Tests" on page 507.

PROC FREQTAB computes one-sided and two-sided p-values for equality tests. When the test statistic z is greater than 0, PROC FREQTAB displays the right-sided p-value, which is the probability of a larger value occurring under the null hypothesis. The one-sided p-value can be expressed as

$$P_1 = \begin{cases} \mathrm{Prob}(Z > z) & \text{if } z > 0 \\ \mathrm{Prob}(Z < z) & \text{if } z \leq 0 \end{cases}$$

where Z has a standard normal distribution. The two-sided p-value is computed as $P_2 = 2 \times P_1$.

Noninferiority Tests If you specify the NONINF *riskdiff-option*, PROC FREQTAB provides a noninferiority test for the risk difference, or the difference between two proportions. The null hypothesis for the noninferiority test is

$$H_0: p_1 - p_2 \leq -\delta$$

versus the alternative

$$H_a: p_1 - p_2 > -\delta$$

where δ is the noninferiority margin. Rejection of the null hypothesis indicates that the row 1 risk is not inferior to the row 2 risk. See Chow, Shao, and Wang (2003) for more information.

You can specify the value of δ with the MARGIN= *riskdiff-option*. By default, $\delta = 0.2$. You can specify the test method with the METHOD= *riskdiff-option*. The following methods are available for the risk difference noninferiority analysis: Wald (with and without continuity correction), Hauck-Anderson, Farrington-Manning (score), and Newcombe (with and without continuity correction). The Wald, Hauck-Anderson, and Farrington-Manning methods provide tests and corresponding test-based confidence limits; the Newcombe method provides only confidence limits. If you do not specify METHOD=, PROC FREQTAB uses the Wald test by default.

The confidence coefficient for the test-based confidence limits is $100(1 - 2\alpha)\%$ (Schuirmann 1999). By default, if you do not specify the ALPHA= option, these are 90% confidence limits. You can compare the confidence limits to the noninferiority limit, $-\delta$.

The following sections describe the noninferiority analysis methods for the risk difference.

Wald Test

If you specify the METHOD=WALD *riskdiff-option*, PROC FREQTAB provides an asymptotic Wald test of noninferiority for the risk difference. This is also the default method. The Wald test statistic is computed as

$$z = (\hat{d} + \delta) \,/\, \mathrm{se}(\hat{d})$$

where ($\hat{d} = \hat{p}_1 - \hat{p}_2$) estimates the risk difference and δ is the noninferiority margin.

By default, the standard error for the Wald test is computed from the sample proportions as

$$\mathrm{se}(\hat{d}) = \sqrt{\hat{p}_1(1-\hat{p}_1)/n_{1\cdot} \;+\; \hat{p}_2(1-\hat{p}_2)/n_{2\cdot}}$$

If you specify the VAR=NULL *riskdiff-option*, the standard error is based on the null hypothesis that the risk difference equals –δ (Dunnett and Gent 1977). The standard error is computed as

$$\mathrm{se}(\hat{d}) = \sqrt{\tilde{p}(1-\tilde{p})/n_{2\cdot} \;+\; (\tilde{p}-\delta)(1-\tilde{p}+\delta)/n_{1\cdot}}$$

where

$$\tilde{p} = (n_{11} + n_{21} + \delta n_{1\cdot})/n$$

If you specify the CORRECT *riskdiff-option*, the test statistic includes a continuity correction. The continuity correction is subtracted from the numerator of the test statistic if the numerator is greater than 0; otherwise, the continuity correction is added to the numerator. The value of the continuity correction is $(1/n_{1\cdot} + 1/n_{2\cdot})/2$.

The p-value for the Wald noninferiority test is $P_z = \mathrm{Prob}(Z > z)$, where Z has a standard normal distribution.

Hauck-Anderson Test

If you specify the METHOD=HA *riskdiff-option*, PROC FREQTAB provides the Hauck-Anderson test for noninferiority. The Hauck-Anderson test statistic is computed as

$$z = (\hat{d} + \delta \;\pm\; c) \,/\, \mathrm{se}(\hat{d})$$

where $\hat{d} = \hat{p}_1 - \hat{p}_2$ and the standard error is computed from the sample proportions as

$$\mathrm{se}(\hat{d}) = \sqrt{\hat{p}_1(1-\hat{p}_1)/(n_{1\cdot}-1) \;+\; \hat{p}_2(1-\hat{p}_2)/(n_{2\cdot}-1)}$$

The Hauck-Anderson continuity correction c is computed as

$$c = 1 \,/\, \left(2\, \min(n_{1\cdot},\, n_{2\cdot})\right)$$

The p-value for the Hauck-Anderson noninferiority test is $P_z = \mathrm{Prob}(Z > z)$, where Z has a standard normal distribution. For more information, see Hauck and Anderson (1986) and Schuirmann (1999).

Farrington-Manning (Score) Test

If you specify the METHOD=FM *riskdiff-option*, PROC FREQTAB provides the Farrington-Manning (score) test of noninferiority for the risk difference. A score test statistic for the null hypothesis that the risk difference equals –δ can be expressed as

$$z = (\hat{d} + \delta) \,/\, \mathrm{se}(\hat{d})$$

where $\hat{d}$ is the observed value of the risk difference $(\hat{p}_1 - \hat{p}_2)$,

$$\mathrm{se}(\hat{d}) = \sqrt{\tilde{p}_1(1-\tilde{p}_1)/n_{1\cdot} + \tilde{p}_2(1-\tilde{p}_2)/n_{2\cdot}}$$

and $\tilde{p}_1$ and $\tilde{p}_2$ are the maximum likelihood estimates of the row 1 and row 2 risks (proportions) under the restriction that the risk difference is $-\delta$. The p-value for the noninferiority test is $P_z = \mathrm{Prob}(Z > z)$, where Z has a standard normal distribution. For more information, see Miettinen and Nurminen (1985); Miettinen (1985); Farrington and Manning (1990); Dann and Koch (2005).

The maximum likelihood estimates of p_1 and p_1, subject to the constraint that the risk difference is $-\delta$, are computed as

$$\tilde{p}_1 = 2u\cos(w) - b/3a \quad \text{and} \quad \tilde{p}_2 = \tilde{p}_1 + \delta$$

where

$$\begin{aligned}
w &= (\pi + \cos^{-1}(v/u^3))/3 \\
v &= b^3/(3a)^3 - bc/6a^2 + d/2a \\
u &= \mathrm{sign}(v)\sqrt{b^2/(3a)^2 - c/3a} \\
a &= 1 + \theta \\
b &= -(1 + \theta + \hat{p}_1 + \theta\hat{p}_2 - \delta(\theta + 2)) \\
c &= \delta^2 - \delta(2\hat{p}_1 + \theta + 1) + \hat{p}_1 + \theta\hat{p}_2 \\
d &= \hat{p}_1\delta(1-\delta) \\
\theta &= n_{2\cdot}/n_{1\cdot}
\end{aligned}$$

For more information, see Farrington and Manning (1990, p. 1453).

Newcombe Noninferiority Analysis

If you specify the METHOD=NEWCOMBE *riskdiff-option*, PROC FREQTAB provides a noninferiority analysis that is based on Newcombe hybrid-score confidence limits for the risk difference. The confidence coefficient for the confidence limits is $100(1-2\alpha)\%$ (Schuirmann 1999). By default, if you do not specify the ALPHA= option, these are 90% confidence limits. You can compare the confidence limits with the noninferiority limit, $-\delta$. If you specify the CORRECT *riskdiff-option*, the confidence limits includes a continuity correction. See the subsection "Newcombe Confidence Limits" in the section "Confidence Limits for the Risk Difference" on page 502 for more information.

Superiority Test If you specify the SUP *riskdiff-option*, PROC FREQTAB provides a superiority test for the risk difference. The null hypothesis is

$$H_0: p_1 - p_2 \leq \delta$$

versus the alternative

$$H_a: p_1 - p_2 > \delta$$

where δ is the superiority margin. Rejection of the null hypothesis indicates that the row 1 proportion is superior to the row 2 proportion. You can specify the value of δ with the MARGIN= *riskdiff-option*. By default, $\delta = 0.2$.

The superiority analysis is identical to the noninferiority analysis but uses a positive value of the margin δ in the null hypothesis. The superiority computations follow those in the section "Noninferiority Tests" on page 507 by replacing $-\delta$ by δ. See Chow, Shao, and Wang (2003) for more information.

Equivalence Test If you specify the EQUIV *riskdiff-option*, PROC FREQTAB provides an equivalence test for the risk difference, or the difference between two proportions. The null hypothesis for the equivalence test is

$$H_0: p_1 - p_2 \leq -\delta_L \quad \text{or} \quad p_1 - p_2 \geq \delta_U$$

versus the alternative

$$H_a: \delta_L < p_1 - p_2 < \delta_U$$

where δ_L is the lower margin and δ_U is the upper margin. Rejection of the null hypothesis indicates that the two binomial proportions are equivalent. See Chow, Shao, and Wang (2003) for more information.

You can specify the value of the margins δ_L and δ_U with the MARGIN= *riskdiff-option*. If you do not specify MARGIN=, PROC FREQTAB uses lower and upper margins of –0.2 and 0.2 by default. If you specify a single margin value δ, PROC FREQTAB uses lower and upper margins of $-\delta$ and δ. You can specify the test method with the METHOD= *riskdiff-option*. The following methods are available for the risk difference equivalence analysis: Wald (with and without continuity correction), Hauck-Anderson, Farrington-Manning (score), and Newcombe (with and without continuity correction). The Wald, Hauck-Anderson, and Farrington-Manning methods provide tests and corresponding test-based confidence limits; the Newcombe method provides only confidence limits. If you do not specify METHOD=, PROC FREQTAB uses the Wald test by default.

PROC FREQTAB computes two one-sided tests (TOST) for equivalence analysis (Schuirmann 1987). The TOST approach includes a right-sided test for the lower margin δ_L and a left-sided test for the upper margin δ_U. The overall p-value is taken to be the larger of the two p-values from the lower and upper tests.

The section "Noninferiority Tests" on page 507 gives details about the Wald, Hauck-Anderson, Farrington-Manning (score), and Newcombe methods for the risk difference. The lower margin equivalence test statistic takes the same form as the noninferiority test statistic but uses the lower margin value δ_L in place of $-\delta$. The upper margin equivalence test statistic take the same form as the noninferiority test statistic but uses the upper margin value δ_U in place of $-\delta$.

The test-based confidence limits for the risk difference are computed according to the equivalence test method that you select. If you specify METHOD=WALD with VAR=NULL, or METHOD=FM, separate standard errors are computed for the lower and upper margin tests. In this case, the test-based confidence limits are computed by using the maximum of these two standard errors. These confidence limits have a confidence coefficient of $100(1 - 2\alpha)\%$ (Schuirmann 1999). By default, if you do not specify the ALPHA= option, these are 90% confidence limits. You can compare the test-based confidence limits to the equivalence limits, (δ_L, δ_U).

Barnard's Unconditional Exact Test

The BARNARD option in the EXACT statement provides an unconditional exact test for the risk (proportion) difference for 2×2 tables. The reference set for the unconditional exact test consists of all 2×2 tables that have the same row sums as the observed table (Barnard 1945, 1947, 1949). This differs from the reference set for exact conditional inference, which is restricted to the set of tables that have the same row sums and the same column sums as the observed table. See the sections "Fisher's Exact Test" on page 480 and "Exact Statistics" on page 547 for more information.

The test statistic is the standardized risk difference, which is computed as

$$T = d / \sqrt{p_{\cdot 1}(1 - p_{\cdot 1})(1/n_1 + 1/n_2)}$$

where the risk difference d is defined as the difference between the row 1 and row 2 risks (proportions), $d = (n_{11}/n_1 - n_{21}/n_2)$; n_1 and n_2 are the row 1 and row 2 totals, respectively; and $p_{.1}$ is the overall proportion in column 1, $(n_{11} + n_{21})/n$.

Under the null hypothesis that the risk difference is 0, the joint probability function for a table can be expressed in terms of the table cell frequencies, the row totals, and the unknown parameter π as

$$f(n_{11}, n_{21}; n_1, n_2, \pi) = \binom{n_1}{n_{11}} \binom{n_2}{n_{21}} \pi^{n_{11}+n_{21}} (1-\pi)^{n-n_{11}-n_{21}}$$

where π is the common value of the risk (proportion).

PROC FREQTAB sums the table probabilities over the reference set for those tables where the test statistic is greater than or equal to the observed value of the test statistic. This sum can be expressed as

$$\text{Prob}(\pi) = \sum_{A, T(a) \geq t_0} f(n_{11}, n_{21}; n_1, n_2, \pi)$$

where the set A contains all 2×2 tables with row sums equal to n_1 and n_2, and $T(a)$ denotes the value of the test statistic for table a in A. The sum includes probabilities of those tables for which $(T(a) \geq t_0)$, where t_0 is the value of the test statistic for the observed table.

The sum $\text{Prob}(\pi)$ depends on the unknown value of π. To compute the exact p-value, PROC FREQTAB eliminates the nuisance parameter π by taking the maximum value of $\text{Prob}(\pi)$ over all possible values of π,

$$\text{Prob} = \sup_{(0 \leq \pi \leq 1)} (\text{Prob}(\pi))$$

See Suissa and Shuster (1985) and Mehta and Senchaudhuri (2003).

Common Risk Difference

If you specify the COMMONRISKDIFF option in the TABLES statement, PROC FREQTAB provides estimates, confidence limits, and tests for the common (overall) risk difference for multiway 2×2 tables.

Mantel-Haenszel Confidence Limits and Test

PROC FREQTAB computes the Mantel-Haenszel estimate, confidence limits, and test for the common risk difference by using Mantel-Haenszel stratum weights (Mantel and Haenszel 1959) and the Sato variance estimator (Sato 1989). The Mantel-Haenszel estimate of the common risk difference is

$$\widehat{d}_{\text{MH}} = \sum_h \widehat{d}_h \, w_h$$

where $\widehat{d}_h$ is the risk difference in stratum h and

$$w_h = \frac{n_{h1\cdot} n_{h2\cdot}}{n_h} \Big/ \sum_i \frac{n_{i1\cdot} n_{i2\cdot}}{n_i}$$

is the Mantel-Haenszel weight of stratum h. The column 1 risk difference in stratum (2×2 table) h is computed as

$$\widehat{d}_h = \hat{p}_{h1} - \hat{p}_{h2} = (n_{h11}/n_{h1\cdot}) - (n_{h21}/n_{h2\cdot})$$

where $\hat{p}_{h1}$ is the proportion of row 1 observations that are classified in column 1 and $\hat{p}_{h2}$ is the proportion or row 2 observations that are classified in column 1. The column 2 risk is computed in the same way. For more information, see Agresti (2013, p. 231).

PROC FREQTAB computes the variance of $\widehat{d}_{\mathrm{MH}}$ (Sato 1989) as

$$\widehat{\sigma}^2(\widehat{d}_{\mathrm{MH}}) = \left(\widehat{d}_{\mathrm{MH}} \sum_h P_h + \sum_h Q_h\right) / \left(\sum_h n_{h1\cdot} n_{h2\cdot} / n_h\right)^2$$

where

$$P_h = \left(n_{h1\cdot}^2 n_{h21} - n_{h2\cdot}^2 n_{h11} + n_{h1\cdot} n_{h2\cdot}(n_{h2\cdot} - n_{h1\cdot})/2\right) / n_h^2$$

$$Q_h = \left(n_{h11}(n_{h2\cdot} - n_{h21}) + n_{h21}(n_{h1\cdot} - n_{h11})\right) / 2n_h$$

The $100(1-\alpha)\%$ confidence limits for the common risk difference are

$$\widehat{d}_{\mathrm{MH}} \pm \left(z_{\alpha/2} \times \widehat{\sigma}(\widehat{d}_{\mathrm{MH}})\right)$$

If you specify the COMMONRISKDIFF(TEST=MH) option, PROC FREQTAB provides a Mantel-Haenszel test of the null hypothesis that the common risk difference is 0, which is computed as $z_{\mathrm{MH}} = \widehat{d}_{\mathrm{MH}} / \widehat{\sigma}(\widehat{d}_{\mathrm{MH}})$. The two-sided p-value is $\mathrm{Prob}(|Z| > |z_{\mathrm{MH}}|)$, where Z has a standard normal distribution.

Klingenberg Confidence Limits

Klingenberg confidence limits (Klingenberg 2014) for the Mantel-Haenszel common risk difference are based on inverting a test of homogeneity that uses the null form of the Sato variance estimator (Sato 1989). For performance evaluation of Klingenberg confidence limits, see Fisher (2015) and Klingenberg (2014).

The $100(1-\alpha)\%$ Klingenberg confidence limits for the common risk difference are

$$\widehat{d}_{\mathrm{Mid}} \pm M_{\alpha/2}$$

where M (margin of error) is computed as

$$M_{\alpha/2} = \sqrt{\widehat{d}_{\mathrm{Mid}}^2 - \widehat{d}_{\mathrm{MH}}^2 + z_{\alpha/2}^2 \, (Q/W^2)}$$

and the confidence interval midpoint is computed as

$$\widehat{d}_{\mathrm{Mid}} = \widehat{d}_{\mathrm{MH}} + 0.5 \, z_{\alpha/2}^2 \, (P/W^2)$$

The values P, Q, and W are computed as

$$P = \sum_h P_h$$

$$Q = \sum_h Q_h$$

$$W = \sum_h n_{h1\cdot} \, n_{h2\cdot} \, / \, n_h$$

where h denotes the stratum, and P_h and Q_h are defined in the section "Mantel-Haenszel Confidence Limits and Test" on page 511.

Minimum Risk Confidence Limits and Test

PROC FREQTAB computes the minimum risk estimate, confidence limits, and test for the common risk difference by using the method of Mehrotra and Railkar (2000). The stratum estimates are weighted by minimum risk weights, which minimize the mean square error of the estimate of the common risk difference. Minimum risk weights are designed to improve precision and reduce bias (compared to other weighting strategies) and can minimize the power loss that can occur when underlying assumptions are not met. For more information, see Mehrotra (2001) and Dmitrienko et al. (2005, section 1.3.3).

The minimum risk estimate of the common risk difference is

$$\widehat{d}_{\mathrm{MR}} = \sum_h \widehat{d}_h \; w_h^*$$

where $\widehat{d}_h$ is the risk difference in stratum h and w_h^* is the minimum risk weight of stratum h (which is described in the section "Minimum Risk Weights" on page 514). The variance of $\widehat{d}_{\mathrm{MR}}$ is estimated by

$$\widehat{V}(\widehat{d}_{\mathrm{MR}}) = \sum_h w_h^{*2} \; \widehat{V}_h$$

where $\widehat{V}_h$ (the variance estimate of the stratum h risk difference) is computed as

$$\widehat{V}_h = \hat{p}_{h1}(1 - \hat{p}_{h1})/n_{h1\cdot} \; + \; \hat{p}_{h2}(1 - \hat{p}_{h2})/n_{h2\cdot}$$

The $100(1-\alpha)\%$ minimum risk confidence limits for the common risk difference are

$$\widehat{d}_{\mathrm{MR}} \; \pm \; \left(c \; + \; z_{\alpha/2} \sqrt{\widehat{V}(\widehat{d}_{\mathrm{MR}})} \right)$$

where the continuity correction is

$$c = 0.1875 \; / \; \sum_h (n_{h1\cdot} n_{h2\cdot} / n_h)$$

The continuity correction is applied only when $c < |\widehat{d}_{\mathrm{MR}}|$ (Fleiss, Levin, and Paik 2003). You can remove the continuity correction by specifying the COMMONRISKDIFF(CORRECT=NO) option.

By default, the minimum risk test is computed as

$$z_{\mathrm{MR}} = \left(\widehat{d}_{\mathrm{MR}} \pm c \right) \; / \; \sqrt{\widehat{V}_0(\widehat{d}_{\mathrm{MR}})}$$

The continuity correction c is subtracted from $\widehat{d}_{\mathrm{MR}}$ if $\widehat{d}_{\mathrm{MR}} > 0$ and added to $\widehat{d}_{\mathrm{MR}}$ if $\widehat{d}_{\mathrm{MR}} < 0$. The null variance of the common risk difference is estimated by

$$\widehat{V}_0(\widehat{d}_{\mathrm{MR}}) = \sum_h w_h^{*2} \; \widehat{V}_0(\widehat{d}_h)$$

where $\widehat{V}_0(\widehat{d}_h)$ (an estimate of the variance of the stratum h risk difference under the null hypothesis) is

$$\widehat{V}_0(\widehat{d}_h) = \bar{p}_h(1 - \bar{p}_h) \, (1/n_{h1\cdot} + 1/n_{h2\cdot})$$

and

$$\bar{p}_h = (n_{h1\cdot} \hat{p}_{h1\cdot} + n_{h2\cdot} \hat{p}_{h2\cdot}) \; / \; (n_{h1\cdot} + n_{h2\cdot})$$

The two-sided p-value is $\mathrm{Prob}(|Z| > |z_{\mathrm{MR}}|)$, where Z has a standard normal distribution.

If you specify the VAR=SAMPLE option for COMMONRISKDIFF(TEST=MR), PROC FREQTAB uses the sample variance estimate $\widehat{V}(\widehat{d}_{\mathrm{MR}})$ instead of the null variance estimate $\widehat{V}_0(\widehat{d}_{\mathrm{MR}})$ in the denominator of the test statistic z_{MR}. If you specify the COMMONRISKDIFF(CORRECT=NO) option, the continuity correction is not included in the test statistic.

Minimum Risk Weights The estimate of the minimum risk weight for stratum h is defined by Mehrotra and Railkar (2000) as

$$w_h^* = \frac{\beta_h}{\sum_i \widehat{V}_i^{-1}} - \left(\frac{\alpha_h \widehat{V}_h^{-1}}{\sum_i \widehat{V}_i^{-1} + \sum_i \alpha_i \widehat{d}_i \widehat{V}_i^{-1}} \right) \left(\frac{\sum_i \widehat{d}_i \beta_i}{\sum_i \widehat{V}_i^{-1}} \right)$$

where

$$\alpha_h = \widehat{d}_h \sum_i \widehat{V}_i^{-1} - \sum_i \widehat{d}_i \widehat{V}_i^{-1}$$

$$\beta_h = \widehat{V}_h^{-1} \left(1 + \alpha_h \sum_i f_i \widehat{d}_i \right)$$

and f_h is the fraction in stratum h

$$f_h = n_h \; / \; \sum_i n_h$$

All sums are over the s strata (2×2 tables) in the multiway table request, $\widehat{d}_i$ denotes the risk difference estimate in stratum i, and $\widehat{V}_i$ denotes the sample variance estimate of the risk difference in stratum i.

Summary Score Confidence Limits

PROC FREQTAB computes the summary score estimate of the common risk difference (Agresti 2013, p. 231) by using inverse-variance stratum weights and Miettinen-Nurminen (score) confidence limits for the stratum risk differences. For more information, see the section "Miettinen-Nurminen (Score) Confidence Limits."

The score confidence interval for the risk difference in stratum h can be expressed as $\widehat{d}'_h \pm z_{\alpha/2}\, s'_h$, where $\widehat{d}'_h$ is the midpoint of the score confidence interval and s'_h is the width of the confidence interval divided by $2z_{\alpha/2}$. The summary score estimate of the common risk difference is computed as

$$\widehat{d}_S = \sum_h \widehat{d}'_h \, w'_h$$

where

$$w'_h = (1/{s'_h}^2) \; / \; \sum_i (1/{s'_i}^2)$$

The variance of $\widehat{d}_S$ is computed as

$$\widehat{\sigma}^2(\widehat{d}_S) = 1 \; / \; \sum_h (1/{s'_h}^2)$$

The $100(1-\alpha)\%$ summary score confidence limits for the common risk difference are

$$\widehat{d}_S \pm \left(z_{\alpha/2} \times \widehat{\sigma}(\widehat{d}_S)\right)$$

If you specify the COMMONRISKDIFF(TEST=SCORE) option, PROC FREQTAB provides a summary score test of the null hypothesis that the common risk difference is 0. The test statistic is $z_S = \widehat{d}_S / \widehat{\sigma}(\widehat{d}_S)$ The two-sided p-value is $\mathrm{Prob}(|Z| > |z_S|)$ where Z has a standard normal distribution.

Stratified Newcombe Confidence Limits

PROC FREQTAB computes stratified Newcombe confidence limits for the common risk (proportion) difference by using the method of Yan and Su (2010). The stratified Newcombe confidence limits are constructed from stratified Wilson confidence limits for the common (overall) row proportions. By default, the strata are weighted by Mantel-Haenszel weights; if you specify the COMMONRISKDIFF(CL=NEWCOMBEMR) option, the strata are weighted by minimum risk weights.

PROC FREQTAB first computes individual Wilson confidence limits for the row proportions in each 2×2 table (stratum), as described in the section "Wilson (Score) Confidence Limits" on page 495. These stratum Wilson confidence limits are then combined to form stratified Wilson confidence limits for the overall row proportions by using stratum weights (either Mantel-Haenszel or minimum risk). The confidence levels of the stratum Wilson confidence limits are chosen so that the overall confidence coefficient (for the stratified Wilson confidence limits) is $100(1-\alpha)\%$ (Yan and Su 2010).

Denote the lower and upper stratified Wilson score confidence limits for the common row 1 proportion as L_1 and U_1, respectively, and denote the lower and upper stratified Wilson confidence limits for the common row 2 proportion as L_2 and U_2, respectively. The $100(1-\alpha)\%$ stratified Newcombe confidence limits for the common risk (proportion) difference are

$$\begin{aligned} L &= \widehat{d} - z_{\alpha/2}\sqrt{\lambda_1 L_1(1-L_1) + \lambda_2 U_2(1-U_2)} \\ U &= \widehat{d} + z_{\alpha/2}\sqrt{\lambda_2 L_2(1-L_2) + \lambda_1 U_1(1-U_1)} \end{aligned}$$

where $\widehat{d}$ is the weighted estimate of the common risk difference and

$$\begin{aligned} \lambda_1 &= \sum_h w_h^2 \,/\, n_{h1\cdot} \\ \lambda_2 &= \sum_h w_h^2 \,/\, n_{h2\cdot} \end{aligned}$$

By default, the strata are weighted by Mantel-Haenszel weights, which are defined as

$$w_h = \frac{n_{h1\cdot} n_{h2\cdot}}{n_h} \,/\, \sum_i \frac{n_{i1\cdot} n_{i2\cdot}}{n_i}$$

and the weighted estimate of the common risk difference is $\widehat{d}_{\mathrm{MH}}$. For more information, see the section "Mantel-Haenszel Confidence Limits and Test" on page 511. Optionally, the strata are weighted by minimum risk weights, and the weighted estimate of the common risk difference is $\widehat{d}_{\mathrm{MR}}$. For more information, see the section "Minimum Risk Confidence Limits and Test" on page 513.

When there is a single stratum, the stratified Newcombe confidence interval is equivalent to the (unstratified) Newcombe confidence interval. For more information, see the subsection "Newcombe Confidence Limits" in the section "Confidence Limits for the Risk Difference" on page 502. See also Kim and Won (2013).

Odds Ratio and Relative Risks

Odds Ratio

The odds ratio is a useful measure of association for a variety of study designs. For a retrospective design called a *case-control study*, the odds ratio can be used to estimate the relative risk when the probability of positive response is small (Agresti 2002). In a case-control study, two independent samples are identified based on a binary (yes-no) response variable, and the conditional distribution of a binary explanatory variable is examined within fixed levels of the response variable. For more information, see Stokes, Davis, and Koch (2012), Agresti (2013), and Agresti (2007).

The odds of a positive response (column 1) in row 1 is n_{11}/n_{12}. Similarly, the odds of a positive response in row 2 is n_{21}/n_{22}. The odds ratio is formed as the ratio of the row 1 odds to the row 2 odds. The odds ratio for a 2×2 table is defined as

$$\mathrm{OR} = \frac{n_{11}/n_{12}}{n_{21}/n_{22}} = \frac{n_{11}\, n_{22}}{n_{12}\, n_{21}}$$

The odds ratio can be any nonnegative number. When the row and column variables are independent, the true value of the odds ratio is 1. An odds ratio greater than 1 indicates that the odds of a positive response are higher in row 1 than in row 2. An odds ratio less than 1 indicates that the odds of a positive response are higher in row 2. The strength of association increases as the deviation from 1 increases.

The transformation $G = (\mathrm{OR} - 1)/(\mathrm{OR} + 1)$ transforms the odds ratio to the range (–1,1), where $G = 0$ when OR = 1; $G = -1$ when OR = 0; and G approaches 1 as OR approaches infinity. G is the gamma statistic, which PROC FREQTAB computes when you specify the MEASURES option.

Confidence Limits for the Odds Ratio The following types of confidence limits are available for the odds ratio: exact, exact mid-*p*, likelihood ratio, score, Wald, and Wald modified.

Wald Confidence Limits

The asymptotic Wald confidence limits are based on a log transformation of the odds ratio (Woolf 1955; Haldane 1956). PROC FREQTAB computes the Wald confidence limits as

$$\left(\mathrm{OR} \times \exp(-z\sqrt{v}),\ \ \mathrm{OR} \times \exp(z\sqrt{v}) \right)$$

where

$$v = \mathrm{Var}(\log \mathrm{OR}) = 1/n_{11} + 1/n_{12} + 1/n_{21} + 1/n_{22}$$

and z is the $100(1 - \alpha/2)$th percentile of the standard normal distribution. The confidence level α is determined by the ALPHA= option in the TABLES statement; by default, ALPHA=0.05, which produces 95% confidence limits for the odds ratio. If any of the four cell frequencies are 0, v is undefined and the Wald confidence limits cannot be computed. For more information, see Agresti (2013, p. 70).

Wald Modified Confidence Limits

PROC FREQTAB computes Wald modified confidence limits (Haldane 1956) for the odds ratio by replacing the n_{ij} by $(n_{ij} + 0.5)$ in the estimate and variance as follows:

$$\mathrm{OR}_m = \frac{(n_{11} + 0.5)\ (n_{22} + 0.5)}{(n_{12} + 0.5)\ (n_{21} + 0.5)}$$

$$v = \mathrm{Var}(\log \mathrm{OR}_m) = 1/(n_{11} + 0.5) + 1/(n_{12} + 0.5) + 1/(n_{21} + 0.5) + 1/(n_{22} + 0.5)$$

The modified confidence limits are computed as

$$\left(\mathrm{OR}_m \times \exp(-z\sqrt{v}),\ \ \mathrm{OR}_m \times \exp(z\sqrt{v}) \right)$$

where z is the $100(1-\alpha/2)$th percentile of the standard normal distribution. For more information, see Fleiss, Levin, and Paik (2003) and Agresti (2013).

Score Confidence Limits

Score confidence limits for the odds ratio (Miettinen and Nurminen 1985) are computed by inverting score tests for the odds ratio. A score-based chi-square test statistic for the null hypothesis that the odds ratio is θ can be expressed as

$$Q(\theta) = \{n_{1\cdot}\,(\hat{p}_1 - \tilde{p}_1)\}^2 \,/\, \{n/(n-1)\}\,\{1/\,(n_{1\cdot}\tilde{p}_1(1-\tilde{p}_1)) + 1/\,(n_{2\cdot}\tilde{p}_2(1-\tilde{p}_2))\}^{-1}$$

where $\hat{p}_1$ is the observed row 1 risk ($n_{11}/n_{1\cdot}$), and $\tilde{p}_1$ and $\tilde{p}_2$ are the maximum likelihood estimates of the row 1 and row 2 risks under the restriction that the odds ratio ($n_{11}n_{22}/n_{12}n_{21}$) is θ. For more information, see Miettinen and Nurminen (1985) and Miettinen (1985, chapter 14).

The $100(1-\alpha)\%$ score confidence interval for the odds ratio consists of all values of θ for which the test statistic $Q(\theta)$ falls in the acceptance region,

$$\{\theta : Q(\theta) < \chi^2_{1,\alpha}\}$$

where $\chi^2_{1,\alpha}$ is the $100(1-\alpha)$th percentile of the chi-square distribution with 1 degree of freedom. For more information about score confidence limits, see Agresti (2013).

By default, the score confidence limits include the bias correction factor $n/(n-1)$ in the denominator of $Q(\theta)$ (Miettinen and Nurminen 1985, p. 217). If you specify the CL=SCORE(CORRECT=NO) option, PROC FREQTAB does not include this factor in the computation.

The maximum likelihood estimates of p_1 and p_2, subject to the constraint that the odds ratio is θ, are computed as

$$\tilde{p}_2 = \left(-b + \sqrt{b^2 - 4ac}\right)/2a \quad \text{and} \quad \tilde{p}_1 = \tilde{p}_2\theta/\,(1 + \tilde{p}_2(\theta - 1))$$

where

$$\begin{aligned} a &= n_{2\cdot}(\theta - 1) \\ b &= n_{1\cdot}\theta + n_{2\cdot} - \hat{p}_{\cdot 1}(\theta - 1) \\ c &= -\hat{p}_{\cdot 1} \end{aligned}$$

For more information, see Miettinen and Nurminen (1985, pp. 217–218) and Miettinen (1985, chapter 14).

Likelihood Ratio Confidence Limits

Likelihood ratio (profile likelihood) confidence limits for the odds ratio are computed by inverting likelihood ratio tests. The likelihood ratio test statistic for the null hypothesis that the odds ratio is θ can be expressed as

$$G^2(\theta) \;=\; 2\left(n_{11}\log(\hat{p}_1/\tilde{p}_1) + n_{12}\log((1-\hat{p}_1)/(1-\tilde{p}_1)) + n_{21}\log(\hat{p}_2/\tilde{p}_2) + n_{22}\log((1-\hat{p}_2)/(1-\tilde{p}_2))\right)$$

where $\hat{p}_i$ is the observed row i risk ($n_{11}/n_{1\cdot}$) and $\tilde{p}_i$ is the maximum likelihood estimate of the row i risk under the restriction that the odds ratio is θ. The computation of the maximum likelihood estimates is described in the subsection "Score Confidence Limits" in this section. For more information, see Agresti (2013), Miettinen and Nurminen (1985), and Miettinen (1985, chapter 14).

The $100(1-\alpha)\%$ likelihood ratio confidence interval for the odds ratio consists of all values of θ for which the test statistic $G^2(\theta)$ falls in the acceptance region,

$$\{\theta : G^2(\theta) < \chi^2_{1,\alpha}\}$$

where $\chi^2_{1,\alpha}$ is the $100(1-\alpha)$th percentile of the chi-square distribution with 1 degree of freedom.

Exact Confidence Limits

PROC FREQTAB computes exact confidence limits for the odds ratio by inverting two one-sided (equal-tail) exact tests that are based on the noncentral hypergeometric distribution, where the distribution is conditional on the observed marginal totals of the 2×2 table. The exact confidence limits ϕ_1 and ϕ_2 are the solutions to the equations

$$\begin{aligned} \sum_{i=n_{11}}^{n_{\cdot 1}} f(i : n_{\cdot 1},\ n_{1\cdot},\ n_{2\cdot},\ \phi_1) &= \alpha/2 \\ \sum_{i=0}^{n_{11}} f(i : n_{\cdot 1},\ n_{1\cdot},\ n_{2\cdot},\ \phi_2) &= \alpha/2 \end{aligned}$$

where

$$f(i : n_{\cdot 1},\ n_{1\cdot},\ n_{2\cdot},\ \phi) = \binom{n_{1\cdot}}{i}\binom{n_{2\cdot}}{n_{\cdot 1}-i}\phi^i \ \Big/ \ \sum_{i=0}^{n_{\cdot 1}}\binom{n_{1\cdot}}{i}\binom{n_{2\cdot}}{n_{\cdot 1}-i}\phi^i$$

For more information, see Fleiss, Levin, and Paik (2003), Thomas (1971), and Gart (1971).

Because this is a discrete problem, the confidence coefficient for the exact confidence interval is not exactly $(1-\alpha)$ but is at least $(1-\alpha)$; thus, these confidence limits are conservative. For more information, see Agresti (1992).

When the odds ratio is 0, which occurs when either $n_{11} = 0$ or $n_{22} = 0$, PROC FREQTAB sets the lower exact confidence limit to 0 and determines the upper limit by using the level α (instead of $\alpha/2$). Similarly, when the odds ratio is infinity, which occurs when either $n_{12} = 0$ or $n_{21} = 0$, PROC FREQTAB sets the upper exact confidence limit to infinity and determines the lower limit by using level α.

Exact Mid-p Confidence Limits

PROC FREQTAB computes exact mid-p confidence limits for the odds ratio by inverting two one-sided hypergeometric tests that include mid-p tail areas. The mid-p approach replaces the probability of the observed table with half of that probability in the hypergeometric probability sums, which are described in the subsection "Exact Confidence Limits" in this section. The exact mid-p confidence limits ϕ_1 and ϕ_2 are

the solutions to the equations

$$\begin{aligned}
\sum_{i=n_{11}+1}^{n_{\cdot 1}} \big(f(i : n_{\cdot 1},\, n_{1\cdot},\, n_{2\cdot},\, \phi_1)\big) \;+\; (1/2) f(n_{11} : n_{\cdot 1},\, n_{1\cdot},\, n_{2\cdot},\, \phi_1) &= \alpha/2 \\
\sum_{i=0}^{n_{11}-1} \big(f(i : n_{\cdot 1},\, n_{1\cdot},\, n_{2\cdot},\, \phi_2)\big) \;+\; (1/2) f(n_{11} : n_{\cdot 1},\, n_{1\cdot},\, n_{2\cdot},\, \phi_2) &= \alpha/2
\end{aligned}$$

where

$$f(i : n_{\cdot 1}, n_{1\cdot}, n_{2\cdot}, \phi) = \binom{n_{1\cdot}}{i}\binom{n_{2\cdot}}{n_{\cdot 1}-i}\phi^i \;\Big/\; \sum_{i=0}^{n_{\cdot 1}} \binom{n_{1\cdot}}{i}\binom{n_{2\cdot}}{n_{\cdot 1}-i}\phi^i$$

For more information, see Agresti (2013).

When the odds ratio is 0, which occurs when either $n_{11} = 0$ or $n_{22} = 0$, PROC FREQTAB sets the lower exact confidence limit to 0 and determines the upper limit by using the level α (instead of $\alpha/2$). Similarly, when the odds ratio is infinity, which occurs when either $n_{12} = 0$ or $n_{21} = 0$, PROC FREQTAB sets the upper exact confidence limit to infinity and determines the lower limit by using level α.

Relative Risks

Relative risks are useful measures in *cohort* (prospective) study designs, where two samples are identified based on the presence or absence of an explanatory factor. The two samples are observed in future time for the binary (yes-no) response variable under study. Relative risks are also useful in cross-sectional studies, where two variables are observed simultaneously. For more information, see Stokes, Davis, and Koch (2012) and Agresti (2007).

The relative risk is the ratio of the row 1 risk to the row 2 risk in a 2×2 table. The column 1 risk in row 1 is the proportion of row 1 observations that are classified in column 1, which can be expressed as

$$p_1 = n_{11} \,/\, n_{1\cdot}$$

Similarly, the column 1 risk in row 2 is

$$p_2 = n_{21} \,/\, n_{2\cdot}$$

The column 1 relative risk is defined as

$$R = p_1 \,/\, p_2$$

A relative risk greater than 1 indicates that the probability of positive response is greater in row 1 than in row 2. Similarly, a relative risk less than 1 indicates that the probability of positive response is less in row 1 than in row 2. The strength of association increases as the deviation from 1 increases.

Confidence Limits for the Relative Risk PROC FREQTAB provides the following types of confidence limits for the relative risk: exact unconditional, likelihood ratio, score, Wald, and Wald modified.

Wald Confidence Limits

The asymptotic Wald confidence limits are based on a log transformation of the relative risk. PROC FREQTAB computes the Wald confidence limits for the column 1 relative risk as

$$\left(\hat{r} \times \exp(-z\sqrt{v}),\ \ \hat{r} \times \exp(z\sqrt{v}) \right)$$

where $\hat{r}$ is the observed value of the relative risk, $\hat{p}_1/\hat{p}_2$, and

$$v = \mathrm{Var}(\log(\hat{r})) = 1/n_{11} + 1/n_{21} - 1/n_{1\cdot} - 1/n_{2\cdot}$$

and z is the $100(1-\alpha/2)$th percentile of the standard normal distribution. The confidence level α is determined by the ALPHA= option in the TABLES statement; by default, ALPHA=0.05, which produces 95% confidence limits. If either cell frequency n_{11} or n_{21} is 0, v is undefined and the Wald confidence limits cannot be computed.

PROC FREQTAB computes the confidence limits for the column 2 relative risk in the same way.

Wald Modified Confidence Limits

PROC FREQTAB computes Wald modified confidence limits (Haldane 1956) for the relative risk by replacing the n_{ij} with $(n_{ij}+0.5)$ and the $n_{i\cdot}$ with $(n_{i\cdot}+0.5)$ in the estimate and variance as follows:

$$\hat{r}_m = \frac{(n_{11}+0.5)/(n_{1\cdot}+0.5)}{(n_{21}+0.5)/(n_{2\cdot}+0.5)}$$

$$v = \mathrm{Var}(\log(\hat{r}_m)) = 1/(n_{11}+0.5) + 1/(n_{21}+0.5) - 1/(n_{1\cdot}+0.5) - 1/(n_{2\cdot}+0.5)$$

The confidence limits are computed as

$$\left(\hat{r}_m \times \exp(-z\sqrt{v}),\ \ \hat{r}_m \times \exp(z\sqrt{v}) \right)$$

where z is the $100(1-\alpha/2)$th percentile of the standard normal distribution. For more information, see Fleiss, Levin, and Paik (2003) and Agresti (2013).

Score Confidence Limits

Score confidence limits (Miettinen and Nurminen 1985; Farrington and Manning 1990) are computed by inverting score tests for the relative risk. A score-based chi-square test statistic for the null hypothesis that the relative risk is r_0 can be expressed as

$$Q(r_0) = (\hat{p}_1 - r_0\hat{p}_2)^2 \ / \ \widetilde{\mathrm{Var}}(r_0)$$

where $\hat{p}_1$ and $\hat{p}_2$ are the observed row 1 and row 2 risks (proportions), respectively,

$$\widetilde{\mathrm{Var}}(r_0) = (n/(n-1))\ \left(\tilde{p}_1(1-\tilde{p}_1)/n_{1\cdot} \ + \ {r_0}^2\ \tilde{p}_2(1-\tilde{p}_2)/n_{2\cdot} \right)$$

where $\tilde{p}_1$ and $\tilde{p}_2$ are the maximum likelihood estimates of p_1 and p_2, respectively, under the null hypothesis that the relative risk is r_0. For more information, see Miettinen and Nurminen (1985) and Miettinen (1985, chapter 13).

The $100(1-\alpha)\%$ score confidence interval for the relative risk consists of all values of r_0 for which the test statistic $Q(r_0)$ falls in the acceptance region,

$$\{r_0 : Q(r_0) < \chi^2_{1,\alpha}\}$$

where $\chi^2_{1,\alpha}$ is the $100(1-\alpha)$th percentile of the chi-square distribution with 1 degree of freedom. For more information, see Agresti (2013).

By default, the score confidence limits include the bias correction factor $n/(n-1)$ in the denominator of $Q(r_0)$ (Miettinen and Nurminen 1985, p. 217). If you specify the CL=SCORE(CORRECT=NO) option, PROC FREQTAB does not include this factor in the computation.

The maximum likelihood estimates of p_1 and p_2, subject to the constraint that the relative risk is r_0, are computed as

$$\tilde{p}_1 = \left(-b - \sqrt{b^2 - 4ac}\right)/2a \quad \text{and} \quad \tilde{p}_2 = \tilde{p}_1/r_0$$

where

$$\begin{aligned} a &= 1+\theta \\ b &= -(r_0(1+\theta\hat{p}_2)+\theta+\hat{p}_1) \\ c &= r_0(\hat{p}_1+\theta\hat{p}_2) \\ \theta &= n_{2\cdot}/n_{1\cdot} \end{aligned}$$

For more information, see Farrington and Manning (1990, p. 1454) and Miettinen and Nurminen (1985, p. 217).

Likelihood Ratio Confidence Limits

Likelihood ratio (profile likelihood) confidence limits for the relative risk are computed by inverting likelihood ratio tests. The likelihood ratio test statistic for the null hypothesis that the relative risk ratio is r_0 can be expressed as

$$G^2(r_0) = 2\left(n_{11}\log(\hat{p}_1/\tilde{p}_1) + n_{12}\log((1-\hat{p}_1)/(1-\tilde{p}_1)) + n_{21}\log(\hat{p}_2/\tilde{p}_2) + n_{22}\log((1-\hat{p}_2)/(1-\tilde{p}_2))\right)$$

where $\hat{p}_i$ is the observed row i risk ($n_{i1}/n_{i\cdot}$) and $\tilde{p}_i$ is the maximum likelihood estimate of the row i risk under the restriction that the relative risk is r_0. Expressions for the maximum likelihood estimates $\tilde{p}_1$ and $\tilde{p}_2$ are given in the subsection "Score Confidence Limits" in this section. For more information, see Miettinen and Nurminen (1985) and Miettinen (1985, chapter 13).

The $100(1-\alpha)\%$ likelihood ratio confidence interval for the relative risk consists of all values of r_0 for which the test statistic $G^2(r_0)$ falls in the acceptance region,

$$\{\theta : G^2(r_0) < \chi^2_{1,\alpha}\}$$

where $\chi^2_{1,\alpha}$ is the $100(1-\alpha)$th percentile of the chi-square distribution with 1 degree of freedom.

Exact Unconditional Confidence Limits

If you specify the RELRISK option in the EXACT statement, PROC FREQTAB provides exact unconditional confidence limits for the relative risk. The exact unconditional approach fixes the row margins of the 2×2 table and eliminates the nuisance parameter p_2 by using the maximum p-value (worst-case scenario) over all possible values of p_2 (Santner and Snell 1980). The conditional approach, which is described in the section "Exact Statistics" on page 547, does not apply to the relative risk because of the nuisance parameter (Agresti 1992).

By default, PROC FREQTAB computes the confidence limits by the tail method, which inverts two separate one-sided exact tests of the relative risk, where the tests are based on the score statistic (Chan and Zhang 1999). The size of each one-sided exact test is at most $\alpha/2$, and the confidence coefficient is at least $(1-\alpha)$. If you specify the RELRISK(METHOD=NOSCORE) option in the EXACT statement, PROC FREQTAB computes the confidence limits by inverting two separate one-sided exact tests that are based on the unstandardized relative risk. If you specify the RELRISK(METHOD=SCORE2) option in the EXACT statement, PROC FREQTAB computes the confidence limits by inverting a single two-sided exact test that is based on the score statistic (Agresti and Min 2001).

PROC FREQTAB uses the relative risk score statistic (or the modified form of the unstandardized relative risk) to compute the exact confidence limits as described in the subsection "Exact Unconditional Confidence Limits" in the section "Confidence Limits for the Risk Difference" on page 502.

The score statistic is a less discrete statistic than the unstandardized risk difference and produces less conservative confidence limits (Agresti and Min 2001). For more information, see Santner et al. (2007). The relative risk score statistic (Miettinen and Nurminen 1985; Farrington and Manning 1990) is computed as

$$z(r_0) = (\hat{p}_1 - r_0\hat{p}_2) \;/\; \mathrm{se}(r_0)$$

where

$$\mathrm{se}(r_0) = \sqrt{\tilde{p}_1(1-\tilde{p}_1)/n_{1\cdot} \;+\; {r_0}^2\;\tilde{p}_2(1-\tilde{p}_2)/n_{2\cdot}}$$

where $\tilde{p}_1$ and $\tilde{p}_2$ are the maximum likelihood estimates of p_1 and p_2 under the restriction that the relative risk is r_0. Expressions for the maximum likelihood estimates $\tilde{p}_1$ and $\tilde{p}_2$ are given in the subsection "Score Confidence Limits" in this section. For more information, see Farrington and Manning (1990, p. 1454) and Miettinen and Nurminen (1985, p. 217).

When the confidence limits are computed by using the unstandardized relative risk as the test statistic (METHOD=NOSCORE), PROC FREQTAB uses a modified form of the relative risk to ensure that the statistic is defined when there are zero-frequency table cells. The modified form adds 0.5 to the table cell and row frequencies (Gart and Nam 1988) and is computed as

$$\hat{r}_m = \frac{(n_{11}+0.5) \;/\; (n_{1\cdot}+0.5)}{(n_{21}+0.5) \;/\; (n_{2\cdot}+0.5)}$$

For more information, see the subsection "Wald Modified Confidence Limits" in this section.

Relative Risk Tests PROC FREQTAB provides tests of equality, noninferiority, superiority, and equivalence for the relative risk. The following analysis methods are available: Wald (which is based on a log transformation), Wald modified, score, and likelihood ratio. You can specify the method by using the METHOD= *relrisk-option*; by default, PROC FREQTAB provides Wald tests.

Equality Test

An equality test for the relative risk can be expressed as

$$H_0\colon R = r_0$$

versus the alternative

$$H_a\colon R \neq r_0$$

where $R = p_1/p_2$ denotes the relative risk (for column 1 or column 2) and r_0 denotes the null value. You can specify a null value by using the EQUAL(NULL=) *relrisk-option*; by default, the null value is 1.

The test statistic is computed by the method that you specify; by default, PROC FREQTAB uses the Wald test. For information about test statistic computation, see the subsections "Wald Test," "Wald Modified Test," "Farrington-Manning (Score) Test," and "Likelihood Ratio Test" in this section.

For the Wald and score methods, the test statistics z have standard normal distributions under the null hypothesis. For the likelihood ratio test, the test statistic G^2 has a chi-square distribution with 1 degree of freedom under the null hypothesis.

When the test statistic z is greater than 0, PROC FREQTAB displays the right-sided p-value, which is the probability of a larger value occurring under the null hypothesis. The one-sided p-value can be expressed as

$$P_1 = \begin{cases} \text{Prob}(Z > z) & \text{if } z > 0 \\ \text{Prob}(Z < z) & \text{if } z \leq 0 \end{cases}$$

where Z has a standard normal distribution. The two-sided p-value is computed as $P_2 = 2 \times P_1$.

Noninferiority Test

A noninferiority test for the relative risk can be expressed as

$$H_0: R \leq \delta$$

versus the alternative

$$H_a: R > \delta$$

where $R = p_1/p_2$ denotes the relative risk (for column 1 or column 2) and δ denotes the noninferiority margin (limit). You can specify the margin by using the MARGIN= *relrisk-option*; by default, the noninferiority margin is 0.8. The noninferiority margin for a relative risk test should be less than 1. Rejection of the null hypothesis indicates that the row 1 risk is not inferior to the row 2 risk. For more information, see Chow, Shao, and Wang (2008).

The test statistic z is computed by the method that you specify. For information about test statistic computation, see the subsections "Wald Test," "Wald Modified Test," "Farrington-Manning (Score) Test," and "Likelihood Ratio Test" in this section. The test statistic z is computed by using the noninferiority margin (limit) as the null value of the relative risk. Under the null hypothesis, the test statistic has a standard normal distribution. The p-value for the noninferiority test is the right-sided p-value (the probability that $Z > z$).

As part of the noninferiority analysis, PROC FREQTAB also provides confidence limits for the relative risk. The confidence coefficient is $100(1 - 2\alpha)\%$ (Schuirmann 1999). The confidence level α is determined by the ALPHA= option in the TABLES statement; by default, ALPHA=0.05, which produces 90% confidence limits for the noninferiority analysis. You can compare the confidence limits to the value of the noninferiority limit δ.

Superiority Test

A superiority test for the relative risk can be expressed as

$$H_0: R \leq \delta$$

versus the alternative

$$H_a: R > \delta$$

where $R = p_1/p_2$ denotes the relative risk (for column 1 or column 2) and δ denotes the superiority margin (limit). You can specify the margin by using the MARGIN= *relrisk-option*; by default, the superiority margin is 1.25. The superiority margin for a relative risk test should be greater than 1. Rejection of the null hypothesis indicates that the row 1 risk is superior to the row 2 risk. For more information, see Chow, Shao, and Wang (2008).

The test statistic z is computed by using the superiority margin (limit) as the null value of the relative risk. Under the null hypothesis, the test statistic has a standard normal distribution. The p-value for the superiority test is the right-sided p-value (the probability that $Z > z$).

The computations for the superiority analysis are the same as the computations for the noninferiority analysis, which are described in the subsection "Noninferiority Test" in this section.

Equivalence Test

An equivalence test for the relative risk can be expressed as

$$H_0: R \leq \delta_L \quad \text{or} \quad R \geq \delta_U$$

versus the alternative

$$H_a: \delta_L < R < \delta_U$$

where δ_L is the lower margin and δ_U is the upper margin. Rejection of the null hypothesis indicates that the two risks are equivalent. For more information, see Chow, Shao, and Wang (2008).

You can specify the margins by using the MARGIN= *relrisk-option*; by default, the lower margin is 0.8 and the upper margin is 1.25. If you specify a single margin value, PROC FREQTAB uses this value as the lower margin for the equivalence test and computes the upper margin as the inverse of the lower margin.

PROC FREQTAB computes two one-sided tests (TOST) for equivalence analysis (Schuirmann 1987), which include a right-sided test for the lower margin δ_L and a left-sided test for the upper margin δ_U. The lower test statistic uses the lower margin as the null relative risk value, and the p-value is the right-sided probability ($Z > z_L$). The upper test statistic uses the upper margin as the null value, and the p-value is the left-sided probability ($Z < z_U$). The overall p-value is taken to be the larger of the two p-values for the lower and upper tests.

The test statistics are computed by the method that you specify. For more information about the test statistic computation, see the subsections "Wald Test," "Wald Modified Test," "Farrington-Manning (Score) Test," and "Likelihood Ratio Test" in this section.

As part of the equivalence analysis, PROC FREQTAB also provides confidence limits for the relative risk. The confidence coefficient is $100(1 - 2\alpha)\%$ (Schuirmann 1999). The confidence level α is determined by the ALPHA= option in the TABLES statement; by default, ALPHA=0.05, which produces 90% confidence limits for the equivalence analysis. You can compare the confidence limits to the equivalence limits, which are δ_L and δ_U.

Wald Test

The Wald test statistic (which is based on a log transformation of the relative risk) is computed as $z(r_0) = (\log(\hat{r}) - \log(r_0))/\sqrt{v}$, where $\hat{r}$ is the relative risk estimate $(\hat{p}_1/\hat{p}_2)$, r_0 is the null value of the relative risk, and

$$v = \mathrm{Var}(\log(\hat{r})) = 1/n_{11} + 1/n_{21} - 1/n_{1\cdot} - 1/n_{2\cdot}$$

The null value is determined by the type of test (equality, noninferiority, superiority, or equivalence) and the null or margin values that you specify. The side of the p-value and the interpretation of the test are also determined by the type of test; for more information, see the subsections "Equality Test," "Noninferiority Test," "Superiority Test," and "Equivalence Test" in this section.

Wald Modified Test

The Wald modified test statistic is computed by replacing the n_{ij} with $(n_{ij} + 0.5)$ and the $n_{i\cdot}$ with $(n_{i\cdot} + 0.5)$ in the relative risk estimate and variance. The test statistic is computed as $z(r_0) = (\log(\hat{r}_m) - \log(r_0))/\sqrt{v}$, where r_0 is the null value of the relative risk,

$$\hat{r}_m = \frac{(n_{11} + 0.5)/(n_{1\cdot} + 0.5)}{(n_{21} + 0.5)/(n_{2\cdot} + 0.5)}$$

$$v = \mathrm{Var}(\log(\hat{r}_m)) = 1/(n_{11} + 0.5) + 1/(n_{21} + 0.5) - 1/(n_{1\cdot} + 0.5) - 1/(n_{2\cdot} + 0.5)$$

The null value is determined by the type of test (equality, noninferiority, superiority, or equivalence) and the null or margin values that you specify. The side of the p-value and the interpretation of the test are also determined by the type of test; for more information, see the subsections "Equality Test," "Noninferiority Test," "Superiority Test," and "Equivalence Test" in this section.

Farrington-Manning (Score) Test

The relative risk score test statistic (Miettinen and Nurminen 1985; Farrington and Manning 1990) for the null value r_0 is computed as

$$z(r_0) = (\hat{p}_1 - r_0\hat{p}_2) \;/\; \mathrm{se}(r_0)$$

where

$$\mathrm{se}(r_0) = \sqrt{\tilde{p}_1(1-\tilde{p}_1)/n_{1\cdot} \;+\; r_0{}^2\;\tilde{p}_2(1-\tilde{p}_2)/n_{2\cdot}}$$

where $\tilde{p}_1$ and $\tilde{p}_2$ are the maximum likelihood estimates of p_1 and p_2 under the null value r_0. Expressions for the maximum likelihood estimates $\tilde{p}_1$ and $\tilde{p}_2$ are given in the subsection "Score Confidence Limits" in this section.

The null value is determined by the type of test (equality, noninferiority, superiority, or equivalence) and the null or margin values that you specify. The side of the p-value and the interpretation of the test are also determined by the type of test; for more information, see the subsections "Equality Test," "Noninferiority Test," "Superiority Test," and "Equivalence Test" in this section.

Likelihood Ratio Test

The likelihood ratio statistic for the null relative risk value r_0 is computed as

$$G^2(r_0) \;=\; 2\left(n_{11}\log(\hat{p}_1/\tilde{p}_1) + n_{12}\log((1-\hat{p}_1)/(1-\tilde{p}_1)) + n_{21}\log(\hat{p}_2/\tilde{p}_2) + n_{22}\log((1-\hat{p}_2)/(1-\tilde{p}_2))\right)$$

where $\tilde{p}_1$ and $\tilde{p}_2$ are the maximum likelihood estimates of p_1 and p_2 under the null value r_0. Expressions for the maximum likelihood estimates $\tilde{p}_1$ and $\tilde{p}_2$ are given in the subsection "Score Confidence Limits" in this section. For more information, see Miettinen and Nurminen (1985) and Miettinen (1985, chapter 13).

PROC FREQTAB computes the likelihood ratio test statistic $z(r_0)$ for the noninferiority, superiority, and equivalence tests as $\sqrt{G^2(r_0)}$, where the sign is positive if the estimate is greater than the null value ($\hat{r} \geq r_0$) and negative otherwise ($\hat{r} < r_0$).

The null value is determined by the type of test (equality, noninferiority, superiority, or equivalence) and the null or margin values that you specify. The side of the p-value and the interpretation of the test are also determined by the type of test; for more information, see the subsections "Equality Test," "Noninferiority Test," "Superiority Test," and "Equivalence Test" in this section.

Cochran-Armitage Test for Trend

The TREND option in the TABLES statement provides the Cochran-Armitage test for trend, which tests for trend in binomial proportions across levels of a single factor or covariate. This test is appropriate for a two-way table where one variable has two levels and the other variable is ordinal. The two-level variable represents the response, and the other variable represents an explanatory variable with ordered levels. When the two-way has two columns and R rows, PROC FREQTAB tests for trend across the R levels of the row variable, and the binomial proportion is computed as the proportion of observations in the first column. When the table has two rows and C columns, PROC FREQTAB tests for trend across the C levels of the column variable, and the binomial proportion is computed as the proportion of observations in the first row.

The trend test is based on the regression coefficient for the weighted linear regression of the binomial proportions on the scores of the explanatory variable levels. For more information, see Margolin (1988) and Agresti (2002). If the table has two columns and R rows, the trend test statistic is computed as

$$T = \sum_{i=1}^{R} n_{i1}(R_i - \bar{R}) \; / \; \sqrt{p_{\cdot 1}\,(1 - p_{\cdot 1})\,s^2}$$

where R_i is the score of row i, $\bar{R}$ is the average row score, and

$$s^2 = \sum_{i=1}^{R} n_{i\cdot}(R_i - \bar{R})^2$$

The SCORES= option in the TABLES statement determines the type of row scores used in computing the trend test (and other score-based statistics). By default, SCORES=TABLE. For more information, see the section "Scores" on page 475. For character variables, the table scores for the row variable are the row numbers (for example, 1 for the first row, 2 for the second row, and so on). For numeric variables, the table score for each row is the numeric value of the row level. When you perform the trend test, the explanatory variable might be numeric (for example, dose of a test substance), and the variable values might be appropriate scores. If the explanatory variable has ordinal levels that are not numeric, you can assign meaningful scores to the variable levels. Sometimes equidistant scores, such as the table scores for a character variable, might be appropriate. For more information on choosing scores for the trend test, see Margolin (1988).

The null hypothesis for the Cochran-Armitage test is no trend, which means that the binomial proportion $p_{i1} = n_{i1}/n_{i\cdot}$ is the same for all levels of the explanatory variable. Under the null hypothesis, the trend statistic has an asymptotic standard normal distribution.

PROC FREQTAB computes one-sided and two-sided p-values for the trend test. When the test statistic is greater than its null hypothesis expected value of 0, PROC FREQTAB displays the right-sided p-value, which is the probability of a larger value of the statistic occurring under the null hypothesis. A small right-sided p-value supports the alternative hypothesis of increasing trend in proportions from row 1 to row R. When the test statistic is less than or equal to 0, PROC FREQTAB displays the left-sided p-value. A small left-sided p-value supports the alternative of decreasing trend.

The one-sided p-value for the trend test is computed as

$$P_1 = \begin{cases} \text{Prob}(Z > T) & \text{if } T > 0 \\ \text{Prob}(Z < T) & \text{if } T \leq 0 \end{cases}$$

where Z has a standard normal distribution. The two-sided p-value is computed as

$$P_2 = \text{Prob}(|Z| > |T|)$$

PROC FREQTAB also provides exact p-values for the Cochran-Armitage trend test. You can request the exact test by specifying the TREND option in the EXACT statement. See the section "Exact Statistics" on page 547 for more information.

Jonckheere-Terpstra Test

The JT option in the TABLES statement provides the Jonckheere-Terpstra test, which is a nonparametric test for ordered differences among classes. It tests the null hypothesis that the distribution of the response variable does not differ among classes. It is designed to detect alternatives of ordered class differences, which can be expressed as $\tau_1 \leq \tau_2 \leq \cdots \leq \tau_R$ (or $\tau_1 \geq \tau_2 \geq \cdots \geq \tau_R$), with at least one of the inequalities being strict, where τ_i denotes the effect of class i. For such ordered alternatives, the Jonckheere-Terpstra test can be preferable to tests of more general class difference alternatives, such as the Kruskal–Wallis test (produced by the WILCOXON option in the NPAR1WAY procedure). See Pirie (1983) and Hollander and Wolfe (1999) for more information about the Jonckheere-Terpstra test.

The Jonckheere-Terpstra test is appropriate for a two-way table in which an ordinal column variable represents the response. The row variable, which can be nominal or ordinal, represents the classification variable. The levels of the row variable should be ordered according to the ordering you want the test to detect. The order of variable levels is determined by the ORDER= option in the PROC FREQTAB statement. By default, ORDER=INTERNAL, which orders by unformatted values. For more information about how to order variable levels, see the ORDER= option.

The Jonckheere-Terpstra test statistic is computed by first forming $R(R-1)/2$ Mann-Whitney counts $M_{i,i'}$, where $i < i'$, for pairs of rows in the contingency table,

$$\begin{aligned} M_{i,i'} &= \{ \text{ number of times } X_{i,j} < X_{i',j'}, \quad j = 1,\ldots,n_{i.}; \quad j' = 1,\ldots,n_{i'.} \} \\ &\quad + \tfrac{1}{2} \{ \text{ number of times } X_{i,j} = X_{i',j'}, \quad j = 1,\ldots,n_{i.}; \quad j' = 1,\ldots,n_{i'.} \} \end{aligned}$$

where $X_{i,j}$ is response j in row i. The Jonckheere-Terpstra test statistic is computed as

$$J = \sum_{1 \leq i < i' \leq R} \sum M_{i,i'}$$

This test rejects the null hypothesis of no difference among classes for large values of J. Asymptotic p-values for the Jonckheere-Terpstra test are obtained by using the normal approximation for the distribution of the standardized test statistic. The standardized test statistic is computed as

$$J^* = (J - \mathrm{E}_0(J)) \ / \ \sqrt{\mathrm{Var}_0(J)}$$

where $\mathrm{E}_0(J)$ and $\mathrm{Var}_0(J)$ are the expected value and variance of the test statistic under the null hypothesis,

$$\mathrm{E}_0(J) = \left(n^2 - \sum_i n_{i\cdot}^2 \right) / 4$$

$$\mathrm{Var}_0(J) = A/72 + B/\left(36n(n-1)(n-2)\right) + C/\left(8n(n-1)\right)$$

where

$$A = n(n-1)(2n+5) - \sum_i n_{i\cdot}(n_{i\cdot}-1)(2n_{i\cdot}+5) - \sum_j n_{\cdot j}(n_{\cdot j}-1)(2n_{\cdot j}+5)$$

$$B = \left(\sum_i n_{i\cdot}(n_{i\cdot}-1)(n_{i\cdot}-2) \right) \left(\sum_j n_{\cdot j}(n_{\cdot j}-1)(n_{\cdot j}-2) \right)$$

$$C = \left(\sum_i n_{i\cdot}(n_{i\cdot}-1) \right) \left(\sum_j n_{\cdot j}(n_{\cdot j}-1) \right)$$

PROC FREQTAB computes one-sided and two-sided p-values for the Jonckheere-Terpstra test. When the standardized test statistic is greater than its null hypothesis expected value of 0, PROC FREQTAB displays the right-sided p-value, which is the probability of a larger value of the statistic occurring under the null hypothesis. A small right-sided p-value supports the alternative hypothesis of increasing order from row 1 to row R. When the standardized test statistic is less than or equal to 0, PROC FREQTAB displays the left-sided p-value. A small left-sided p-value supports the alternative of decreasing order from row 1 to row R.

The one-sided p-value for the Jonckheere-Terpstra test, P_1, is computed as

$$P_1 = \begin{cases} \mathrm{Prob}(Z > J^*) & \text{if } J^* > 0 \\ \mathrm{Prob}(Z < J^*) & \text{if } J^* \leq 0 \end{cases}$$

where Z has a standard normal distribution. The two-sided p-value, P_2, is computed as

$$P_2 = \mathrm{Prob}(|Z| > |J^*|)$$

PROC FREQTAB also provides exact p-values for the Jonckheere-Terpstra test. You can request the exact test by specifying the JT option in the EXACT statement. See the section "Exact Statistics" on page 547 for more information.

Tests and Measures of Agreement

When you specify the AGREE option in the TABLES statement, PROC FREQTAB computes tests and measures of agreement for square tables (for which the number of rows equals the number of columns). By default, these statistics include McNemar's test for 2×2 tables, Bowker's symmetry test, the simple kappa coefficient, and the weighted kappa coefficient. For multiple strata (n-way tables, where $n > 2$), the AGREE option provides the overall simple and weighted kappa coefficients, in addition to tests for equal kappas (simple and weighted) among strata. For $2 \times 2 \times \cdots \times 2$ tables, the AGREE option provides Cochran's Q test.

Optionally, PROC FREQTAB provides kappa tests and other agreement statistics. In addition to the asymptotic tests described in this section, PROC FREQTAB provides exact p-values for McNemar's test, the simple kappa coefficient test, and the weighted kappa coefficient test. You can request these exact tests by specifying the corresponding options in the EXACT statement. For more information, see the section "Exact Statistics" on page 547.

The following sections provide the formulas that PROC FREQTAB uses to compute agreement statistics. For information about the use and interpretation of these statistics, see Agresti (2002, 2007); Fleiss, Levin, and Paik (2003), and the other references cited for each statistic.

McNemar's Test

PROC FREQTAB computes McNemar's test (McNemar 1947) for 2×2 tables when you specify the AGREE option. This test is appropriate when you are analyzing data from matched pairs of subjects with a dichotomous (yes-no) response. By default, the null hypothesis for McNemar's test is marginal homogeneity, which can be expressed as $p_{1\cdot} = p_{\cdot 1}$; this is equivalent to a discordant proportion ratio (p_{12}/p_{21}) of 1. The corresponding test statistic is computed as

$$Q_M = (n_{12} - n_{21})^2 \;/\; (n_{12} + n_{21})$$

Under the null hypothesis, Q_M has an asymptotic chi-square distribution with 1 degree of freedom.

Optionally, you can specify the null ratio of discordant proportions (p_{12}/p_{21}) by using the AGREE(MNULLRATIO=) option. When the null ratio is r, McNemar's test is computed as

$$Q_M(r) = (n_{12} - e_{12})^2)/e_{12} \;+\; (n_{21} - e_{21})^2/e_{21}$$

where $e_{12} = D/(1 + 1/r)$, $e_{21} = D/(1 + r)$, and D is the number of discordant pairs, $(n_{12} + n_{21})$. Under the null hypothesis, $Q_M(r)$ has an asymptotic chi-square distribution with 1 degree of freedom.

PROC FREQTAB also computes an exact p-value for McNemar's test when you specify the MCNEM option in the EXACT statement.

Bowker's Symmetry Test

The null hypothesis for Bowker's symmetry test (Bowker 1948) is symmetric table-cell proportions, which can be expressed as $p_{ij} = p_{ji}$ for all off-diagonal pairs of table cells. For 2×2 tables, Bowker's test is identical to McNemar's test; therefore, PROC FREQTAB provides Bowker's test only for square tables that are larger than 2×2.

Bowker's symmetry test is computed as

$$Q_B = \sum_{i<j}\sum (n_{ij} - n_{ji})^2 \;/\; (n_{ij} + n_{ji})$$

For large samples, Q_B has an asymptotic chi-square distribution with $R(R-1)/2$ degrees of freedom under the null hypothesis of symmetry, where R is the dimension of the square, two-way table.

By default, the number of degrees of freedom for this test ($R(R-1)/2$) is the number of off-diagonal table-cell comparisons. You can specify the number of degrees of freedom in the AGREE(DFSYM=) option. Alternatively, you can specify the AGREE(DFSYM=ADJUST) option, which reduces the degrees of freedom by the number of off-diagonal table-cell pairs that have a total frequency of 0. For more information, see Hoenig, Morgan, and Brown (1995).

Exact Symmetry Test When you specify the SYMMETRY option in the EXACT statement, PROC FREQTAB provides an exact symmetry test by using the method of Krauth (1973). This exact test is computed by conditioning on the observed frequency sums of the complementary off-diagonal table-cell pairs ($n_{ij} + n_{ji}$). PROC FREQTAB evaluates the symmetry test statistic for all tables in the reference set, which includes all possible tables in which the frequency sums of the off-diagonal table-cell pairs match the corresponding frequency sums in the observed table. The exact p-value is then computed as the sum of the table probabilities for those tables for which the symmetry test statistic is greater than or equal to the observed test statistic. The table probabilities are computed as products of $R(R-1)/2$ binomial probabilities (which correspond to the off-diagonal table-cell pairs in tables of dimension R) by using the binomial proportion 0.5 under the null hypothesis of symmetry. For more information, see the section "Exact Statistics" on page 547.

Alternatively, you can request a Monte Carlo estimate of the exact p-value by specifying the SYMMETRY option together with the MC *computation-option* in the EXACT statement. The Monte Carlo computation for the exact symmetry test is conditional on the same reference set that the exact test uses (tables in which the frequency sums of the off-diagonal table-cell pairs match the corresponding sums in the observed table). For more information, see the section "Monte Carlo Estimation" on page 550.

Simple Kappa Coefficient

The simple kappa coefficient (Cohen 1960) is a measure of interrater agreement. PROC FREQTAB computes the simple kappa coefficient as

$$\hat{\kappa} = (P_o - P_e) \ / \ (1 - P_e)$$

where $P_o = \sum_i p_{ii}$ and $P_e = \sum_i p_{i.}p_{.i}$. The component P_o is the proportion of observed agreement, and the component P_e represents the proportion of chance-expected agreement.

If the two response variables are viewed as two independent ratings of the n subjects, the kappa coefficient is +1 when there is complete agreement of the raters. When the observed agreement exceeds the chance-expected agreement, the kappa coefficient is positive, and its magnitude reflects the strength of agreement. When the observed agreement is less than the chance-expected agreement, the kappa coefficient is negative. The minimum value of kappa is between –1 and 0, depending on the marginal proportions of the table.

PROC FREQTAB computes the asymptotic variance of the simple kappa coefficient as

$$\mathrm{Var}(\hat{\kappa}) = (A + B - C) \ / \ (1 - P_e)^2 \ n$$

where

$$A = \sum_i p_{ii}\,(1-(p_{i\cdot}+p_{\cdot i})(1-\hat{\kappa}))^2$$

$$B = (1-\hat{\kappa})^2 \sum_{i\neq j}\sum p_{ij}(p_{\cdot i}+p_{j\cdot})^2$$

$$C = (\,\hat{\kappa} - P_e(1-\hat{\kappa})\,)^2$$

For more information, see Fleiss, Cohen, and Everitt (1969).

Confidence limits for the simple kappa coefficient are computed as

$$\hat{\kappa} \pm \left(z_{\alpha/2} \times \sqrt{\mathrm{Var}(\hat{\kappa})} \right)$$

where $z_{\alpha/2}$ is the $100(1-\alpha/2)$th percentile of the standard normal distribution. The value of α is determined by the ALPHA= option; by default ALPHA=0.05, which produces 95% confidence limits.

PROC FREQTAB provides an asymptotic test for the simple kappa coefficient. By default, the null hypothesis value of kappa is 0; alternatively, you can specify a nonzero null value of kappa (by using the AGREE(NULLKAPPA=) option in the TABLES statement). When the null value of kappa is nonzero, PROC FREQTAB computes the test statistic as

$$z = (\hat{\kappa} - \kappa_0) \;/\; \sqrt{\mathrm{Var}(\hat{\kappa})}$$

where κ_0 is the null value that you specify and $\mathrm{Var}(\hat{\kappa})$ is the variance of the kappa coefficient.

When the null value of kappa is 0, PROC FREQTAB computes the test statistic as

$$z = \hat{\kappa} \;/\; \sqrt{\mathrm{Var}_0(\hat{\kappa})}$$

where $\mathrm{Var}_0(\hat{\kappa})$ is the variance of the kappa coefficient under the null hypothesis (that kappa is 0) and is computed as

$$\mathrm{Var}_0(\hat{\kappa}) = \left(P_e + P_e^2 - \sum_i p_{i\cdot}\,p_{\cdot i}\,(p_{i\cdot}+p_{\cdot i}) \right) \;/\; (1-P_e)^2\, n$$

This test statistic has an asymptotic standard normal distribution under the null hypothesis. For more information, see Fleiss, Levin, and Paik (2003).

PROC FREQTAB also provides an exact test for the simple kappa coefficient. You can request the exact test by specifying the KAPPA or AGREE option in the EXACT statement. For more information, see the section "Exact Statistics" on page 547.

Kappa Details When you specify the AGREE(KAPPADETAILS) option, PROC FREQTAB displays the "Kappa Details" table, which includes the observed agreement P_o, chance-expected agreement P_e, maximum kappa, and B_n measure.

The maximum kappa, which is the maximum possible value of the kappa coefficient given the marginal proportions of the two-way table, is computed as

$$\max(\kappa) = (\max(P_o) - P_e) \;/\; (1-P_e)$$

where

$$\max(P_o) = \left(\sum_i \min(n_{i\cdot}, n_{\cdot i}) \right) / n$$

The B_n measure (Bangdiwala 1988; Bangdiwala et al. 2008) is computed as

$$B_n = \left(\sum_i n_{ii}^2 \right) / \left(\sum_i \sum_j n_{i\cdot} n_{\cdot i} \right)$$

For 2×2 tables, the "Kappa Details" table also includes the prevalence index and the bias index. The prevalence index is the absolute difference between the agreement proportions, $|p_{11} - p_{22}|$. The bias index is the absolute difference between the disagreement proportions, $|p_{12} - p_{21}|$. For more information, see Sim and Wright (2005) and Byrt, Bishop, and Carlin (1993).

Weighted Kappa Coefficient

The weighted kappa coefficient is a generalization of the simple kappa coefficient that uses weights to quantify the relative differences between categories. For 2×2 tables, the weighted kappa coefficient is equivalent to the simple kappa coefficient; therefore, PROC FREQTAB displays the weighted kappa coefficient only for tables larger than 2×2. PROC FREQTAB computes the kappa weights from the column scores, by using either Cicchetti-Allison weights or Fleiss-Cohen weights, both of which are described in the section "Kappa Weights" on page 533. The kappa weights w_{ij} are constructed so that $0 \leq w_{ij} < 1$ for all $i \neq j$, $w_{ii} = 1$ for all i, and $w_{ij} = w_{ji}$. The weighted kappa coefficient is computed as

$$\hat{\kappa}_w = \left(P_{o(w)} - P_{e(w)} \right) \; / \; \left(1 - P_{e(w)} \right)$$

where

$$P_{o(w)} = \sum_i \sum_j w_{ij} \, p_{ij}$$

$$P_{e(w)} = \sum_i \sum_j w_{ij} \, p_{i\cdot} p_{\cdot j}$$

The component $P_{o(w)}$ is the proportion of observed (weighted) agreement, and the component $P_{e(w)}$ represents the proportion of chance-expected (weighted) agreement. When you specify the AGREE(WTKAPDETAILS) option, PROC FREQTAB displays these components in the "Weighted Kappa Details" table.

PROC FREQTAB computes the asymptotic variance of the weighted kappa coefficient as

$$\mathrm{Var}(\hat{\kappa}_w) = \left(\sum_i \sum_j p_{ij} \left(w_{ij} - (\overline{w}_{i\cdot} + \overline{w}_{\cdot j})(1 - \hat{\kappa}_w) \right)^2 - \left(\hat{\kappa}_w - P_{e(w)}(1 - \hat{\kappa}_w) \right)^2 \right) / \, (1 - P_{e(w)})^2 \, n$$

where

$$\overline{w}_{i\cdot} = \sum_j p_{\cdot j} \, w_{ij}$$

$$\overline{w}_{\cdot j} = \sum_i p_{i\cdot} w_{ij}$$

For more information, see Fleiss, Cohen, and Everitt (1969).

Confidence limits for the weighted kappa coefficient are computed as

$$\hat{\kappa}_w \pm \left(z_{\alpha/2} \times \sqrt{\mathrm{Var}(\hat{\kappa}_w)} \right)$$

where $z_{\alpha/2}$ is the $100(1-\alpha/2)$th percentile of the standard normal distribution. The value of α is determined by the ALPHA= option; by default ALPHA=0.05, which produces 95% confidence limits.

PROC FREQTAB provides an asymptotic test for the weighted kappa coefficient. By default, the null hypothesis value of weighted kappa is 0; alternatively, you can specify a nonzero null value of weighted kappa (by using the AGREE(NULLWTKAPPA=) option in the TABLES statement). When the null value of weighted kappa is nonzero, PROC FREQTAB computes the test statistic as

$$z = (\hat{\kappa}_w - \kappa_{w(0)}) \; / \; \sqrt{\mathrm{Var}(\hat{\kappa}_w)}$$

where $\kappa_{w(0)}$ is the null value that you specify and $\mathrm{Var}(\hat{\kappa}_w)$ is the variance of the weighted kappa coefficient.

When the null value of weighted kappa is 0, PROC FREQTAB computes the test statistic as

$$z = \hat{\kappa}_w \; / \; \sqrt{\mathrm{Var}_0(\hat{\kappa}_w)}$$

where $\mathrm{Var}_0(\hat{\kappa}_w)$ is the variance of the weighted kappa coefficient under the null hypothesis (that weighted kappa is 0) and is computed as

$$\mathrm{Var}_0(\hat{\kappa}_w) = \left(\sum_i \sum_j p_{i\cdot} p_{\cdot j} \left(w_{ij} - (\overline{w}_{i\cdot} + \overline{w}_{\cdot j}) \right)^2 - P_{e(w)}^2 \right) \; / \; (1 - P_{e(w)})^2 \, n$$

This test statistic has an asymptotic standard normal distribution under the null hypothesis. For more information, see Fleiss, Levin, and Paik (2003).

PROC FREQTAB also provides an exact test for the weighted kappa coefficient. You can request the exact test by specifying the KAPPA or AGREE option in the EXACT statement. For more information, see the section "Exact Statistics" on page 547.

Kappa Weights PROC FREQTAB computes kappa coefficient weights by using the column scores and one of the two available weight types. The column scores are determined by the SCORES= option in the TABLES statement. The two available types of kappa weights are Cicchetti-Allison and Fleiss-Cohen weights. By default, PROC FREQTAB uses Cicchetti-Allison weights. If you specify the AGREE(WT=FC) option, PROC FREQTAB uses Fleiss-Cohen weights to compute the weighted kappa coefficient.

PROC FREQTAB computes Cicchetti-Allison kappa coefficient weights as

$$w_{ij} = 1 - \frac{|C_i - C_j|}{C_C - C_1}$$

where C_i is the score for column i and C is the number of categories or columns. For more information, see Cicchetti and Allison (1971).

The SCORES= option in the TABLES statement determines the type of column scores used to compute the kappa weights (and other score-based statistics). By default, SCORES=TABLE. For more information, see

the section "Scores" on page 475. For numeric variables, table scores are the values of the variable levels. You can assign numeric values to the levels in a way that reflects their level of similarity. For example, suppose you have four levels and order them according to similarity. If you assign them values of 0, 2, 4, and 10, the Cicchetti-Allison kappa weights take the following values: $w_{12} = 0.8$, $w_{13} = 0.6$, $w_{14} = 0$, $w_{23} = 0.8$, $w_{24} = 0.2$, and $w_{34} = 0.4$. Note that when there are only two categories (that is, $C = 2$), the weighted kappa coefficient is identical to the simple kappa coefficient.

If you specify the AGREE(WT=FC) option in the TABLES statement, PROC FREQTAB computes Fleiss-Cohen kappa coefficient weights as

$$w_{ij} = 1 - \frac{(C_i - C_j)^2}{(C_C - C_1)^2}$$

For more information, see Fleiss and Cohen (1973).

For the preceding example, the Fleiss-Cohen kappa weights are $w_{12} = 0.96$, $w_{13} = 0.84$, $w_{14} = 0$, $w_{23} = 0.96$, $w_{24} = 0.36$, and $w_{34} = 0.64$.

Prevalence-Adjusted Bias-Adjusted Kappa

When you specify the AGREE(PABAK) option, PROC FREQTAB provides the prevalence-adjusted bias-adjusted kappa coefficient (PABAK) (Byrt, Bishop, and Carlin 1993). This coefficient is computed as

$$\hat{\kappa}_a = (P_o - 1/R) \ / \ (1 - 1/R)$$

where $P_o = \sum_i p_{ii}$ and R is the dimension of the square, two-way table. The component P_o is the proportion of observed agreement, and the component $1/R$ represents the chance-expected agreement. When the table is 2×2, $\hat{\kappa}_a = 2P_o - 1$. For more information, see Sim and Wright (2005), Xie (2013), and Holley and Guilford (1964).

PROC FREQTAB computes the variance of the prevalence-adjusted bias-adjusted kappa as

$$\mathrm{Var}(\hat{\kappa}_a) = (R/(R-1))^2 \, (P_o(1 - P_o)/n)$$

Confidence limits are computed as

$$\hat{\kappa}_a \ \pm \ \left(z_{\alpha/2} \times \sqrt{\mathrm{Var}(\hat{\kappa}_a)} \right)$$

where $z_{\alpha/2}$ is the $100(1-\alpha/2)$th percentile of the standard normal distribution. The value of α is determined by the ALPHA= option; by default ALPHA=0.05, which produces 95% confidence limits.

AC1 Agreement Coefficient

When you specify the AGREE(AC1) option, PROC FREQTAB provides Gwet's first-order agreement coefficient, AC1 (Gwet 2008). This coefficient is computed as

$$\hat{\gamma} = \left(P_o - P_{e(\gamma)}\right) \ / \ \left(1 - P_{e(\gamma)}\right)$$

where $P_o = \sum_i p_{ii}$, $P_e = \sum_i e_i(1 - e_i)/(R - 1)$, and $e_i = (p_{i\cdot} + p_{\cdot i})/2$ The component P_o is the proportion of observed agreement, and the component $P_{e(\gamma)}$ represents the proportion of chance-expected agreement. For more information, see Xie (2013) and Blood and Spratt (2007).

PROC FREQTAB computes the variance of AC1 as

$$\mathrm{Var}(\hat{\gamma}) = \left(P_o(1 - P_o) - 4(1 - \hat{\gamma})A + 4(1 - \hat{\gamma}^2)B\right) \ / \ n(1 - P_{e(\gamma)})^2$$

where

$$A = \sum_i p_{ii}(1-e_i)/(R-1) - P_o P_{e(\gamma)}$$

$$B = \sum_i \sum_j p_{ij} \left(1-(e_i+e_j)/2\right)^2 /(R-1)^2 - P^2_{e(\gamma)}$$

Confidence limits for AC1 are computed as

$$\hat{\gamma} \pm \left(z_{\alpha/2} \times \sqrt{\mathrm{Var}(\hat{\gamma})} \right)$$

where $z_{\alpha/2}$ is the $100(1-\alpha/2)$th percentile of the standard normal distribution. The value of α is determined by the ALPHA= option; by default ALPHA=0.05, which produces 95% confidence limits.

Overall Kappa Coefficient

When there are multiple strata, PROC FREQTAB combines the stratum-level estimates of kappa into an overall estimate of the supposed common value of kappa. Assume there are q strata, indexed by $h = 1, 2, \ldots, q$, and let $\mathrm{Var}(\hat{\kappa}_h)$ denote the variance of $\hat{\kappa}_h$. The estimate of the overall kappa coefficient is computed as

$$\hat{\kappa}_T = \sum_{h=1}^{q} \frac{\hat{\kappa}_h}{\mathrm{Var}(\hat{\kappa}_h)} \Big/ \sum_{h=1}^{q} \frac{1}{\mathrm{Var}(\hat{\kappa}_h)}$$

For more information, see Fleiss, Levin, and Paik (2003).

PROC FREQTAB computes an estimate of the overall weighted kappa in the same way.

Tests for Equal Kappa Coefficients

When there are multiple strata, the following chi-square statistic tests whether the stratum-level values of kappa are equal:

$$Q_K = \sum_{h=1}^{q} (\hat{\kappa}_h - \hat{\kappa}_T)^2 \Big/ \mathrm{Var}(\hat{\kappa}_h)$$

Under the null hypothesis of equal kappas for the q strata, Q_K has an asymptotic chi-square distribution with q–1 degrees of freedom. See Fleiss, Levin, and Paik (2003) for more information. PROC FREQTAB computes a test for equal weighted kappa coefficients in the same way.

Cochran's Q Test

Cochran's Q is computed for multiway tables in which each variable has two levels ($2 \times 2 \times \cdots \times 2$ tables). Cochran's Q statistic is used to test the homogeneity of the one-dimensional margins. Let m denote the number of variables and N denote the total number of subjects. Cochran's Q statistic is computed as

$$Q_C = m(m-1) \left(\sum_{j=1}^{m} T_j^2 - T^2/m \right) \Big/ \left(mT - \sum_{k=1}^{N} S_k^2 \right)$$

where T_j is the number of positive responses for variable j, T is the total number of positive responses over all variables, and S_k is the number of positive responses for subject k. Under the null hypothesis, Cochran's Q has an asymptotic chi-square distribution with m–1 degrees of freedom. For more information, see Cochran (1950). When there are only two binary response variables (m=2), Cochran's Q simplifies to McNemar's test. When there are more than two response categories, you can test for marginal homogeneity by using the repeated measures capabilities of the CATMOD procedure.

Tables with Zero-Weight Rows or Columns

The AGREE statistics are defined only for square tables, where the number of rows equals the number of columns; if a table is not square, PROC FREQTAB does not compute AGREE statistics for the table. In the kappa statistic framework, where two independent raters assign ratings to each of n subjects, suppose one of the raters does not use all possible r rating levels. If the corresponding table contains r rows but only r–1 columns, the table is not square and PROC FREQTAB does not compute AGREE statistics. To create a square table in this situation, you can use the ZEROS option in the WEIGHT statement, which includes zero-weight observations in the analysis. You can include zero-weight observations in the input data table to represent any rating levels that are not used by a rater, so that the input data table has at least one observation for each possible rater and rating combination. When you use this input data table and specify the ZEROS option, the analysis includes all rating levels (even when all levels are not actually assigned by both raters). The resulting table (of rater 1 by rater 2) is a square table, and AGREE statistics can be computed.

For more information, see the description of the ZEROS option in the WEIGHT statement. By default, PROC FREQTAB does not process observations that have weights of 0 because these observations do not contribute to the total frequency count, and because many of the tests and measures of association are undefined for tables that contain zero-weight rows or columns. However, kappa statistics are defined for tables that contain zero-weight rows or columns, and the ZEROS option enables you to input zero-weight observations and construct the tables needed to compute kappa statistics.

Cochran-Mantel-Haenszel Statistics

The CMH option in the TABLES statement gives a stratified statistical analysis of the relationship between the row and column variables after controlling for the strata variables in a multiway table. For example, for the table request A*B*C*D, the CMH option provides an analysis of the relationship between C and D, after controlling for A and B. The stratified analysis provides a way to adjust for the possible confounding effects of A and B without being forced to estimate parameters for them.

The CMH analysis produces Cochran-Mantel-Haenszel statistics, which include the correlation statistic, the ANOVA (row mean scores) statistic, and the general association statistic. For 2×2 tables, the CMH option also provides Mantel-Haenszel and logit estimates of the common odds ratio and the common relative risks, in addition to the Breslow-Day test for homogeneity of the odds ratios.

Exact statistics are also available for stratified 2×2 tables. If you specify the EQOR option in the EXACT statement, PROC FREQTAB provides Zelen's exact test for equal odds ratios. If you specify the COMOR option in the EXACT statement, PROC FREQTAB provides exact confidence limits for the common odds ratio and an exact test that the common odds ratio equals one.

Let the number of strata be denoted by q, indexing the strata by $h = 1, 2, \ldots, q$. Each stratum contains a contingency table with X representing the row variable and Y representing the column variable. For table h, denote the cell frequency in row i and column j by n_{hij}, with corresponding row and column marginal totals denoted by $n_{hi.}$ and $n_{h.j}$, and the overall stratum total by n_h.

Because the formulas for the Cochran-Mantel-Haenszel statistics are more easily defined in terms of matrices,

the following notation is used. Vectors are presumed to be column vectors unless they are transposed ($'$).

$$\begin{aligned}
\mathbf{n}'_{hi} &= (n_{hi1}, n_{hi2}, \ldots, n_{hiC}) && (1 \times C)\\
\mathbf{n}'_{h} &= (\mathbf{n}'_{h1}, \mathbf{n}'_{h2}, \ldots, \mathbf{n}'_{hR}) && (1 \times RC)\\
p_{hi\cdot} &= n_{hi\cdot} \,/\, n_h && (1 \times 1)\\
p_{h\cdot j} &= n_{h\cdot j} \,/\, n_h && (1 \times 1)\\
\mathbf{P}'_{h*\cdot} &= (p_{h1\cdot}, p_{h2\cdot}, \ldots, p_{hR\cdot}) && (1 \times R)\\
\mathbf{P}'_{h\cdot *} &= (p_{h\cdot 1}, p_{h\cdot 2}, \ldots, p_{h\cdot C}) && (1 \times C)
\end{aligned}$$

Assume that the strata are independent and that the marginal totals of each stratum are fixed. The null hypothesis, H_0, is that there is no association between X and Y in any of the strata. The corresponding model is the multiple hypergeometric; this implies that, under H_0, the expected value and covariance matrix of the frequencies are, respectively,

$$\mathbf{m}_h = \mathbf{E}[\mathbf{n}_h \mid H_0] = n_h(\mathbf{P}_{h\cdot *} \otimes \mathbf{P}_{h*\cdot})$$

$$\mathbf{Var}[\mathbf{n}_h \mid H_0] = c\left((\mathbf{D}_{\mathbf{P}h\cdot *} - \mathbf{P}_{h\cdot *}\mathbf{P}'_{h\cdot *}) \otimes (\mathbf{D}_{\mathbf{P}h*\cdot} - \mathbf{P}_{h*\cdot}\mathbf{P}'_{h*\cdot}) \right)$$

where

$$c = n_h^2 \,/\, (n_h - 1)$$

and where $\otimes$ denotes Kronecker product multiplication and $\mathbf{D}_\mathbf{a}$ is a diagonal matrix with the elements of $\mathbf{a}$ on the main diagonal.

The generalized CMH statistic (Landis, Heyman, and Koch 1978) is defined as

$$Q_{\mathrm{CMH}} = \mathbf{G}'{\mathbf{V}_\mathbf{G}}^{-1}\mathbf{G}$$

where

$$\begin{aligned}
\mathbf{G} &= \sum_h \mathbf{B}_h(\mathbf{n}_h - \mathbf{m}_h)\\
\mathbf{V_G} &= \sum_h \mathbf{B}_h \left(\mathbf{Var}[\mathbf{n}_h \mid H_0]\right) \mathbf{B}'_h
\end{aligned}$$

and where

$$\mathbf{B}_h = \mathbf{C}_h \otimes \mathbf{R}_h$$

is a matrix of fixed constants based on column scores $\mathbf{C}_h$ and row scores $\mathbf{R}_h$. When the null hypothesis is true, the CMH statistic has an asymptotic chi-square distribution with degrees of freedom equal to the rank of $\mathbf{B}_h$. If $\mathbf{V_G}$ is found to be singular, PROC FREQTAB prints a message and sets the value of the CMH statistic to missing.

PROC FREQTAB computes three CMH statistics by using this formula for the generalized CMH statistic, with different row and column score definitions for each statistic. The CMH statistics that PROC FREQTAB computes are the correlation statistic, the ANOVA (row mean scores) statistic, and the general association

statistic. These statistics test the null hypothesis of no association against different alternative hypotheses. The following sections describe the computation of these CMH statistics.

CAUTION: The CMH statistics have low power for detecting an association in which the patterns of association for some of the strata are in the opposite direction of the patterns displayed by other strata. Thus, a nonsignificant CMH statistic suggests either that there is no association or that no pattern of association has enough strength or consistency to dominate any other pattern.

Correlation Statistic

The correlation statistic, popularized by Mantel and Haenszel, has 1 degree of freedom and is known as the Mantel-Haenszel statistic (Mantel and Haenszel 1959; Mantel 1963).

The alternative hypothesis for the correlation statistic is that there is a linear association between X and Y in at least one stratum. If either X or Y does not lie on an ordinal (or interval) scale, this statistic is not meaningful.

To compute the correlation statistic, PROC FREQTAB uses the formula for the generalized CMH statistic with the row and column scores determined by the SCORES= option in the TABLES statement. See the section "Scores" on page 475 for more information about the available score types. The matrix of row scores $\mathbf{R}_h$ has dimension $1 \times R$, and the matrix of column scores $\mathbf{C}_h$ has dimension $1 \times C$.

When there is only one stratum, this CMH statistic reduces to $(n-1)r^2$, where r is the Pearson correlation coefficient between X and Y. When nonparametric (RANK or RIDIT) scores are specified, the statistic reduces to $(n-1)r_s^2$, where r_s is the Spearman rank correlation coefficient between X and Y. When there is more than one stratum, this CMH statistic becomes a stratum-adjusted correlation statistic.

ANOVA (Row Mean Scores) Statistic

The ANOVA statistic can be used only when the column variable Y lies on an ordinal (or interval) scale so that the mean score of Y is meaningful. For the ANOVA statistic, the mean score is computed for each row of the table, and the alternative hypothesis is that, for at least one stratum, the mean scores of the R rows are unequal. In other words, the statistic is sensitive to location differences among the R distributions of Y.

The matrix of column scores $\mathbf{C}_h$ has dimension $1 \times C$, and the column scores are determined by the SCORES= option.

The matrix of row scores $\mathbf{R}_h$ has dimension $(R-1) \times R$ and is created internally by PROC FREQTAB as

$$\mathbf{R}_h = [\mathbf{I}_{R-1}, -\mathbf{J}_{R-1}]$$

where $\mathbf{I}_{R-1}$ is an identity matrix of rank $R-1$ and $\mathbf{J}_{R-1}$ is an $(R-1) \times 1$ vector of ones. This matrix has the effect of forming $R-1$ independent contrasts of the R mean scores.

When there is only one stratum, this CMH statistic is essentially an analysis of variance (ANOVA) statistic in the sense that it is a function of the variance ratio F statistic that would be obtained from a one-way ANOVA on the dependent variable Y. If nonparametric scores are specified in this case, the ANOVA statistic is a Kruskal-Wallis test.

When there is more than one stratum, this CMH statistic corresponds to a stratum-adjusted ANOVA or Kruskal-Wallis test. In the special case where there is one subject per row and one subject per column in the contingency table of each stratum, this CMH statistic is identical to Friedman's chi-square.

General Association Statistic

The alternative hypothesis for the general association statistic is that, for at least one stratum, there is some kind of association between X and Y. This statistic is always interpretable because it does not require an ordinal scale for either X or Y.

For the general association statistic, the matrix $\mathbf{R}_h$ is the same as the one used for the ANOVA statistic. The matrix $\mathbf{C}_h$ is defined similarly as

$$\mathbf{C}_h = [\mathbf{I}_{C-1}, -\mathbf{J}_{C-1}]$$

PROC FREQTAB generates both score matrices internally. When there is only one stratum, the general association CMH statistic reduces to $Q_P(n-1)/n$, where Q_P is the Pearson chi-square statistic. When there is more than one stratum, the CMH statistic becomes a stratum-adjusted Pearson chi-square statistic. Note that a similar adjustment can be made by summing the Pearson chi-squares across the strata. However, the latter statistic requires a large sample size in each stratum to support the resulting chi-square distribution with $q(R-1)(C-1)$ degrees of freedom. The CMH statistic requires only a large overall sample size because it has only $(R-1)(C-1)$ degrees of freedom.

See Cochran (1954); Mantel and Haenszel (1959); Mantel (1963); Birch (1965); Landis, Heyman, and Koch (1978).

Mantel-Fleiss Criterion

If you specify the CMH(MANTELFLEISS) option in the TABLES statement, PROC FREQTAB computes the Mantel-Fleiss criterion for stratified 2×2 tables. The Mantel-Fleiss criterion can be used to assess the validity of the chi-square approximation for the distribution of the Mantel-Haenszel statistic for 2×2 tables. For more information, see Mantel and Fleiss (1980); Mantel and Haenszel (1959); Stokes, Davis, and Koch (2012); Dmitrienko et al. (2005).

The Mantel-Fleiss criterion is computed as

$$\mathrm{MF} = \min\left(\left[\sum_h m_{h11} - \sum_h (n_{h11})_L\right],\ \left[\sum_h (n_{h11})_U - \sum_h m_{h11}\right]\right)$$

where m_{h11} is the expected value of n_{h11} under the hypothesis of no association between the row and column variables in table h, $(n_{h11})_L$ is the minimum possible value of the table cell frequency, and $(n_{h11})_U$ is the maximum possible value,

$$\begin{aligned} m_{h11} &= n_{h1\cdot}\, n_{h\cdot 1} \,/\, n_h \\ (n_{h11})_L &= \max(\,0,\ \ n_{h1\cdot} - n_{h\cdot 2}\,) \\ (n_{h11})_U &= \min(\,n_{h\cdot 1},\ \ n_{h1\cdot}\,) \end{aligned}$$

The Mantel-Fleiss guideline accepts the validity of the Mantel-Haenszel approximation when the value of the criterion is at least 5. When the criterion is less than 5, PROC FREQTAB displays a warning.

Adjusted Odds Ratio and Relative Risk Estimates

The CMH option provides adjusted odds ratio and relative risk estimates for stratified 2×2 tables. For each of these measures, PROC FREQTAB computes a Mantel-Haenszel estimate and a logit estimate. These estimates apply to n-way table requests in the TABLES statement, when the row and column variables both have two levels.

For example, for the table request A*B*C*D, if the row and column variables C and D both have two levels, PROC FREQTAB provides odds ratio and relative risk estimates, adjusting for the confounding variables A and B.

The choice of an appropriate measure depends on the study design. For case-control (retrospective) studies, the odds ratio is appropriate. For cohort (prospective) or cross-sectional studies, the relative risk is appropriate. See the section "Odds Ratio and Relative Risks" on page 516 for more information on these measures.

Throughout this section, z denotes the $100(1-\alpha/2)$th percentile of the standard normal distribution.

Odds Ratio, Case-Control Studies PROC FREQTAB provides Mantel-Haenszel and logit estimates for the common odds ratio for stratified 2×2 tables.

Mantel-Haenszel Estimator

The Mantel-Haenszel estimate of the common odds ratio is computed as

$$\mathrm{OR}_{\mathrm{MH}} = \left(\sum_h n_{h11}\, n_{h22}/n_h \right) \Big/ \left(\sum_h n_{h12}\, n_{h21}/n_h \right)$$

It is always computed unless the denominator is 0. For more information, see Mantel and Haenszel (1959) and Agresti (2002).

To compute confidence limits for the common odds ratio, PROC FREQTAB uses the Robins, Breslow, and Greenland (1986) variance estimate for $\log(\mathrm{OR}_{\mathrm{MH}})$. The $100(1-\alpha/2)\%$ confidence limits for the common odds ratio are

$$\left(\mathrm{OR}_{\mathrm{MH}} \times \exp(-z\hat{\sigma}),\ \ \mathrm{OR}_{\mathrm{MH}} \times \exp(z\hat{\sigma}) \right)$$

where

$$\begin{aligned}
\hat{\sigma}^2 &= \widehat{\mathrm{Var}}\left(\log(\mathrm{OR}_{\mathrm{MH}}) \right) \\
&= \frac{\sum_h (n_{h11}+n_{h22})(n_{h11}\, n_{h22})/n_h^2}{2\left(\sum_h n_{h11}\, n_{h22}/n_h\right)^2} \\
&\quad + \frac{\sum_h [(n_{h11}+n_{h22})(n_{h12}\, n_{h21}) + (n_{h12}+n_{h21})(n_{h11}\, n_{h22})]/n_h^2}{2\left(\sum_h n_{h11}\, n_{h22}/n_h\right)\left(\sum_h n_{h12}\, n_{h21}/n_h\right)} \\
&\quad + \frac{\sum_h (n_{h12}+n_{h21})(n_{h12}\, n_{h21})/n_h^2}{2\left(\sum_h n_{h12}\, n_{h21}/n_h\right)^2}
\end{aligned}$$

Note that the Mantel-Haenszel odds ratio estimator is less sensitive to small n_h than the logit estimator.

Logit Estimator

The adjusted logit estimate of the common odds ratio (Woolf 1955) is computed as

$$\mathrm{OR_L} = \exp\left(\sum_h w_h \log(\mathrm{OR}_h) \;/\; \sum_h w_h\right)$$

and the corresponding $100(1-\alpha)\%$ confidence limits are

$$\left(\mathrm{OR_L} \times \exp\left(-z/\sqrt{\sum_h w_h}\right),\;\; \mathrm{OR_L} \times \exp\left(z/\sqrt{\sum_h w_h}\right)\right)$$

where OR_h is the odds ratio for stratum h, and

$$w_h = 1/\mathrm{Var}(\log(\mathrm{OR}_h))$$

If any table cell frequency in a stratum h is 0, PROC FREQTAB adds 0.5 to each cell of the stratum before computing OR_h and w_h (Haldane 1956) for the logit estimate. The procedure provides a warning when this occurs.

Relative Risks, Cohort Studies PROC FREQTAB provides Mantel-Haenszel and logit estimates of the common relative risks for stratified 2×2 tables.

Mantel-Haenszel Estimator

The Mantel-Haenszel estimate of the common relative risk for column 1 is computed as

$$\mathrm{RR_{MH}} = \left(\sum_h n_{h11}\, n_{h2\cdot} \,/\, n_h\right) \;/\; \left(\sum_h n_{h21}\, n_{h1\cdot} \,/\, n_h\right)$$

It is always computed unless the denominator is 0. For more information, see Mantel and Haenszel (1959) and Agresti (2002).

To compute confidence limits for the common relative risk, PROC FREQTAB uses the Greenland and Robins (1985) variance estimate for $\log(\mathrm{RR_{MH}})$. The $100(1-\alpha/2)\%$ confidence limits for the common relative risk are

$$\left(\,\mathrm{RR_{MH}} \times \exp(-z\hat{\sigma}),\;\; \mathrm{RR_{MH}} \times \exp(z\hat{\sigma})\,\right)$$

where

$$\hat{\sigma}^2 = \widehat{\mathrm{Var}}(\,\log(\mathrm{RR_{MH}})\,) = \frac{\sum_h (n_{h1\cdot}\, n_{h2\cdot}\, n_{h\cdot 1} - n_{h11}\, n_{h21}\, n_h)/n_h^2}{\left(\sum_h n_{h11}\, n_{h2\cdot}/n_h\right)\left(\sum_h n_{h21}\, n_{h1\cdot}/n_h\right)}$$

Logit Estimator

The adjusted logit estimate of the common relative risk for column 1 is computed as

$$\mathrm{RR_L} = \exp\left(\sum_h w_h \log(\mathrm{RR}_h) \;/\; \sum w_h\right)$$

and the corresponding $100(1-\alpha)\%$ confidence limits are

$$\left(\mathrm{RR_L} \times \exp\left(-z \,/\, \sqrt{\sum_h w_h} \right),\ \mathrm{RR_L} \times \exp\left(z \,/\, \sqrt{\sum_h w_h} \right) \right)$$

where RR_h is the column 1 relative risk estimate for stratum h and

$$w_h = 1 \,/\, \mathrm{Var}(\log(\mathrm{RR}_h))$$

If n_{h11} or n_{h21} is 0, PROC FREQTAB adds 0.5 to each cell of the stratum before computing RR_h and w_h for the logit estimate. The procedure prints a warning when this occurs. For more information, see Kleinbaum, Kupper, and Morgenstern (1982, Sections 17.4 and 17.5).

Breslow-Day Test for Homogeneity of the Odds Ratios

When you specify the CMH option, PROC FREQTAB computes the Breslow-Day test for stratified 2×2 tables. It tests the null hypothesis that the odds ratios for the q strata are equal. When the null hypothesis is true, the statistic has approximately a chi-square distribution with q–1 degrees of freedom. See Breslow and Day (1980) and Agresti (2007) for more information.

The Breslow-Day statistic is computed as

$$Q_{\mathrm{BD}} = \sum_h (n_{h11} - \mathrm{E}(n_{h11} \mid \mathrm{OR_{MH}}))^2 \,/\, \mathrm{Var}(n_{h11} \mid \mathrm{OR_{MH}})$$

where E and Var denote expected value and variance, respectively. The summation does not include any table that contains a row or column that has a total frequency of 0. If $\mathrm{OR_{MH}}$ is 0 or undefined, PROC FREQTAB does not compute the statistic and prints a warning message.

For the Breslow-Day test to be valid, the sample size should be relatively large in each stratum, and at least 80% of the expected cell counts should be greater than 5. Note that this is a stricter sample size requirement than the requirement for the Cochran-Mantel-Haenszel test for $q \times 2 \times 2$ tables, in that each stratum sample size (not just the overall sample size) must be relatively large. Even when the Breslow-Day test is valid, it might not be very powerful against certain alternatives, as discussed in Breslow and Day (1980).

If you specify the BDT option, PROC FREQTAB computes the Breslow-Day test with Tarone's adjustment, which subtracts an adjustment factor from Q_{BD} to make the resulting statistic asymptotically chi-square. The Breslow-Day-Tarone statistic is computed as

$$Q_{\mathrm{BDT}} = Q_{\mathrm{BD}} - \left(\sum_h (n_{h11} - \mathrm{E}(n_{h11} \mid \mathrm{OR_{MH}})) \right)^2 \,/\, \sum_h \mathrm{Var}(n_{h11} \mid \mathrm{OR_{MH}})$$

For more information, see Tarone (1985); Jones et al. (1989); Breslow (1996).

Q Test for Homogeneity of Odds Ratios

PROC FREQTAB computes a Q test for homogeneity of odds ratios as

$$Q = \sum_h w_h (\theta_h - \bar{\theta})^2$$

where θ_h is the log odds ratio in stratum h and $\bar{\theta}$ is the logit estimate of the common log odds ratio. The stratum weights w_h are

$$w_h = 1/\mathrm{Var}(\theta_h)$$

where

$$\mathrm{Var}(\theta_h) = 1/n_{h11} + 1/n_{h12} + 1/n_{h21} + 1/n_{h22}$$

If any table cell frequency in a stratum is 0, PROC FREQTAB adds 0.5 to each cell frequency in the stratum before computing θ_h and w_h. For more information, see the sections "Odds Ratio" on page 516 and "Adjusted Odds Ratio and Relative Risk Estimates" on page 540.

Under the null hypothesis of homogeneity, the Q statistic has approximately a chi-square distribution with k–1 degrees of freedom, where k is the number of strata.

I-Square Measure of Heterogeneity

The I-square statistic (Higgins and Thompson 2002) is a measure of heterogeneity among strata for stratified 2×2 tables. I-square is expressed in percentage form and can be interpreted as the proportion of total variability that is due to between-strata variability. For more information, see Higgins et al. (2003) and Thorlund et al. (2012).

PROC FREQTAB computes I-square for the Q test for odds ratios as

$$I^2 = \max(\ 100\% \times (Q - (k-1))/Q,\ 0\ )$$

where k is the number of strata and Q is described in the section "Q Test for Homogeneity of Odds Ratios" on page 542.

PROC FREQTAB computes uncertainty limits for I-square by using the test-based method of Higgins and Thompson (2002). This method constructs confidence limits for H, where $H^2 = Q/(k-1)$. When $Q > k$ or $k = 2$, the standard error of $\log(H)$ is computed as

$$\mathrm{se}_1(\log(H)) = (\ \log(Q) - \log(k-1)\ )\ /\ 2\left(\sqrt{2Q} - \sqrt{2k-3}\right)$$

When $Q \le k$ and $k > 2$, the standard error of $\log(H)$ is computed as

$$\mathrm{se}_0(\log(H)) = \sqrt{\left(\ 1 - (1/3(k-2)^2)\ \right)\ /\ 2(k-2)}$$

The $100(1-\alpha)\%$ confidence limits for H are

$$\left(\ H \times \exp(-z_{\alpha/2} \times \mathrm{se}(\log(H)),\ \ H \times \exp(z_{\alpha/2} \times \mathrm{se}(\log(H))\ \right)$$

The uncertainty limits for I^2 are computed by transforming the confidence limits for H, where $I^2 = 1 - (1/H^2)$.

When I^2 is 0, PROC FREQTAB sets the lower confidence limit to 0 and determines the upper limit by using the level α (instead of $\alpha/2$).

Zelen's Exact Test for Equal Odds Ratios

If you specify the EQOR option in the EXACT statement, PROC FREQTAB computes Zelen's exact test for equal odds ratios for stratified 2×2 tables. Zelen's test is an exact counterpart to the Breslow-Day asymptotic test for equal odds ratios. The reference set for Zelen's test includes all possible $q \times 2 \times 2$ tables with the same row, column, and stratum totals as the observed multiway table and with the same sum of cell (1,1) frequencies as the observed table. The test statistic is the probability of the observed $q \times 2 \times 2$ table conditional on the fixed margins, which is a product of hypergeometric probabilities.

The p-value for Zelen's test is the sum of all table probabilities that are less than or equal to the observed table probability, where the sum is computed over all tables in the reference set determined by the fixed margins and the observed sum of cell (1,1) frequencies. This test is similar to Fisher's exact test for two-way tables. For more information, see Zelen (1971); Hirji (2006); Agresti (1992). PROC FREQTAB computes Zelen's exact test by using the polynomial multiplication algorithm of Hirji et al. (1996).

Exact Confidence Limits for the Common Odds Ratio

If you specify the COMOR option in the EXACT statement, PROC FREQTAB computes exact confidence limits for the common odds ratio for stratified 2×2 tables. This computation assumes that the odds ratio is constant over all the 2×2 tables. Exact confidence limits are constructed from the distribution of $S = \sum_h n_{h11}$, conditional on the marginal totals of the 2×2 tables.

Because this is a discrete problem, the confidence coefficient for these exact confidence limits is not exactly $(1 - \alpha)$ but is at least $(1 - \alpha)$. Thus, these confidence limits are conservative. See Agresti (1992) for more information.

PROC FREQTAB computes exact confidence limits for the common odds ratio by using an algorithm based on Vollset, Hirji, and Elashoff (1991). See also Mehta, Patel, and Gray (1985).

Conditional on the marginal totals of 2×2 table h, let the random variable S_h denote the frequency of table cell (1,1). Given the row totals $n_{h1\cdot}$ and $n_{h2\cdot}$ and column totals $n_{h\cdot 1}$ and $n_{h\cdot 2}$, the lower and upper bounds for S_h are l_h and u_h,

$$\begin{aligned} l_h &= \max\,(\,0,\;\; n_{h1\cdot} - n_{h\cdot 2}\,) \\ u_h &= \min\,(\,n_{h1\cdot},\;\; n_{h\cdot 1}\,) \end{aligned}$$

Let C_{s_h} denote the hypergeometric coefficient,

$$C_{s_h} = \binom{n_{h\cdot 1}}{s_h} \binom{n_{h\cdot 2}}{n_{h1\cdot} - s_h}$$

and let ϕ denote the common odds ratio. Then the conditional distribution of S_h is

$$P(\,S_h = s_h \mid n_{1\cdot},\; n_{\cdot 1},\; n_{\cdot 2}\,) = C_{s_h}\,\phi^{\,s_h} \,/ \sum_{x\,=\,l_h}^{x\,=\,u_h} C_x\,\phi^{\,x}$$

Summing over all the 2×2 tables, $S = \sum_h S_h$, and the lower and upper bounds of S are l and u,

$$l = \sum_h l_h \quad \text{and} \quad u = \sum_h u_h$$

The conditional distribution of the sum S is

$$P(\,S = s \mid n_{h1\cdot},\; n_{h\cdot 1},\; n_{h\cdot 2};\;\; h = 1, \ldots, q\,) = C_s\,\phi^{\,s} \,/ \sum_{x\,=\,l}^{x\,=\,u} C_x\,\phi^{\,x}$$

where

$$C_s = \sum_{s_1 + \cdots + s_q \,=\, s} \left(\prod_h C_{s_h} \right)$$

Let s_0 denote the observed sum of cell (1,1) frequencies over the q tables. The following two equations are solved iteratively for lower and upper confidence limits for the common odds ratio, ϕ_1 and ϕ_2:

$$\sum_{x=s_0}^{x=u} C_x \, {\phi_1}^x \; / \sum_{x=l}^{x=u} C_x \, {\phi_1}^x \;\; = \;\; \alpha/2$$

$$\sum_{x=l}^{x=s_0} C_x \, {\phi_2}^x \; / \sum_{x=l}^{x=u} C_x \, {\phi_2}^x \;\; = \;\; \alpha/2$$

When the observed sum s_0 equals the lower bound l, PROC FREQTAB sets the lower confidence limit to 0 and determines the upper limit with level α. Similarly, when the observed sum s_0 equals the upper bound u, PROC FREQTAB sets the upper confidence limit to infinity and determines the lower limit with level α.

When you specify the COMOR option in the EXACT statement, PROC FREQTAB also computes the exact test that the common odds ratio equals one. Setting $\phi = 1$, the conditional distribution of the sum S under the null hypothesis becomes

$$P_0(\, S = s \mid n_{h1\cdot},\, n_{h\cdot 1},\, n_{h\cdot 2};\;\; h = 1, \ldots, q \,) = C_s \; / \sum_{x=l}^{x=u} C_x$$

The point probability for this exact test is the probability of the observed sum s_0 under the null hypothesis, conditional on the marginals of the stratified 2×2 tables, and is denoted by $P_0(s_0)$. The expected value of S under the null hypothesis is

$$\mathrm{E}_0(S) = \sum_{x=l}^{x=u} x \, C_x \; / \sum_{x=l}^{x=u} C_x$$

The one-sided exact p-value is computed from the conditional distribution as $P_0(S >= s_0)$ or $P_0(S \leq s_0)$, depending on whether the observed sum s_0 is greater or less than $\mathrm{E}_0(S)$,

$$P_1 \;\; = \;\; P_0(\, S >= s_0 \,) = \sum_{x=s_0}^{x=u} C_x \; / \sum_{x=l}^{x=u} C_x \quad \text{if} \;\; s_0 > \mathrm{E}_0(S)$$

$$P_1 \;\; = \;\; P_0(\, S <= s_0 \,) = \sum_{x=l}^{x=s_0} C_x \; / \sum_{x=l}^{x=u} C_x \quad \text{if} \;\; s_0 \leq \mathrm{E}_0(S)$$

PROC FREQTAB computes two-sided p-values for this test according to three different definitions. A two-sided p-value is computed as twice the one-sided p-value, setting the result equal to one if it exceeds one,

$${P_2}^a = 2 \times P_1$$

In addition, a two-sided p-value is computed as the sum of all probabilities less than or equal to the point probability of the observed sum s_0, summing over all possible values of s, $\;l \leq s \leq u$,

$${P_2}^b = \sum_{l \leq s \leq u : P_0(s) \leq P_0(s_0)} P_0(s)$$

Also, a two-sided p-value is computed as the sum of the one-sided p-value and the corresponding area in the opposite tail of the distribution, equidistant from the expected value,

$${P_2}^c = P_0 \; (\, |S - \mathrm{E}_0(S)| \;\geq\; |s_0 - \mathrm{E}_0(S)| \,)$$

Gail-Simon Test for Qualitative Interactions

The GAILSIMON option in the TABLES statement provides the Gail-Simon test for qualitative interaction for stratified 2×2 tables. For more information, see Gail and Simon (1985); Silvapulle (2001); Dmitrienko et al. (2005).

The Gail-Simon test is based on the risk differences in stratified 2×2 tables, where the risk difference is defined as the row 1 risk (proportion in column 1) minus the row 2 risk. For more information, see the section "Risks and Risk Differences" on page 501. By default, PROC FREQTAB uses column 1 risks to compute the Gail-Simon test. If you specify the GAILSIMON(COLUMN=2) option, PROC FREQTAB uses column 2 risks.

PROC FREQTAB computes the Gail-Simon test statistics as described in Gail and Simon (1985),

$$
\begin{aligned}
Q- &= \sum_h (d_h/s_h)^2 \; I(d_h > 0) \\
Q+ &= \sum_h (d_h/s_h)^2 \; I(d_h < 0) \\
Q &= \min\,(Q-,\; Q+)
\end{aligned}
$$

where d_h is the risk difference in table h, s_h is the standard error of the risk difference, and $I(d_h > 0)$ equals 1 if $d_h > 0$ and 0 otherwise. Similarly, $I(d_h < 0)$ equals 1 if $d_h < 0$ and 0 otherwise. The q 2×2 tables (strata) are indexed by $h = 1, 2, \ldots, q$.

The p-values for the Gail-Simon statistics are computed as

$$
\begin{aligned}
P(Q-) &= \sum_h (1 - F_h(Q-)) \; B(h; n = q, p = 0.5) \\
P(Q+) &= \sum_h (1 - F_h(Q+)) \; B(h; n = q, p = 0.5) \\
P(Q) &= \sum_{h=1}^{q-1} (1 - F_h(Q)) \; B(h; n = (q-1), p = 0.5)
\end{aligned}
$$

where $F_h(\cdot)$ is the cumulative chi-square distribution function with h degrees of freedom and $B(h; n, p)$ is the binomial probability function with parameters n and p. The statistic Q tests the null hypothesis of no qualitative interaction. The statistic $Q-$ tests the null hypothesis of positive risk differences. A small p-value for $Q-$ indicates negative differences; similarly, a small p-value for $Q+$ indicates positive risk differences.

Exact Statistics

Exact statistics can be useful in situations where the asymptotic assumptions are not met and therefore the asymptotic p-values might not be close approximations for the true p-values. Standard asymptotic methods involve the assumption that the test statistic follows a particular distribution when the sample size is sufficiently large. When the sample size is not large, asymptotic results might not be valid. Asymptotic results might also be unreliable when the distribution of the data is sparse, skewed, or heavily tied. For more information, see Agresti (2007) and Bishop, Fienberg, and Holland (1975). Exact computations are based on the statistical theory of exact conditional inference for contingency tables, which is reviewed by Agresti (1992).

In addition to the computation of exact p-values, PROC FREQTAB provides the option to estimate exact p-values by Monte Carlo simulation. This can be useful for large problems where exact computations require a substantial amount of time and memory but asymptotic approximations might not be sufficient.

Exact p-values are available for many tests that PROC FREQTAB performs. For one-way tables, PROC FREQTAB provides exact p-values for the binomial proportion test, the chi-square goodness-of-fit test, and the likelihood ratio chi-square test. PROC FREQTAB also provides exact (Clopper-Pearson) confidence limits for the binomial proportion.

For two-way tables, PROC FREQTAB provides exact p-values for the following tests: Pearson chi-square test, likelihood ratio chi-square test, Mantel-Haenszel chi-square test, Fisher's exact test, Jonckheere-Terpstra test, Cochran-Armitage test for trend, and the symmetry test. PROC FREQTAB also computes exact p-values for tests of the following statistics: Pearson correlation coefficient, Spearman correlation coefficient, Kendall's tau-b, Stuart's tau-c, Somers' $D(C|R)$, Somers' $D(R|C)$, simple kappa coefficient, and weighted kappa coefficient.

For 2×2 tables, PROC FREQTAB provides the exact McNemar's test, exact confidence limits for the odds ratio, and Barnard's unconditional exact test for the risk (proportion) difference. PROC FREQTAB also provides exact unconditional confidence limits for the risk (proportion) difference and for the relative risk (ratio of proportions). For stratified 2×2 tables, PROC FREQTAB provides Zelen's exact test for equal odds ratios, exact confidence limits for the common odds ratio, and an exact test for the common odds ratio.

The following sections summarize the exact computational algorithms, define the exact p-values that PROC FREQTAB computes, discuss the computational resource requirements, and describe the Monte Carlo estimation option.

Computational Algorithms

PROC FREQTAB computes exact p-values for general $R \times C$ tables by using the network algorithm, which was developed by Mehta and Patel (1983). This algorithm provides a substantial advantage over direct enumeration, which can be very time-consuming and feasible only for small problems. See Agresti (1992) for a review of algorithms for computation of exact p-values, and see Mehta, Patel, and Tsiatis (1984) and Mehta, Patel, and Senchaudhuri (1991) for information about the performance of the network algorithm.

To implement the network algorithm, PROC FREQTAB defines a reference set from the input data. For most exact tests that PROC FREQTAB provides, the reference set includes all tables that have the same marginal row and column sums as the observed table. Corresponding to the reference set, the network algorithm forms a directed acyclic network consisting of nodes in a number of stages. A path through the network corresponds to a distinct table in the reference set. The distances between nodes are defined so that the total distance of a path through the network is the corresponding value of the test statistic. At each node, the algorithm computes the shortest and longest path distances for all the paths that pass through that node. For statistics

that can be expressed as a linear combination of cell frequencies multiplied by increasing row and column scores, PROC FREQTAB computes shortest and longest path distances by using the algorithm of Agresti, Mehta, and Patel (1990). For statistics of other forms, PROC FREQTAB computes an upper bound for the longest path and a lower bound for the shortest path by following the approach of Valz and Thompson (1994).

The longest and shortest path distances (bounds) for a node are compared to the value of the test statistic to determine whether all paths through the node contribute to the *p*-value, no paths through the node contribute to the *p*-value, or neither of these situations occurs. If all paths through the node contribute, the *p*-value is incremented accordingly, and these paths are eliminated from further analysis. If no paths contribute, these paths are eliminated from further analysis. Otherwise, the algorithm continues to process this node and the associated paths. The algorithm finishes when all nodes have been accounted for.

PROC FREQTAB performs the network algorithm by using full numerical precision to represent all statistics, row and column scores, and other quantities in the computations. Although it is possible to use rounding to improve the speed and memory requirements of the algorithm, PROC FREQTAB does not use rounding because it might reduce the accuracy of the results.

For one-way tables, PROC FREQTAB computes the exact chi-square goodness-of-fit test by the method of Radlow and Alf (1975). PROC FREQTAB generates all possible one-way tables with the observed total sample size and number of categories. For each possible table, PROC FREQTAB compares its chi-square value with the value for the observed table. If the table's chi-square value is greater than or equal to the observed chi-square, PROC FREQTAB increments the exact *p*-value by the probability of that table, which is calculated under the null hypothesis by using the multinomial frequency distribution. By default, the null hypothesis states that all categories have equal proportions. If you specify null hypothesis proportions or frequencies by using the TESTP= or TESTF= option in the TABLES statement, PROC FREQTAB calculates the exact chi-square test based on that null hypothesis.

Other exact computations are described in sections about the individual statistics. For information about the computation of exact confidence limits and tests for the binomial proportion, see the section "Binomial Proportion" on page 491. For information about computation of exact confidence limits for the odds ratio, see the subsection "Exact Confidence Limits" in the section "Confidence Limits for the Odds Ratio" on page 516. For information about other exact computations, see the subsection "Exact Unconditional Confidence Limits" in the section "Confidence Limits for the Risk Difference" on page 502, the subsection "Exact Unconditional Confidence Limits" in the section "Confidence Limits for the Relative Risk" on page 519, and the sections "Exact Symmetry Test" on page 530, "Exact Confidence Limits for the Common Odds Ratio" on page 544 and "Zelen's Exact Test for Equal Odds Ratios" on page 543.

Definition of *p*-Values

For several tests in PROC FREQTAB, the test statistic is nonnegative, and large values of the test statistic indicate a departure from the null hypothesis. Such nondirectional tests include the Pearson chi-square, the likelihood ratio chi-square, the Mantel-Haenszel chi-square, Fisher's exact test for tables larger than 2×2, McNemar's test, the symmetry test, and the one-way chi-square goodness-of-fit test. The exact *p*-value for a nondirectional test is the sum of probabilities for those tables having a test statistic greater than or equal to the value of the observed test statistic.

There are other tests where it might be appropriate to test against either a one-sided or a two-sided alternative hypothesis. For example, when you test the null hypothesis that the true parameter value equals 0 ($T = 0$), the alternative of interest might be one-sided ($T \leq 0$, or $T \geq 0$) or two-sided ($T \neq 0$). Such tests include the Pearson correlation coefficient, Spearman correlation coefficient, Jonckheere-Terpstra test, Cochran-Armitage test for trend, simple kappa coefficient, and weighted kappa coefficient. For these tests, PROC FREQTAB

displays the right-sided *p*-value when the observed value of the test statistic is greater than its expected value. The right-sided *p*-value is the sum of probabilities for those tables for which the test statistic is greater than or equal to the observed test statistic. Otherwise, when the observed test statistic is less than or equal to the expected value, PROC FREQTAB displays the left-sided *p*-value. The left-sided *p*-value is the sum of probabilities for those tables for which the test statistic is less than or equal to the one observed. The one-sided *p*-value P_1 can be expressed as

$$P_1 = \begin{cases} \text{Prob}(\text{ Test Statistic} \geq t\ ) & \text{if } \ t > \text{E}_0(T) \\ \text{Prob}(\text{ Test Statistic} \leq t\ ) & \text{if } \ t \leq \text{E}_0(T) \end{cases}$$

where t is the observed value of the test statistic and $\text{E}_0(T)$ is the expected value of the test statistic under the null hypothesis. PROC FREQTAB computes the two-sided *p*-value as the sum of the one-sided *p*-value and the corresponding area in the opposite tail of the distribution of the statistic, equidistant from the expected value. The two-sided *p*-value P_2 can be expressed as

$$P_2 = \text{Prob}\,(\,|\text{Test Statistic} - \text{E}_0(T)| \geq |t - \text{E}_0(T)|)$$

If you specify the POINT option in the EXACT statement, PROC FREQTAB provides exact point probabilities for the exact tests. The exact point probability is the exact probability that the test statistic equals the observed value.

If you specify the MIDP option in the EXACT statement, PROC FREQTAB provides exact mid-*p*-values. The exact mid *p*-value is defined as the exact *p*-value minus half the exact point probability, which equals the average of $\text{Prob}(\text{Test Statistic} \geq t)$ and $\text{Prob}(\text{Test Statistic} > t)$ for a right-sided test. The exact mid *p*-value is smaller and less conservative than the non-adjusted exact *p*-value. For more information, see Agresti (2013, section 1.1.4) and Hirji (2006, sections 2.5 and 2.11.1).

Computational Resources

PROC FREQTAB uses relatively fast and efficient algorithms for exact computations. These algorithms, together with improvements in computing power, make it feasible to perform exact computations for data where previously only asymptotic methods could be applied. Nevertheless, depending on your available computing resources, exact computations for some very large problems might require a prohibitive amount of time and memory. For such large problems, consider whether exact methods are really needed or whether asymptotic methods might give results that are very close to the exact results while requiring much less computing time and memory. When asymptotic methods might not be sufficient for such large problems, consider using Monte Carlo estimation of exact *p*-values, which is described in the section "Monte Carlo Estimation" on page 550.

There is no formula that can predict in advance how much time and memory are needed to compute an exact *p*-value for a specific data set and test. The time and memory requirements depend on several factors, which include the following: the total number of observations, the number of rows and columns in the table, the particular arrangement of the observations into table cells, and the test to be performed. Generally, larger problems (in terms of total sample size, number of rows, and number of columns) tend to require more time and memory. For a fixed total sample size, time and memory requirements tend to increase as the number of rows and number of columns increase because of the corresponding increase in the number of reference set tables. For a fixed sample size, time and memory requirements also tend to increase as the marginal row and column totals become more homogeneous. For more information, see Agresti, Mehta, and Patel (1990) and Gail and Mantel (1977).

While PROC FREQTAB is computing an exact *p*-value, you can terminate the computation by pressing the system interrupt key sequence and choosing to stop computations. For more information, see the *SAS Companion* for your system. After you terminate an exact computation, PROC FREQTAB completes all other remaining tasks. The procedure reports missing values for any exact *p*-values that were not computed before termination.

To limit the amount of time that PROC FREQTAB uses for exact computations, you can specify the MAXTIME= option in the EXACT statement. This option sets the maximum amount of clock time (in seconds) that PROC FREQTAB can use to compute an exact *p*-value. If PROC FREQTAB does not finish an exact computation in the time that you specify, the procedure terminates the computation and completes the remaining tasks.

Monte Carlo Estimation

When you specify the MC option in the EXACT statement, PROC FREQTAB computes Monte Carlo estimates of exact *p*-values. Monte Carlo estimation can be useful for large problems where exact computations require a substantial amount of time and memory but asymptotic approximations might not be sufficient. Monte Carlo estimates are available for all exact tests that PROC FREQTAB provides except the binomial proportion test and those tests that apply only to 2×2 or $h \times 2 \times 2$ tables.

To describe the precision of a Monte Carlo estimate, PROC FREQTAB provides the asymptotic standard error and $100(1-\alpha)\%$ confidence limits. You can specify the confidence level α in the ALPHA= option in the EXACT statement; by default, ALPHA=0.01, which produces 99% confidence limits.

You can specify the number of Monte Carlo samples by using the N=*n* option in the EXACT statement. By default, PROC FREQTAB uses 10,000 samples to compute a Monte Carlo estimate. To improve the precision of the Monte Carlo estimates, you can specify a larger value of *n*; this increases the computation time because more samples are generated. To reduce the computation time, you can specify a smaller value of *n*.

PROC FREQTAB computes a Monte Carlo estimate of an exact *p*-value by generating a random sample of tables from the reference set for the exact test. For most exact tests that PROC FREQTAB provides, the reference set includes tables that have the same total sample size, row sums, and column sums as the observed table. (For the exact symmetry test, the reference set includes tables that have the same total sample size as the observed table and the same frequency sums of the off-diagonal table cell pairs.)

PROC FREQTAB generates a random sample of tables from the reference set by using the algorithm of Agresti, Wackerly, and Boyett (1979), which generates tables in proportion to their hypergeometric probabilities conditional on the marginal frequencies. For each sample table, PROC FREQTAB computes the value of the test statistic and compares it to the value of the test statistic for the observed table. To estimate a right-sided *p*-value, PROC FREQTAB counts all sample tables for which the test statistic is greater than or equal to the observed test statistic. The estimate of the *p*-value is the number of these tables divided by the total number of sample tables, which can be expressed as

$$\begin{aligned} \hat{p}_{\mathrm{mc}} &= m \,/\, n \\ m &= \text{number of samples for which (test statistic} \geq t_o) \\ n &= \text{total number of samples} \\ t_o &= \text{observed test statistic} \end{aligned}$$

PROC FREQTAB computes estimates of left-sided and two-sided exact *p*-values similarly. For left-sided exact *p*-values, PROC FREQTAB evaluates whether the sample test statistics are less than or equal to the

observed test statistic. For two-sided exact *p*-values, PROC FREQTAB compares sample test statistics to the observed test statistic by using the definition of the two-sided *p*-value (P_2) for the test. For more information, see the section "Definition of *p*-Values" on page 548 and descriptions of the individual tests.

The variable m has a binomial distribution with n trials and success probability p. The asymptotic standard error of the Monte Carlo estimate is

$$\mathrm{se}(\hat{p}_{\mathrm{mc}}) = \sqrt{\hat{p}_{\mathrm{mc}}\,(1-\hat{p}_{\mathrm{mc}})\,/\,(n-1)}$$

PROC FREQTAB constructs asymptotic confidence limits for the exact *p*-value as

$$\hat{p}_{\mathrm{mc}} \pm \left(z_{\alpha/2} \times \mathrm{se}(\hat{p}_{\mathrm{mc}})\right)$$

where $z_{\alpha/2}$ is the $100(1-\alpha/2)$th percentile of the standard normal distribution and the confidence level α is determined by the ALPHA= option in the EXACT statement.

When the Monte Carlo estimate $\hat{p}_{\mathrm{mc}}$ is 0, PROC FREQTAB computes confidence limits for the *p*-value as

$$\left(0,\ \ 1-\alpha^{(1/n)}\right)$$

When the Monte Carlo estimate $\hat{p}_{\mathrm{mc}}$ is 1, PROC FREQTAB computes confidence limits for the *p*-value as

$$\left(\alpha^{(1/n)},\ \ 1\right)$$

Output Data Sets

PROC FREQTAB produces two types of output data sets that you can use with other statistical and reporting procedures. You can request these data sets as follows:

- Specify the OUT= option in a TABLES statement. This creates an output data set that contains frequency or crosstabulation table counts and percentages
- Specify an OUTPUT statement. This creates an output data set that contains statistics.

PROC FREQTAB does not display the output data sets. Use PROC PRINT, PROC REPORT, or any other SAS reporting tool to display an output data set.

In addition to these two output data sets, you can create a SAS data set from any piece of PROC FREQTAB output by using the Output Delivery System. See the section "ODS Table Names" on page 565 for more information.

Contents of the TABLES Statement Output Data Set

The OUT= option in the TABLES statement creates an output data set that contains one observation for each combination of variable values (or table cell) in the last table request. By default, each observation contains the frequency and percentage for the table cell. The output data set includes the following variables:

- BY variables
- table request variables, such as A, B, C, and D in the table request A*B*C*D
- COUNT, which contains the table cell frequency
- PERCENT, which contains the table cell percentage

If you specify the OUTEXPECT option in the TABLES statement for a two-way or multiway table, the output data set also includes expected cell frequencies. If you specify the OUTPCT option for a two-way or multiway table, the output data set also includes row, column, and table percentages. The additional variables are as follows:

- EXPECTED, which contains the expected frequency
- PCT_TABL, which contains the percentage of two-way table frequency, for n-way tables where $n > 2$
- PCT_ROW, which contains the percentage of row frequency
- PCT_COL, which contains the percentage of column frequency

The OUTEXPECT and OUTPCT options have no effect for one-way tables or for tables that you display in list format (which you can request by specifying the LIST option).

If you specify the OUTCUM option in the TABLES statement for a one-way table or for a table in list format, the output data set also includes cumulative frequencies and cumulative percentages. The additional variables are as follows:

- CUM_FREQ, which contains the cumulative frequency
- CUM_PCT, which contains the cumulative percentage

The following PROC FREQTAB statements create an output data set of frequencies and percentages:

```
proc freqtab;
   tables A A*B / out=D;
run;
```

The output data set D contains frequencies and percentages for the table of A by B, which is the last table request listed in the TABLES statement. If A has two levels (1 and 2), B has three levels (1,2, and 3), and no table cell count is 0 or missing, the output data set D includes six observations, one for each combination of A and B levels. The first observation corresponds to A=1 and B=1; the second observation corresponds to A=1 and B=2; and so on. The data set includes the variables COUNT and PERCENT. The value of COUNT is

the number of observations with the given combination of A and B levels. The value of PERCENT is the percentage of the total number of observations with that A and B combination.

When PROC FREQTAB combines different variable values into the same formatted level, the output data set contains the smallest internal value for the formatted level. For example, suppose a variable X has the values 1.1., 1.4, 1.7, 2.1, and 2.3. When you submit the statement

```
format X 1.;
```

in a PROC FREQTAB step, the formatted levels listed in the frequency table for X are 1 and 2. If you create an output data set with the frequency counts, the internal values of the levels of X are 1.1 and 1.7. To report the internal values of X when you display the output data set, use a format of 3.1 for X.

Contents of the OUTPUT Statement Output Data Set

The OUTPUT statement creates a SAS data set that contains statistics computed by PROC FREQTAB. Table 11.6 lists the statistics that can be stored in the output data set. You identify which statistics to include by specifying *output-options*. For more information, see the description of the OUTPUT statement.

If you specify multiple TABLES statements or multiple table requests in a single TABLES statement, the contents of the output data set correspond to the last table request.

For a one-way table or a two-way table, the output data set contains one observation that stores the requested statistics for the table. For a multiway table, the output data set contains an observation for each two-way table (stratum) of the multiway crosstabulation. If you request summary statistics for the multiway table, the output data set also contains an observation that stores the across-strata summary statistics. If you use a BY statement, the output data set contains an observation (for one-way or two-way tables) or set of observations (for multiway tables) for each BY group.

The OUTPUT data set can include the following variables:

- BY variables
- Variables that identify the stratum for multiway tables, such as A and B in the table request A*B*C*D
- Variables that contain the specified statistics

In addition to the specified estimate or test statistic, the output data set includes associated values such as standard errors, confidence limits, *p*-values, and degrees of freedom.

PROC FREQTAB constructs variable names for the statistics in the output data set by enclosing the *output-option* names in underscores. Variable names for the corresponding standard errors, confidence limits, *p*-values, and degrees of freedom are formed by combining the *output-option* names with prefixes that identify the associated values. Table 11.20 lists the prefixes and their descriptions.

Table 11.20 Output Data Set Variable Name Prefixes

Prefix	Description
E_	Asymptotic standard error (ASE)
L_	Lower confidence limit
U_	Upper confidence limit
E0_	Null hypothesis ASE

Table 11.20 *continued*

Prefix	Description
Z_	Standardized value
DF_	Degrees of freedom
P_	*p*-value
P2_	Two-sided *p*-value
PL_	Left-sided *p*-value
PR_	Right-sided *p*-value
XP_	Exact *p*-value
XP2_	Exact two-sided *p*-value
XPL_	Exact left-sided *p*-value
XPR_	Exact right-sided *p*-value
XPT_	Exact point probability
XMP_	Exact mid *p*-value
XL_	Exact lower confidence limit
XU_	Exact upper confidence limit

For example, the PCHI *output-option* in the OUTPUT statement includes the Pearson chi-square test in the output data set. The variable names for the Pearson chi-square statistic, its degrees of freedom, and the corresponding *p*-value are _PCHI_, DF_PCHI, and P_PCHI, respectively.

Displayed Output

Number of Variable Levels Table

If you specify the NLEVELS option in the PROC FREQTAB statement, PROC FREQTAB displays the "Number of Variable Levels" table. This table provides the number of levels for all variables named in the TABLES statements. PROC FREQTAB determines the variable levels by using the formatted variable values. The "Number of Variable Levels" table contains the following information:

- Variable name
- Levels, which is the total number of levels of the variable

One-Way Frequency Tables

PROC FREQTAB displays one-way frequency tables for all one-way table requests in the TABLES statements, unless you specify the NOPRINT option in the PROC FREQTAB statement or the NOPRINT option in the TABLES statement. For a one-way table showing the frequency distribution of a single variable, PROC FREQTAB displays the name of the variable and its values. For each variable value or level, PROC FREQTAB displays the following information:

- Frequency count, which is the number of observations in the level. (The ONEWAY(NOFREQ) option suppresses the frequencies.)

- Test Frequency count, if you specify the CHISQ(TESTF=) option to request a chi-square goodness-of-fit test for specified frequencies. (The ONEWAY(NOFREQ) option suppresses the test frequencies.)
- Expected frequency, if you specify the ONEWAY(EXPECTED) option
- Deviation of the frequency from the expected value, if you specify the ONEWAY(DEVIATION) option
- Pearson Residual, if you specify the ONEWAY(RESIDUAL) option
- Cell Chi-Square, which is the contribution to the Pearson chi-square statistic, if you specify the ONEWAY(CELLCHI2) option
- Percent, which is the percentage of the total number of observations. (The ONEWAY(NOPERCENT) option suppresses the percentages.)
- Test Percent, if you specify the CHISQ(TESTP=) option to request a chi-square goodness-of-fit test for specified percentages. (The ONEWAY(NOPERCENT) option suppresses the test percentages.)
- Cumulative Frequency, which is the accumulated frequency of the current level and all levels that are listed above it in the table. The last cumulative frequency in the table is the total number of nonmissing observations. (Both the ONEWAY(NOCUM) option and the ONEWAY(NOFREQ) option suppress the cumulative frequencies.)
- Cumulative Percent, which is the accumulated percentage of the current level and all levels that are listed above it in the table. (Both the ONEWAY(NOCUM) option and the ONEWAY(NOPERCENT) option suppress the cumulative percentages.)

The one-way table also displays the Frequency Missing, which is the total frequency of missing values.

Statistics for One-Way Frequency Tables

For one-way tables, two statistical options are available in the TABLES statement. The CHISQ option provides a chi-square goodness-of-fit test, and the BINOMIAL option provides binomial proportion statistics and tests. PROC FREQTAB displays the following information, unless you specify the NOPRINT option in the PROC FREQTAB statement:

- If you specify the CHISQ option for a one-way table, PROC FREQTAB provides a chi-square goodness-of-fit test, displaying the Chi-Square statistic, the degrees of freedom (DF), and the probability value (Pr > ChiSq). If you specify the CHISQ option in the EXACT statement, PROC FREQTAB also displays the exact probability value for this test. If you specify the POINT option with the CHISQ option in the EXACT statement, PROC FREQTAB displays the exact point probability for the test statistic. If you specify the MIDP option in the EXACT statement, PROC FREQTAB displays the exact mid *p*-value for the chi-square test.
- If you specify the BINOMIAL option for a one-way table, PROC FREQTAB displays the estimate of the binomial Proportion, which is the proportion of observations in the first class listed in the one-way table. PROC FREQTAB also displays the asymptotic standard error (ASE) and the asymptotic (Wald) and exact (Clopper-Pearson) confidence limits by default. For the binomial proportion test, PROC FREQTAB displays the asymptotic standard error under the null hypothesis (ASE Under H0), the standardized test statistic (Z), and the one-sided and two-sided probability values.

If you specify the BINOMIAL option in the EXACT statement, PROC FREQTAB also displays the exact one-sided and two-sided probability values for this test. If you specify the POINT option with the BINOMIAL option in the EXACT statement, PROC FREQTAB displays the exact point probability for the test. If you specify the MIDP option in the EXACT statement, PROC FREQTAB displays the exact mid *p*-value for the binomial proportion test.

- If you request binomial confidence limits by specifying the BINOMIAL(CL=) option, PROC FREQTAB displays the "Binomial Confidence Limits" table, which includes the Lower and Upper Confidence Limits for each confidence limit Type that you request. In addition to Wald and Clopper-Pearson (Exact) confidence limits, you can request the following confidence limit types for the binomial proportion: Agresti-Coull, Blaker, Jeffreys, Likelihood Ratio, Logit, Mid-p, and Wilson (score).

- If you request a binomial noninferiority or superiority test by specifying the NONINF or SUP *binomial-option*, PROC FREQTAB displays a Noninferiority Analysis or Superiority Analysis table that contains the following information: the binomial Proportion, the test ASE (under H0 or Sample), the test statistic Z, the probability value, the noninferiority or superiority limit, and the test confidence limits. If you specify the BINOMIAL option in the EXACT statement, PROC FREQTAB also provides the exact probability value for the test, and exact test confidence limits.

- If you request a binomial equivalence test by specifying the EQUIV *binomial-option*, PROC FREQTAB displays an Equivalence Analysis table that contains the following information: binomial Proportion and the test ASE (under H0 or Sample). PROC FREQTAB displays two one-sided tests (TOST) for equivalence, which include test statistics (Z) and probability values for the Lower and Upper tests, together with the Overall probability value. PROC FREQTAB also displays the equivalence limits and the test-based confidence limits. If you specify the BINOMIAL option in the EXACT statement, PROC FREQTAB provides exact probability values for the TOST and exact test-based confidence limits.

Two-Way and Multiway Tables

PROC FREQTAB displays all multiway table requests in the TABLES statements, unless you specify the NOPRINT option in the PROC FREQTAB statement or the NOPRINT option in the TABLES statement.

For two-way to multiway crosstabulation tables, the values of the last variable in the table request form the table columns. The values of the next-to-last variable form the rows. Each level (or combination of levels) of the other variables forms one stratum.

There are three ways to display multiway tables in PROC FREQTAB. By default, PROC FREQTAB displays multiway tables as separate two-way crosstabulation tables for each stratum of the multiway table. Also by default, PROC FREQTAB displays these two-way crosstabulation tables in table cell format. Alternatively, if you specify the CROSSLIST option, PROC FREQTAB displays the two-way crosstabulation tables in ODS column format. If you specify the LIST option, PROC FREQTAB displays multiway tables in list format, which presents the entire multiway crosstabulation in a single table.

Crosstabulation Tables

By default, PROC FREQTAB displays two-way crosstabulation tables in table cell format. The row variable values are listed down the side of the table, the column variable values are listed across the top of the table, and each row and column variable level combination forms a table cell.

Each cell of a crosstabulation table can contain the following information:

- Frequency, which is the number of observations in the table cell. (The NOFREQ option suppresses this information.)
- Expected frequency under the hypothesis of independence, if you specify the EXPECTED option
- Deviation of the cell frequency from the expected value, if you specify the DEVIATION option
- Cell Chi-Square, which is the cell's contribution to the total chi-square statistic, if you specify the CELLCHI2 option
- Tot Pct, which is the cell's percentage of the total multiway table frequency, for n-way tables when $n > 2$, if you specify the TOTPCT option
- Percent, which is the cell's percentage of the total (two-way table) frequency. (The NOPERCENT option suppresses this information.)
- Row Pct, or the row percentage, which is the cell's percentage of the total frequency for its row. (The NOROW option suppresses this information.)
- Col Pct, or column percentage, which is the cell's percentage of the total frequency for its column. (The NOCOL option suppresses this information.)
- Cumulative Col%, or cumulative column percentage, if you specify the CUMCOL option

The table also displays the Frequency Missing, which is the total frequency of missing values.

CROSSLIST Tables

If you specify the CROSSLIST option, PROC FREQTAB displays two-way crosstabulation tables in ODS column format. The CROSSLIST column format is different from the default crosstabulation table cell format, but the CROSSLIST table provides the same information (frequencies, percentages, and other statistics) as the default crosstabulation table.

In the CROSSLIST table format, the rows of the display correspond to the crosstabulation table cells, and the columns of the display correspond to descriptive statistics such as frequencies and percentages. Each table cell is identified by the values of its TABLES row and column variable levels, and all column variable levels are listed within each row variable level. The CROSSLIST table also provides row totals, column totals, and overall table totals.

PROC FREQTAB displays the following information in a CROSSLIST table:

- row variable name and values
- column variable name and values
- Frequency, which is the number of observations in the table cell. (The CROSSLIST(NOFREQ) option suppresses the frequencies.)
- Expected cell frequency under the hypothesis of independence, if you specify the CROSSLIST(EXPECTED) option
- Deviation of the observed frequency from the expected value, if you specify the CROSSLIST(DEVIATION) option

- Standardized Residual, if you specify the CROSSLIST(STDRES) option
- Pearson Residual, if you specify the CROSSLIST(PEARSONRES) option
- Cell Chi-Square, which is the cell's contribution to the total chi-square statistic, if you specify the CROSSLIST(CELLCHI2) option
- Total Percent, which is the cell's percentage of the total multiway table frequency, for n-way tables when $n > 2$, if you specify the CROSSLIST(TOTPCT) option
- Percent, which is the cell's percentage of the total (two-way table) frequency. (The CROSSLIST(NOPERCENT) option suppresses the percentages.)
- Row Percent, which is the cell's percentage of the total frequency in its row. (The CROSSLIST(NOROW) option suppresses the row percentages.)
- Column Percent, the cell's percentage of the total frequency in its column. (The CROSSLIST(NOCOL) option suppresses the column percentages.)

The table also displays the Frequency Missing, which is the total frequency of missing values.

LIST Tables

If you specify the LIST option in the TABLES statement, PROC FREQTAB displays two-way and multiway tables by using a list format instead of the default crosstabulation cell format. The list format displays an entire multiway table in a single table instead of separate two-way (stratum) tables. Each row of a table in list format corresponds to a single crosstabulation cell. Tables that are displayed in list format do not include row percentages, column percentages, or optional two-way table information such as expected frequencies and cell chi-squares.

PROC FREQTAB displays the following information in a list table:

- variable names and values
- Frequency, which is the number of observations in the table cell (level). (The LIST(NOFREQ) option suppresses the frequencies.)
- Percent, which is the level's percentage of the total number of observations. (The LIST(NOPERCENT) option suppresses the percentages.)
- Cumulative Frequency, which is the accumulated frequency of the current level and all levels that are listed above it in the table. The last cumulative frequency in the table is the total number of nonmissing observations. (Both the LIST(NOFREQ) option and the LIST(NOCUM) option suppress the cumulative frequencies.)
- Cumulative Percent, which is the accumulated percentage of the current level and all levels that are listed above it in the table. (Both the LIST(NOPERCENT) option and the LIST(NOCUM) option suppress the cumulative percentages.)

The table also displays the Frequency Missing, which is the total frequency of missing values.

Statistics for Two-Way and Multiway Tables

PROC FREQTAB computes statistical tests and measures for crosstabulation tables, depending on which statements and options you specify. You can suppress the display of these results by specifying the NOPRINT option in the PROC FREQTAB statement. With any of the following information, PROC FREQTAB also displays the Sample Size and the Frequency Missing.

- If you specify the SCOROUT option in the TABLES statement, PROC FREQTAB displays the Row Scores and Column Scores that it uses for statistical computations. The Row Scores table displays the row variable values and the Score corresponding to each value. The Column Scores table displays the column variable values and the corresponding Scores. PROC FREQTAB also identifies the score type used to compute the row and column scores. You can specify the score type with the SCORES= option in the TABLES statement.

- If you specify the CHISQ option, PROC FREQTAB displays the following statistics for each two-way table: Pearson Chi-Square, Likelihood Ratio Chi-Square, Continuity-Adjusted Chi-Square (for 2×2 tables), Mantel-Haenszel Chi-Square, the Phi Coefficient, the Contingency Coefficient, and Cramér's V. For each test statistic, PROC FREQTAB also displays the degrees of freedom (DF) and the probability value (Prob).

- If you specify the CHISQ option for 2×2 tables, PROC FREQTAB also displays Fisher's exact test. The test output includes the cell (1,1) frequency (F), the exact left-sided and right-sided probability values, the table probability (P), and the exact two-sided probability value. If you specify the POINT option in the EXACT statement, PROC FREQTAB displays the exact point probability for Fisher's exact test. If you specify the MIDP option in the EXACT statement, PROC FREQTAB displays the Mid p-Value for the test.

- If you specify the FISHER option in the TABLES statement (or, equivalently, the FISHER option in the EXACT statement), PROC FREQTAB displays Fisher's exact test for tables larger than 2×2. The test output includes the table probability (P) and the probability value. If you specify the POINT option in the EXACT statement, PROC FREQTAB displays the exact point probability for Fisher's exact test. If you specify the MIDP option in the EXACT statement, PROC FREQTAB displays the Mid p-Value for the test.

- If you specify the PCHI, LRCHI, or MHCHI option in the EXACT statement, PROC FREQTAB displays the corresponding exact test: Pearson Chi-Square, Likelihood Ratio Chi-Square, or Mantel-Haenszel Chi-Square, respectively. The test output includes the test statistic, the degrees of freedom (DF), and the asymptotic and exact probability values. If you also specify the POINT option in the EXACT statement, PROC FREQTAB displays the point probability for each exact test requested. If you specify the MIDP option in the EXACT statement, PROC FREQTAB displays the exact mid p-value for each test. If you specify the CHISQ option in the EXACT statement, PROC FREQTAB displays exact probability values for all three of these chi-square tests.

- If you specify the MC option in the EXACT statement, PROC FREQTAB displays Monte Carlo estimates for all exact p-values that you request in the EXACT statement. The Monte Carlo output includes the p-value Estimate, its Confidence Limits, the Number of Samples used to compute the Monte Carlo estimate, and the Initial Seed for random number generation.

- If you specify the MEASURES option, PROC FREQTAB displays the following statistics and their asymptotic standard errors (ASE) for each two-way table: Gamma, Kendall's Tau-b, Stuart's Tau-c,

Somers' $D(C|R)$, Somers' $D(R|C)$, Pearson Correlation, Spearman Correlation, Lambda Asymmetric $(C|R)$, Lambda Asymmetric $(R|C)$, Lambda Symmetric, Uncertainty Coefficient $(C|R)$, Uncertainty Coefficient $(R|C)$, and Uncertainty Coefficient Symmetric. If you specify the CL option, PROC FREQTAB also displays confidence limits for these measures.

- If you specify the PLCORR option, PROC FREQTAB displays the polychoric correlation and its asymptotic standard error (ASE). For 2×2 tables, this statistic is known as the tetrachoric correlation (and is labeled as such in the displayed output). If you specify the CL option, PROC FREQTAB also displays confidence limits for the polychoric correlation. If you specify the PLCORR option in the TEST statement, PROC FREQTAB displays the polychoric correlation, asymptotic standard error (ASE), confidence limits, and the following: the standardized test statistic (Z), the corresponding one-sided and two-sided probability values, the likelihood ratio (LR) chi-square, and the probability value (Pr > ChiSq).

- If you specify the GAMMA, KENTB, STUTC, SMDCR, SMDRC, PCORR, or SCORR option in the TEST statement, PROC FREQTAB displays asymptotic tests for Gamma, Kendall's Tau-*b*, Stuart's Tau-*c*, Somers' $D(C|R)$, Somers' $D(R|C)$, the Pearson Correlation, or the Spearman Correlation, respectively. If you specify the MEASURES option in the TEST statement, PROC FREQTAB displays all these asymptotic tests. The test output includes the statistic, its asymptotic standard error (ASE), Confidence Limits, the ASE under the null hypothesis H0, the standardized test statistic (Z), and the one-sided and two-sided probability values.

- If you specify the KENTB, STUTC, SMDCR, SMDRC, PCORR, or SCORR option in the EXACT statement, PROC FREQTAB displays asymptotic and exact tests for the corresponding measure of association: Kendall's Tau-*b*, Stuart's Tau-*c*, Somers' $D(C|R)$, Somers' $D(R|C)$, the Pearson Correlation, or the Spearman Correlation, respectively. The test output includes the correlation, its asymptotic standard error (ASE), Confidence Limits, the ASE under the null hypothesis H0, the standardized test statistic (Z), and the asymptotic and exact one-sided and two-sided probability values. If you also specify the POINT option in the EXACT statement, PROC FREQTAB displays the point probability for each exact test requested. If you specify the MIDP option in the EXACT statement, PROC FREQTAB displays the exact Mid *p*-Value for each test.

- If you specify the SENSPEC option for 2×2 tables, PROC FREQTAB displays the "Sensitivity and Specificity" table. This table displays the Estimate, Standard Error, and Confidence Limits for the following statistics: Sensitivity, Specificity, Accuracy, Positive Predictive Value, and Negative Predictive Value.

- If you specify the RISKDIFF option for 2×2 tables, PROC FREQTAB displays the Column 1 and Column 2 Risk Estimates. For each column, PROC FREQTAB displays the Row 1 Risk, Row 2 Risk, Total Risk, and Risk Difference, together with their asymptotic standard errors (ASE) and Asymptotic Confidence Limits. PROC FREQTAB also displays Exact Confidence Limits for the Row 1 Risk, Row 2 Risk, and Total Risk. If you specify the RISKDIFF option in the EXACT statement, PROC FREQTAB provides unconditional Exact Confidence Limits for the Risk Difference. You can suppress this table by specifying the RISKDIFF(NORISKS) option.

- If you specify the RISKDIFF(CL=) option for 2×2 tables, PROC FREQTAB displays the "Confidence Limits for the Proportion (Risk) Difference" table, which includes the Lower and Upper Confidence Limits for each confidence limit Type that you request (Agresti-Caffo, Exact, Hauck-Anderson, Miettinen-Nurminen, Newcombe, or Wald).

- If you specify the RISKDIFF(NONINF) option for 2 × 2 tables, PROC FREQTAB displays the "Noninferiority Analysis for the Risk Difference" table, which includes the Risk Difference, test ASE, standardized test statistic Z, probability value (Pr > Z), Noninferiority Limit, and (test-based) Confidence Limits.

- If you specify the RISKDIFF(SUP) option for 2 × 2 tables, PROC FREQTAB displays the "Superiority Analysis for the Risk Difference" table, which includes the Risk Difference, test ASE, standardized test statistic Z, probability value (Pr > Z), Superiority Limit, and (test-based) Confidence Limits.

- If you specify the RISKDIFF(EQUIV) option for 2 × 2 tables, PROC FREQTAB displays the "Equivalence Analysis for the Risk Difference" table, which includes the Risk Difference, test ASE, Equivalence Limits, and (test-based) Confidence Limits. PROC FREQTAB also displays the "Two One-Sided Tests (TOST)" table, which includes test statistics (Z) and P-Values for the Lower Margin and Upper Margin tests, along with the Overall P-Value.

- If you specify the RISKDIFF(EQUAL) option for 2 × 2 tables, PROC FREQTAB displays the "Risk Difference Test" table, which includes the Risk Difference, test ASE, standardized test statistic Z, One-sided probability value (Pr > Z or Pr < Z), and Two-sided probability value (Pr > |Z|).

- If you specify the MEASURES option or the RELRISK option for 2 × 2 tables, PROC FREQTAB displays the "Odds Ratio and Relative Risks" table, which includes the following statistics with their confidence limits: Odds Ratio, Relative Risk (Column 1), and Relative Risk (Column 2). If you specify the OR option in the EXACT statement, PROC FREQTAB also displays the "Exact Confidence Limits for the Odds Ratio" table. If you specify the RELRISK option in the EXACT statement, PROC FREQTAB displays the "Exact Confidence Limits for the Relative Risk" table.

- If you specify the OR(CL=) option for 2 × 2 tables, PROC FREQTAB displays the "Confidence Limits for the Odds Ratio" table, which includes the Lower and Upper Confidence Limits for each confidence limit Type that you request (Exact, Mid-p, Likelihood Ratio, Score, Wald, or Wald Modified).

- If you specify the RELRISK(CL=) option for 2 × 2 tables, PROC FREQTAB displays the "Confidence Limits for the Relative Risk" table, which includes the Lower and Upper Confidence Limits for each confidence limit Type that you request (Exact, Likelihood Ratio, Score, Wald, or Wald Modified).

- If you specify the RELRISK(NONINF) option, PROC FREQTAB displays the "Noninferiority Analysis for the Relative Risk" table, which includes the Relative Risk, standardized test statistic Z, probability value (Pr > Z), Noninferiority Limit, and Confidence Limits.

- If you specify the RELRISK(SUP) option, PROC FREQTAB displays the "Superiority Analysis for the Relative Risk" table, which includes the Relative Risk, standardized test statistic Z, probability value (Pr > Z), Superiority Limit, and Confidence Limits.

- If you specify the RELRISK(EQUIV) option, PROC FREQTAB displays the "Equivalence Analysis for the Relative Risk" table, which includes the Relative Risk, Equivalence Limits, and Confidence Limits. PROC FREQTAB also displays the "Two One-Sided Tests(TOST)" table, which includes test statistics (Z) and P-Values for the Lower Margin and Upper Margin tests, along with the Overall P-Value.

- If you specify the RELRISK(EQUAL) option, PROC FREQTAB displays the "Relative Risk Test" table, which includes the Relative Risk, standardized test statistic Z, One-sided probability value (Pr > Z or Pr < Z), and Two-sided probability value (Pr > |Z|).

- If you specify the TREND option, PROC FREQTAB displays the Cochran-Armitage Trend Test for tables that are $2 \times C$ or $R \times 2$. For this test, PROC FREQTAB gives the Statistic (Z) and the one-sided and two-sided probability values. If you specify the TREND option in the EXACT statement, PROC FREQTAB also displays the exact one-sided and two-sided probability values for this test. If you specify the POINT option with the TREND option in the EXACT statement, PROC FREQTAB displays the exact point probability for the test statistic. If you specify the MIDP option in the EXACT statement, PROC FREQTAB displays the exact Mid p-Value for the trend test.

- If you specify the JT option, PROC FREQTAB displays the Jonckheere-Terpstra Test, showing the Statistic (JT), the standardized test statistic (Z), and the one-sided and two-sided probability values. If you specify the JT option in the EXACT statement, PROC FREQTAB also displays the exact one-sided and two-sided probability values for this test. If you specify the POINT option with the JT option in the EXACT statement, PROC FREQTAB displays the exact point probability for the test statistic. If you specify the MIDP option in the EXACT statement, PROC FREQTAB displays the exact Mid p-Value for the Jonckheere-Terpstra test.

- If you specify the AGREE option for a 2×2 table, PROC FREQTAB displays the "McNemar's Test" table. This table includes the McNemar test statistic (chi-square), the degrees of freedom, and the p-value. If you specify the MCNEM option in the EXACT statement, this table also includes the exact p-value. If you specify the POINT option or the MIDP option in the EXACT statement, the "McNemar's Test" table includes the exact point probability or the exact mid p-value, respectively.

- If you specify the AGREE option for a square table of dimension greater than 2, PROC FREQTAB displays the "Symmetry Test" table. This table displays Bowker's symmetry test statistic (chi-square), the degrees of freedom, and the p-value. If you specify the SYMMETRY option in the EXACT statement, this table also includes the exact p-value. If you specify the POINT option or the MIDP option in the EXACT statement, the "Symmetry Test" table includes the exact point probability or the exact mid p-value, respectively.

- The AGREE option also produces the "Kappa Statistics" table, which displays the simple kappa coefficient. If the dimension of the two-way table is greater than 2, the "Kappa Statistics" table includes the weighted kappa coefficient. If you specify the AGREE(AC1) option or the AGREE(PABAK) option, this table includes the AC1 agreement coefficient or the prevalence-adjusted bias-adjusted kappa (PABAK), respectively. The "Kappa Statistics" table displays the standard error and confidence limits for each agreement statistic.

- If you specify the AGREE(KAPPADETAILS) option, PROC FREQTAB displays the "Kappa Details" table, which includes the observed agreement, the chance-expected agreement, the maximum kappa, and the B_N measure. For 2×2 tables, the "Kappa Details" table also includes the prevalence index and the bias index.

- If you specify the AGREE(WTKAPPADETAILS) or AGREE(KAPPADETAILS) option for a square table of dimension greater than 2, PROC FREQTAB produces the "Weighted Kappa Details" table, which displays the observed agreement and the chance-expected agreement components of the weighted kappa coefficient.

- If you specify the AGREE(PRINTKWTS) option for a square table of dimension greater than 2, PROC FREQTAB displays the matrix of agreement weights in the "Kappa Coefficient Weights" table.

- If you request a simple kappa coefficient test, PROC FREQTAB produces the "Kappa Test" table. You can request this test by specifying the KAPPA option in the TEST statement, the KAPPA option in the

EXACT statement, or the AGREE(NULLKAPPA=) option in the TABLES statement. The "Kappa Test" table displays the kappa coefficient, null test value, standard error (when the null value is 0), standardized test statistic (Z), and one-sided and two-sided *p*-values.

If you request an exact test (by specifying the KAPPA option in the EXACT statement), the "Kappa Test" table also includes the exact one-sided and two-sided *p*-values. If you specify the POINT option or the MIDP option in the EXACT statement, the "Kappa Test" table includes the point probability or the exact mid *p*-value, respectively.

- If you request a weighted kappa coefficient test for a square table of dimension greater than 2, PROC FREQTAB produces the "Weighted Kappa Test" table. You can request this test by specifying the WTKAPPA option in the TEST statement, the WTKAPPA option in the EXACT statement, or the AGREE(NULLWTKAPPA=) option in the TABLES statement. The "Weighted Kappa Test" table displays the weighted kappa coefficient, null test value, standard error (when the null value is 0), standardized test statistic (Z), and one-sided and two-sided *p*-values.

  If you request an exact test (by specifying the WTKAPPA option in the EXACT statement), the "Weighted Kappa Test" table also includes the exact one-sided and two-sided *p*-values. If you specify the POINT option or the MIDP option in the EXACT statement, the "Weighted Kappa Test" table includes the point probability or the exact mid *p*-value, respectively.

- If you specify the AGREE option for a multiway square table, PROC FREQTAB displays the "Overall Kappa Coefficients" table, which includes the overall simple kappa coefficient together with its standard error and confidence limits. This table also includes the overall weighted kappa coefficient if the two-way table dimension is greater than 2.

- For multiway square tables, the AGREE option also produces the "Tests for Equal Kappa Coefficients" table. This table includes the chi-square statistic, degrees of freedom, and *p*-value for the test of equal simple kappa coefficients (over all strata). If the two-way table dimension is greater than 2, this table also includes the test for equal weighted kappa coefficients.

- For multiway tables in which each variable has two levels, both the AGREE option and the CQ option display the "Cochran's Q" table, which includes Cochran's Q statistic (to test for marginal homogeneity), the degrees of freedom, and the *p*-value.

- If you specify the COMMONRISKDIFF option for a multiway 2×2 table, PROC FREQTAB displays the "Confidence Limits for the Common Risk Difference" table, which includes the Method, Value of the common risk difference, Standard Error, and Confidence Limits for each confidence limit type that you request (Klingenberg, Mantel-Haenszel, Miettinen-Nurminen, Miettinen-Nurminen MH, Minimum Risk, Newcombe, Newcombe MR, or Summary Score).

- If you specify the COMMONRISKDIFF(TEST) option for a multiway 2×2 table, PROC FREQTAB displays the "Common Risk Difference Tests" table, which includes Method, Risk Difference, Z, and Pr > |Z| for each test that you request (Mantel-Haenszel, Minimum Risk, or Summary Score).

- If you specify the COMMONRISKDIFF(PRINTWTS) option for a multiway 2×2 table, PROC FREQTAB displays the "Stratum Weights" table, which includes the following information for each stratum (2×2 table): Stratum index, variable levels, Risk Difference, Frequency, Fraction, and the stratum weights that you request (Mantel-Haenszel, Miettinen-Nurminen, Minimum Risk, or Summary Score Weights).

- If you specify the CMH option, PROC FREQTAB displays Cochran-Mantel-Haenszel Statistics for the following three alternative hypotheses: Nonzero Correlation, Row Mean Scores Differ (ANOVA Statistic), and General Association. For each of these statistics, PROC FREQTAB gives the degrees of freedom (DF) and the probability value (Prob). If you specify the MANTELFLEISS option, PROC FREQTAB displays the Mantel-Fleiss Criterion for 2×2 tables. For 2×2 tables, PROC FREQTAB also displays Estimates of the Common Relative Risk for Case-Control and Cohort studies, together with their confidence limits. These include both Mantel-Haenszel and Logit stratum-adjusted estimates of the common Odds Ratio, Column 1 Relative Risk, and Column 2 Relative Risk. Also for 2×2 tables, PROC FREQTAB displays the Breslow-Day Test for Homogeneity of the Odds Ratios. For this test, PROC FREQTAB gives the Chi-Square, the degrees of freedom (DF), and the probability value (Pr > ChiSq).

- If you specify the CMH(QOR) option for a stratified 2×2 table, PROC FREQTAB displays the "Q Test for Homogeneity of Odds Ratios" table, which includes the Chi-Square, the degrees of freedom (DF), and the probability value (Pr > ChiSq).

- If you specify the CMH(I2) option for a stratified 2×2 table, PROC FREQTAB displays the "I-Square Measure of Heterogeneity" table, which includes the I-Square, the degrees of freedom (DF), and the Confidence Limits.

- If you specify the CMH option in the TABLES statement and also specify the COMOR option in the EXACT statement for a multiway 2×2 table, PROC FREQTAB displays exact confidence limits for the Common Odds Ratio. PROC FREQTAB also displays the Exact Test of H0: Common Odds Ratio = 1. The test output includes the Cell (1,1) Sum (S), Mean of S Under H0, One-sided Pr <= S, and Point Pr = S. PROC FREQTAB also provides exact two-sided probability values for the test, computed according to the following three methods: 2 * One-sided, Sum of probabilities <= Point probability, and Pr >= |S - Mean|. If you specify the MIDP option in the EXACT statement, PROC FREQTAB provides the exact Mid *p*-Value for the common odds ratio test.

- If you specify the CMH option in the TABLES statement and also specify the EQOR option in the EXACT statement for a multiway 2×2 table, PROC FREQTAB computes Zelen's exact test for equal odds ratios. PROC FREQTAB displays Zelen's test along with the asymptotic Breslow-Day test produced by the CMH option. PROC FREQTAB displays the test statistic, Zelen's Exact Test (P), and the probability value, Exact Pr <= P.

- If you specify the GAILSIMON option in the TABLES statement for a multiway 2×2 tables, PROC FREQTAB displays the Gail-Simon test for qualitative interactions. The display include the following statistics and their *p*-values: Q+ (Positive Risk Differences), Q- (Negative Risk Differences), and Q (Two-Sided).

ODS Table Names

PROC FREQTAB assigns a name to each table that it creates. You can use these names to refer to tables when you use the Output Delivery System (ODS) to select tables and create output data sets. For more information about ODS, see the *SAS Output Delivery System: User's Guide*.

Table 11.21 lists the ODS table names together with their descriptions and the options that are required to produce the tables.

Table 11.21 ODS Tables Produced by PROC FREQTAB

ODS Table Name	Description	Statement	Option
BarnardsTest	Barnard's exact test	EXACT	BARNARD
Binomial	Binomial proportion	TABLES	BINOMIAL
BinomialCLs	Binomial confidence limits	TABLES	BINOMIAL(CL=)
BinomialEquiv	Binomial equivalence analysis	TABLES	BINOMIAL(EQUIV)
BinomialEquivLimits	Binomial equivalence limits	TABLES	BINOMIAL(EQUIV)
BinomialEquivTest	Binomial equivalence test	TABLES	BINOMIAL(EQUIV)
BinomialNoninf	Binomial noninferiority test	TABLES	BINOMIAL(NONINF)
BinomialSup	Binomial superiority test	TABLES	BINOMIAL(SUP)
BinomialTest	Binomial proportion test	TABLES	BINOMIAL
BnMeasure	Agreement measures	TABLES	PLOTS=AGREEPLOT(STATS)
BreslowDayTest	Breslow-Day test	TABLES	CMH ($h \times 2 \times 2$ table)
ChiSq	Chi-square tests	TABLES	CHISQ
CMH	Cochran-Mantel-Haenszel statistics	TABLES	CMH
CochransQ	Cochran's Q	TABLES	AGREE or CQ ($2 \times 2 \times \cdots \times 2$ table)
ColScores	Column scores	TABLES	SCOROUT
CommonOddsRatioCl	Exact confidence limits for the common odds ratio	EXACT	COMOR ($h \times 2 \times 2$ table)
CommonOddsRatioTest	Common odds ratio exact test	EXACT	COMOR ($h \times 2 \times 2$ table)
CommonPdiff	Common risk difference confidence limits	TABLES	COMMONRISKDIFF ($h \times 2 \times 2$ table)
CommonPdiffTests	Common risk difference tests	TABLES	COMMONRISKDIFF(TESTS) ($h \times 2 \times 2$ table)
CommonRelRisks	Common relative risks	TABLES	CMH ($h \times 2 \times 2$ table)
CrossList	Crosstabulation table in column format	TABLES	CROSSLIST (n-way table, $n > 1$)
CrossTabFreqs	Crosstabulation table	TABLES	(n-way table, $n > 1$)
EqualKappaTest	Test for equal simple kappas	TABLES	AGREE ($h \times 2 \times 2$ table)
EqualKappaTests	Tests for equal kappas	TABLES	AGREE ($h \times r \times r$ table, $r > 2$)
EqualOddsRatios	Tests for equal odds ratios	EXACT	EQOR ($h \times 2 \times 2$ table)

Table 11.21 *continued*

ODS Table Name	Description	Statement	Option
FishersExact	Fisher's exact test	EXACT	FISHER
		or TABLES	FISHER or EXACT
		or TABLES	CHISQ (2×2 table)
FishersExactMC	Monte Carlo estimates for Fisher's exact test	EXACT	FISHER / MC
GailSimonTest	Gail-Simon test	TABLES	GAILSIMON ($h \times 2 \times 2$ table)
Gamma	Gamma	TEST	GAMMA
GammaTest	Gamma test	TEST	GAMMA
I2	I-square measure	TABLES	CMH(I2) ($h \times 2 \times 2$ table)
JTTest	Jonckheere-Terpstra test	TABLES	JT
JTTestMC	Monte Carlo estimates for Jonckheere-Terpstra exact test	EXACT	JT / MC
KappaDetails	Kappa details	TABLES	AGREE(KAPPADETAILS)
KappaMC	Monte Carlo exact test for simple kappa coefficient	EXACT	KAPPA / MC
KappaStatistics	Kappa statistics	TABLES	AGREE
KappaTest	Simple kappa test	TEST	KAPPA
		or EXACT	KAPPA
		or TABLES	AGREE(NULLKAPPA=)
KappaWeights	Kappa weights	TABLES	AGREE(PRINTKWTS)
List	List format multiway table	TABLES	LIST
LRChiSq	Likelihood ratio chi-square exact test	EXACT	LRCHI
LRChiSqMC	Monte Carlo exact test for likelihood ratio chi-square	EXACT	LRCHI / MC
MantelFleiss	Mantel-Fleiss criterion	TABLES	CMH(MANTELFLEISS) ($h \times 2 \times 2$ table)
McNemarsTest	McNemar's test	TABLES	AGREE (2×2 table)
Measures	Measures of association	TABLES	MEASURES
MHChiSq	Mantel-Haenszel chi-square exact test	EXACT	MHCHI
MHChiSqMC	Monte Carlo exact test for Mantel-Haenszel chi-square	EXACT	MHCHI / MC
NLevels	Number of variable levels	PROC	NLEVELS
OddsRatioCLs	Odds ratio confidence limits	TABLES	OR(CL=) (2×2 table)
OddsRatioExactCL	Exact confidence limits for the odds ratio	EXACT	OR (2×2 table)
OneWayChiSq	One-way chi-square test	TABLES	CHISQ (one-way table)
OneWayChiSqMC	Monte Carlo exact test for one-way chi-square	EXACT	CHISQ / MC (one-way table)

Table 11.21 *continued*

ODS Table Name	Description	Statement	Option
OneWayFreqs	One-way frequencies	PROC or TABLES	(no TABLES stmt) (one-way table)
OneWayLRChiSq	One-way likelihood ratio chi-square test	TABLES	CHISQ(LRCHI) (one-way table)
OverallKappa	Overall simple kappa	TABLES	AGREE ($h \times 2 \times 2$ table)
OverallKappas	Overall kappa coefficients	TABLES	AGREE ($h \times r \times r$ table, $r > 2$)
PdiffCLs	Risk difference confidence limits	TABLES	RISKDIFF(CL=) (2×2 table)
PdiffEquiv	Equivalence analysis for the risk difference	TABLES	RISKDIFF(EQUIV) (2×2 table)
PdiffEquivTest	Equivalence test for the risk difference	TABLES	RISKDIFF(EQUIV) (2×2 table)
PdiffNoninf	Noninferiority test for the risk difference	TABLES	RISKDIFF(NONINF) (2×2 table)
PdiffSup	Superiority test for the risk difference	TABLES	RISKDIFF(SUP) (2×2 table)
PdiffTest	Risk difference test	TABLES	RISKDIFF(EQUAL) (2×2 table)
PearsonChiSq	Pearson chi-square exact test	EXACT	PCHI
PearsonChiSqMC	Monte Carlo exact test for Pearson chi-square	EXACT	PCHI / MC
PearsonCorr	Pearson correlation	TEST or EXACT	PCORR PCORR
PearsonCorrMC	Monte Carlo exact test for Pearson correlation	EXACT	PCORR / MC
PearsonCorrTest	Pearson correlation test	TEST or EXACT	PCORR PCORR
PlCorr	Polychoric correlation	TEST	PLCORR
PlCorrTest	Polychoric correlation test	TEST	PLCORR
QOR	Q test for odds ratios	TABLES	CMH(QOR) or CMH(I2) ($h \times 2 \times 2$ table)
RelativeRiskCLs	Relative risk confidence limits	TABLES	RELRISK(CL=) (2×2 table)
RelativeRisks	Relative risk estimates	TABLES	RELRISK or MEASURES (2×2 table)
RelRisk1ExactCL	Exact confidence limits for column 1 relative risk	EXACT	RELRISK (2×2 table)
RelRisk2ExactCL	Exact confidence limits for column 2 relative risk	EXACT	RELRISK (2×2 table)
RelriskEquiv	Equivalence analysis for the relative risk	TABLES	RELRISK(EQUIV) (2×2 table)

Table 11.21 *continued*

ODS Table Name	Description	Statement	Option
RelriskEquivTest	Equivalence test for the relative risk	TABLES	RELRISK(EQUIV) (2×2 table)
RelriskNoninf	Noninferiority test for the relative risk	TABLES	RELRISK(NONINF) (2×2 table)
RelriskSup	Superiority test for the relative risk	TABLES	RELRISK(SUP) (2×2 table)
RelriskTest	Relative risk test	TABLES	RELRISK(EQUAL) (2×2 table)
RiskDiffCol1	Column 1 risk estimates	TABLES	RISKDIFF (2×2 table)
RiskDiffCol2	Column 2 risk estimates	TABLES	RISKDIFF (2×2 table)
RowScores	Row scores	TABLES	SCOROUT
SenSpec	Sensitivity and specificity	TABLES	SENSPEC (2×2 table)
SomersDCR	Somers' $D(C\|R)$	TEST	SMDCR
		or EXACT	SMDCR
SomersDCRMC	Monte Carlo exact test for Somers' $D(C\|R)$	EXACT	SMDCR / MC
SomersDCRTest	Somers' $D(C\|R)$ test	TEST	SMDCR
		or EXACT	SMDCR
SomersDRC	Somers' $D(R\|C)$	TEST	SMDRC
		or EXACT	SMDRC
SomersDRCMC	Monte Carlo exact test for Somers' $D(R\|C)$	EXACT	SMDRC / MC
SomersDRCTest	Somers' $D(R\|C)$ test	TEST	SMDRC
		or EXACT	SMDRC
SpearmanCorr	Spearman correlation	TEST	SCORR
		or EXACT	SCORR
SpearmanCorrMC	Monte Carlo exact test for Spearman correlation	EXACT	SCORR / MC
SpearmanCorrTest	Spearman correlation test	TEST	SCORR
		or EXACT	SCORR
StratumWeights	Stratum risk differences and weights	TABLES	COMMONRISKDIFF ($h \times 2 \times 2$ table)
SymmetryMC	Monte Carlo exact symmetry test	EXACT	SYMMETRY / MC
SymmetryTest	Symmetry test	TABLES	AGREE
TauB	Kendall's tau-b	TEST	KENTB
		or EXACT	KENTB
TauBMC	Monte Carlo exact test for Kendall's tau-b	EXACT	KENTB / MC
TauBTest	Kendall's tau-b test	TEST	KENTB
		or EXACT	KENTB
TauC	Stuart's tau-c	TEST	STUTC
		or EXACT	STUTC

Table 11.21 *continued*

ODS Table Name	Description	Statement	Option
TauCMC	Monte Carlo exact test for Stuart's tau-*c*	EXACT	STUTC / MC
TauCTest	Stuart's tau-*c* test	TEST or EXACT	STUTC STUTC
TrendTest	Cochran-Armitage trend test	TABLES	TREND
TrendTestMC	Monte Carlo exact test for trend	EXACT	TREND / MC
WtKappaDetails	Weighted kappa details	TABLES	AGREE(WTKAPPADETAILS)
WtKappaMC	Monte Carlo exact test for weighted kappa coefficient	EXACT	WTKAPPA / MC
WtKappaTest	Weighted kappa test	TEST or EXACT or TABLES	WTKAPPA WTKAPPA AGREE(NULLWTKAPPA=)

ODS Graphics

Statistical procedures use ODS Graphics to create graphs as part of their output. ODS Graphics is described in detail in the "Statistical Graphics Using ODS" chapter in *SAS/STAT User's Guide*.

Before you create graphs, ODS Graphics must be enabled (for example, by using the ODS GRAPHICS ON statement). For more information about enabling and disabling ODS Graphics, see the section "Enabling and Disabling ODS Graphics" in that chapter.

The overall appearance of graphs is controlled by ODS styles. Styles and other aspects of using ODS Graphics are discussed in the section "A Primer on ODS Statistical Graphics" in that chapter.

When ODS Graphics is enabled, you can request specific plots by specifying the PLOTS= option in the TABLES statement. To produce a frequency plot or cumulative frequency plot, you must specify the FREQPLOT or CUMFREQPLOT *plot-request*, respectively, in the PLOTS= option. To produce a mosaic plot, you must specify the MOSAICPLOT *plot-request* in the PLOTS= option. You can also produce frequency, cumulative frequency, and mosaic plots by specifying the PLOTS=ALL option. By default, PROC FREQTAB produces all other plots that are associated with the analyses that you request in the TABLES statement. You can suppress the default plots and request specific plots by using the PLOTS(ONLY)= option. For more information, see the description of the PLOTS= option.

PROC FREQTAB assigns a name to each graph that it creates by using ODS Graphics. You can use these names to refer to the graphs. Table 11.22 lists the names of the graphs that PROC FREQTAB generates together with their descriptions, their PLOTS= options (*plot-requests*), and the TABLES statement options that are required to produce the graphs.

Table 11.22 Graphs Produced by PROC FREQTAB

ODS Graph Name	Description	PLOTS= Option	TABLES Statement Option
AgreePlot	Agreement plot	AGREEPLOT	AGREE ($r \times r$ table)
CumFreqPlot	Cumulative frequency plot	CUMFREQPLOT	One-way table request
DeviationPlot	Deviation plot	DEVIATIONPLOT	CHISQ (one-way table)
FreqPlot	Frequency plot	FREQPLOT	Any table request
KappaPlot	Kappa plot	KAPPAPLOT	AGREE ($h \times r \times r$ table)
MosaicPlot	Mosaic plot	MOSAICPLOT	Two-way or multiway table request
ORPlot	Odds ratio plot	ODDSRATIOPLOT	MEASURES, OR, or RELRISK ($h \times 2 \times 2$ table)
RelRiskPlot	Relative risk plot	RELRISKPLOT	MEASURES or RELRISK ($h \times 2 \times 2$ table)
RiskDiffPlot	Risk difference plot	RISKDIFFPLOT	RISKDIFF ($h \times 2 \times 2$ table)
WtKappaPlot	Weighted kappa plot	WTKAPPAPLOT	AGREE ($h \times r \times r$ table, $r > 2$)

The examples in this chapter are available in the GitHub repository located at `https://github.com/sascommunities/doc-supplement-casstat`.

References

Agresti, A. (1992). "A Survey of Exact Inference for Contingency Tables." *Statistical Science* 7:131–177.

Agresti, A. (2002). *Categorical Data Analysis*. 2nd ed. New York: John Wiley & Sons.

Agresti, A. (2007). *An Introduction to Categorical Data Analysis*. 2nd ed. New York: John Wiley & Sons.

Agresti, A. (2013). *Categorical Data Analysis*. 3rd ed. Hoboken, NJ: John Wiley & Sons.

Agresti, A., and Caffo, B. (2000). "Simple and Effective Confidence Intervals for Proportions and Differences of Proportions Result from Adding Two Successes and Two Failures." *American Statistician* 54:280–288.

Agresti, A., and Coull, B. A. (1998). "Approximate Is Better Than 'Exact' for Interval Estimation of Binomial Proportions." *American Statistician* 52:119–126.

Agresti, A., and Gottard, A. (2007). "Nonconservative Exact Small-Sample Inference for Discrete Data." *Computational Statistics and Data Analysis* 51:6447–6458.

Agresti, A., Mehta, C. R., and Patel, N. R. (1990). "Exact Inference for Contingency Tables with Ordered Categories." *Journal of the American Statistical Association* 85:453–458.

Agresti, A., and Min, Y. (2001). "On Small-Sample Confidence Intervals for Parameters in Discrete Distributions." *Biometrics* 57:963–971.

Agresti, A., Wackerly, D., and Boyett, J. M. (1979). "Exact Conditional Tests for Cross-Classifications: Approximation of Attained Significance Levels." *Psychometrika* 44:75–83.

Bangdiwala, S. I. (1988). *The Agreement Chart*. Technical report, Department of Biostatistics, University of North Carolina at Chapel Hill.

Bangdiwala, S. I., and Bryan, H. E. (1987). "Using SAS Software Graphical Procedures for the Observer Agreement Chart." In *Proceedings of the Twelfth Annual SAS Users Group International Conference*, 1083–1088. Cary, NC: SAS Institute Inc.

Bangdiwala, S. I., Haedo, A. S., Natal, M. L., and Villaveces, A. (2008). "The Agreement Chart as an Alternative to the Receiver-Operating Characteristic Curve for Diagnostic Tests." *Journal of Clinical Epidemiology* 61:866–874.

Barker, L., Rolka, H., Rolka, D., and Brown, C. (2001). "Equivalence Testing for Binomial Random Variables: Which Test to Use?" *American Statistician* 55:279–287.

Barnard, G. A. (1945). "A New Test for 2×2 Tables." *Nature* 156:177.

Barnard, G. A. (1947). "Significance Tests for 2×2 Tables." *Biometrika* 34:123–138.

Barnard, G. A. (1949). "Statistical Inference." *Journal of the Royal Statistical Society, Series B* 11:115–139.

Berger, J. O. (1985). *Statistical Decision Theory and Bayesian Analysis*. 2nd ed. New York: Springer-Verlag.

Birch, M. W. (1965). "The Detection of Partial Association, Part 2: The General Case." *Journal of the Royal Statistical Society, Series B* 27:111–124.

Bishop, Y. M. M., Fienberg, S. E., and Holland, P. W. (1975). *Discrete Multivariate Analysis: Theory and Practice*. Cambridge, MA: MIT Press.

Blaker, H. (2000). "Confidence Curves and Improved Exact Confidence Intervals for Discrete Distributions." *Canadian Journal of Statistics* 28:783–798.

Blood, E., and Spratt, K. F. (2007). "Disagreement on Agreement: Two Alternative Agreement Coefficients." In *Proceedings of the SAS Global Forum 2007 Conference*. Cary, NC: SAS Institute Inc. http://www2.sas.com/proceedings/forum2007/186-2007.pdf.

Bowker, A. H. (1948). "Bowker's Test for Symmetry." *Journal of the American Statistical Association* 43:572–574.

Breslow, N. E. (1996). "Statistics in Epidemiology: The Case-Control Study." *Journal of the American Statistical Association* 91:14–26.

Breslow, N. E., and Day, N. E. (1980). *The Analysis of Case-Control Studies*. Statistical Methods in Cancer Research, IARC Scientific Publications, vol. 1, no. 32. Lyon: International Agency for Research on Cancer.

Breslow, N. E., and Day, N. E. (1987). *The Design and Analysis of Cohort Studies*. Statistical Methods in Cancer Research, IARC Scientific Publications, vol. 2, no. 82. Lyon: International Agency for Research on Cancer.

Bross, I. D. J. (1958). "How to Use Ridit Analysis." *Biometrics* 14:18–38.

Brown, L. D., Cai, T. T., and DasGupta, A. (2001). "Interval Estimation for a Binomial Proportion." *Statistical Science* 16:101–133.

Brown, M. B., and Benedetti, J. K. (1977a). "On the Mean and Variance of the Tetrachoric Correlation Coefficient." *Psychometrika* 42:347–355.

Brown, M. B., and Benedetti, J. K. (1977b). "Sampling Behavior of Tests for Correlation in Two-Way Contingency Tables." *Journal of the American Statistical Association* 72:309–315.

Byrt, T., Bishop, J., and Carlin, J. B. (1993). "Bias, Prevalence, and Kappa." *Journal of Clinical Epidemiology* 46:423–429.

Chan, I. S. F. (1998). "Exact Tests of Equivalence and Efficacy with a Non-zero Lower Bound for Comparative Studies." *Statistics in Medicine* 17:1403–1413.

Chan, I. S. F. (2003). "Proving Non-inferiority or Equivalence of Two Treatments with Dichotomous Endpoints Using Exact Methods." *Statistical Methods in Medical Research* 12:37–58.

Chan, I. S. F., and Zhang, Z. (1999). "Test-Based Exact Confidence Intervals for the Difference of Two Binomial Proportions." *Biometrics* 55:1202–1209.

Chow, S.-C., Shao, J., and Wang, H. (2003). *Sample Size Calculations in Clinical Research*. Boca Raton, FL: CRC Press.

Chow, S.-C., Shao, J., and Wang, H. (2008). *Sample Size Calculations in Clinical Research*. 2nd ed. Boca Raton, FL: Chapman & Hall/CRC.

Cicchetti, D. V., and Allison, T. (1971). "A New Procedure for Assessing Reliability of Scoring EEG Sleep Recordings." *American Journal of EEG Technology* 11:101–109.

Clopper, C. J., and Pearson, E. S. (1934). "The Use of Confidence or Fiducial Limits Illustrated in the Case of the Binomial." *Biometrika* 26:404–413.

Cochran, W. G. (1950). "The Comparison of Percentages in Matched Samples." *Biometrika* 37:256–266.

Cochran, W. G. (1954). "Some Methods for Strengthening the Common χ^2 Tests." *Biometrics* 10:417–451.

Cohen, J. (1960). "A Coefficient of Agreement for Nominal Scales." *Educational and Psychological Measurement* 20:37–46.

Collett, D. (1991). *Modelling Binary Data*. London: Chapman & Hall.

Dann, R. S., and Koch, G. G. (2005). "Review and Evaluation of Methods for Computing Confidence Intervals for the Ratio of Two Proportions and Considerations for Non-inferiority Clinical Trials." *Journal of Biopharmaceutical Statistics* 15:85–107.

Dmitrienko, A., Molenberghs, G., Chuang-Stein, C., and Offen, W. (2005). *Analysis of Clinical Trials Using SAS: A Practical Guide*. Cary, NC: SAS Institute Inc.

Drasgow, F. (1986). "Polychoric and Polyserial Correlations." In *Encyclopedia of Statistical Sciences*, vol. 7, edited by S. Kotz, N. L. Johnson, and C. B. Read. New York: John Wiley & Sons.

Dunnett, C. W., and Gent, M. (1977). "Significance Testing to Establish Equivalence between Treatments, with Special Reference to Data in the Form of 2×2 Tables." *Biometrics* 33:593–602.

Farrington, C. P., and Manning, G. (1990). "Test Statistics and Sample Size Formulae for Comparative Binomial Trials with Null Hypothesis of Non-zero Risk Difference or Non-unity Relative Risk." *Statistics in Medicine* 9:1447–1454.

Fienberg, S. E. (1980). *The Analysis of Cross-Classified Categorical Data*. 2nd ed. Cambridge, MA: MIT Press.

Fisher, K. (2015). "Evaluating Methods of Estimating Common Risk Difference for Stratified Binomial Trials for Less Common Events." Poster presented at the Duke-Industry Statistics Symposium, Durham, NC. https://sites.duke.edu/biostatsworkshop2015/files/2015/10/P8_Poster_for_Duke_Symposium_Kate_Fisher_101915_Final.pdf.

Fleiss, J. L., and Cohen, J. (1973). "The Equivalence of Weighted Kappa and the Intraclass Correlation Coefficient as Measures of Reliability." *Educational and Psychological Measurement* 33:613–619.

Fleiss, J. L., Cohen, J., and Everitt, B. S. (1969). "Large-Sample Standard Errors of Kappa and Weighted Kappa." *Psychological Bulletin* 72:323–327.

Fleiss, J. L., Levin, B., and Paik, M. C. (2003). *Statistical Methods for Rates and Proportions*. 3rd ed. Hoboken, NJ: John Wiley & Sons.

Freeman, G. H., and Halton, J. H. (1951). "Note on an Exact Treatment of Contingency, Goodness of Fit, and Other Problems of Significance." *Biometrika* 38:141–149.

Friendly, M. (2000). *Visualizing Categorical Data*. Cary, NC: SAS Institute Inc.

Gail, M. H., and Mantel, N. (1977). "Counting the Number of $r \times c$ Contingency Tables with Fixed Margins." *Journal of the American Statistical Association* 72:859–862.

Gail, M. H., and Simon, R. (1985). "Tests for Qualitative Interactions between Treatment Effects and Patient Subsets." *Biometrics* 41:361–372.

Gart, J. J. (1971). "The Comparison of Proportions: A Review of Significance Tests, Confidence Intervals, and Adjustments for Stratification." *Review of the International Statistical Institute* 39:148–169.

Gart, J. J., and Nam, J. (1988). "Approximate Interval Estimation of the Ratio of Binomial Parameters: A Review and Corrections for Skewness." *Biometrics* 44:323–338.

Goodman, L. A., and Kruskal, W. H. (1979). *Measures of Association for Cross Classification*. New York: Springer-Verlag.

Greenland, S., and Robins, J. M. (1985). "Estimation of a Common Effect Parameter from Sparse Follow-Up Data." *Biometrics* 41:55–68.

Gwet, K. L. (2008). "Computing Inter-rater Reliability and Its Variance in the Presence of High Agreement." *British Journal of Mathematical and Statistical Psychology* 61:29–48.

Haldane, J. B. S. (1956). "The Estimation and Significance of the Logarithm of a Ratio of Frequencies." *Annals of Human Genetics* 20:309–311.

Hauck, W. W., and Anderson, S. (1986). "A Comparison of Large-Sample Confidence Interval Methods for the Difference of Two Binomial Probabilities." *American Statistician* 40:318–322.

Higgins, J. P. T., and Thompson, S. G. (2002). "Quantifying Heterogeneity in a Meta-analysis." *Statistics in Medicine* 21:1539–1558.

Higgins, J. P. T., Thompson, S. G., Deeks, J. J., and Altman, D. G. (2003). "Measuring Inconsistency in Meta-analyses." *BMJ* 327:557–560.

Hirji, K. F. (2006). *Exact Analysis of Discrete Data*. Boca Raton, FL: Chapman & Hall/CRC.

Hirji, K. F., Vollset, S. E., Reis, I. M., and Afifi, A. A. (1996). "Exact Tests for Interaction in Several 2×2 Tables." *Journal of Computational and Graphical Statistics* 5:209–224.

Hoenig, J. M., Morgan, M. J., and Brown, C. A. (1995). "Analysing Differences between Two Age Determination Methods by Tests of Symmetry." *Canadian Journal of Fisheries and Aquatic Sciences* 52:364–368.

Hollander, M., and Wolfe, D. A. (1999). *Nonparametric Statistical Methods*. 2nd ed. New York: John Wiley & Sons.

Holley, J. W., and Guilford, J. P. (1964). "A Note on the G Index of Agreement." *Educational and Psychological Measurement* 24:749–753.

Jones, M. P., O'Gorman, T. W., Lemka, J. H., and Woolson, R. F. (1989). "A Monte Carlo Investigation of Homogeneity Tests of the Odds Ratio under Various Sample Size Configurations." *Biometrics* 45:171–181.

Kendall, M. G. (1955). *Rank Correlation Methods*. 2nd ed. London: Charles Griffin.

Kendall, M. G., and Stuart, A. (1979). *The Advanced Theory of Statistics.* 4th ed. Vol. 2. New York: Macmillan.

Kim, Y., and Won, S. (2013). "Adjusted Proportion Difference and Confidence Interval in Stratified Randomized Trials." In *Proceedings of PharmaSUG 2013 (Pharmaceutical Industry SAS Users Group).* Paper SP04. Cary, NC: SAS Institute Inc.

Kleinbaum, D. G., Kupper, L. L., and Morgenstern, H. (1982). *Epidemiologic Research: Principles and Quantitative Methods*. New York: Van Nostrand Reinhold.

Klingenberg, B. (2014). "A New and Improved Confidence Interval for the Mantel-Haenszel Risk Difference." *Statistics in Medicine* 33:2968–2983.

Korn, E. L., and Graubard, B. I. (1998). "Confidence Intervals for Proportions with Small Expected Number of Positive Counts Estimated from Survey Data." *Survey Methodology* 24:193–201.

Krauth, J. (1973). "Nichtparametrische Ansätze zur Auswertung von Verlaufskurven [Nonparametric approaches to analyzing time effect curves]." *Biometrische Zeitschrift* 15:557–566.

Landis, J. R., Heyman, E. R., and Koch, G. G. (1978). "Average Partial Association in Three-Way Contingency Tables: A Review and Discussion of Alternative Tests." *International Statistical Review* 46:237–254.

Leemis, L. M., and Trivedi, K. S. (1996). "A Comparison of Approximate Interval Estimators for the Bernoulli Parameter." *American Statistician* 50:63–68.

Lehmann, E. L., and D'Abrera, H. J. M. (2006). *Nonparametrics: Statistical Methods Based on Ranks*. Rev. ed. New York: Springer.

Liebetrau, A. M. (1983). *Measures of Association.* Vol. 32 of Quantitative Applications in the Social Sciences. Beverly Hills, CA: Sage Publications.

Lu, K. (2008). "Cochran-Mantel-Haenszel Weighted Miettinen and Nurminen Method for Confidence Intervals of the Difference in Binomial Proportions from Stratified 2x2 Samples." In *Proceedings of the Joint Statistical Meetings*, 3888–3893. Alexandria, VA: American Statistical Association.

Mack, G. A., and Skillings, J. H. (1980). "A Friedman-Type Rank Test for Main Effects in a Two-Factor ANOVA." *Journal of the American Statistical Association* 75:947–951.

Mantel, N. (1963). "Chi-Square Tests with One Degree of Freedom: Extensions of the Mantel-Haenszel Procedure." *Journal of the American Statistical Association* 58:690–700.

Mantel, N., and Fleiss, J. L. (1980). "Minimum Expected Cell Size Requirements for the Mantel-Haenszel One-Degree-of-Freedom Chi-Square Test and a Related Rapid Procedure." *American Journal of Epidemiology* 112:129–134.

Mantel, N., and Haenszel, W. (1959). "Statistical Aspects of Analysis of Data from Retrospective Studies of Disease." *Journal of the National Cancer Institute* 22:719–748.

Margolin, B. H. (1988). "Test for Trend in Proportions." In *Encyclopedia of Statistical Sciences*, vol. 9, edited by S. Kotz, N. L. Johnson, and C. B. Read. New York: John Wiley & Sons.

McNemar, Q. (1947). "Note on the Sampling Error of the Difference between Correlated Proportions or Percentages." *Psychometrika* 12:153–157.

Mee, R. W. (1984). "Confidence Bounds for the Difference between Two Probabilities." *Biometrics* 40:1175–1176.

Mehrotra, D. V. (2001). "Stratification Issues with Binary Endpoints." *Drug Information Journal* 35:1343–1350.

Mehrotra, D. V., and Railkar, R. (2000). "Minimum Risk Weights for Comparing Treatments in Stratified Binomial Trials." *Statistics in Medicine* 19:811–825.

Mehta, C. R., and Patel, N. R. (1983). "A Network Algorithm for Performing Fisher's Exact Test in $r \times c$ Contingency Tables." *Journal of the American Statistical Association* 78:427–434.

Mehta, C. R., Patel, N. R., and Gray, R. J. (1985). "Computing an Exact Confidence Interval for the Common Odds Ratio in Several 2×2 Contingency Tables." *Journal of the American Statistical Association* 80:969–973.

Mehta, C. R., Patel, N. R., and Senchaudhuri, P. (1991). "Exact Stratified Linear Rank Tests for Binary Data." In *Computing Science and Statistics: Proceedings of the Twenty-Third Symposium on the Interface*, edited by E. M. Keramidas, 200–207. Fairfax Station, VA: Interface Foundation.

Mehta, C. R., Patel, N. R., and Tsiatis, A. A. (1984). "Exact Significance Testing to Establish Treatment Equivalence with Ordered Categorical Data." *Biometrics* 40:819–825.

Mehta, C. R., and Senchaudhuri, P. (2003). "Conditional versus Unconditional Exact Tests for Comparing Two Binomials." Cambridge, MA: Cytel Software Corporation.

Miettinen, O. S. (1985). *Theoretical Epidemiology: Principles of Occurrence in Research Medicine*. New York: John Wiley & Sons.

Miettinen, O. S., and Nurminen, M. M. (1985). "Comparative Analysis of Two Rates." *Statistics in Medicine* 4:213–226.

Newcombe, R. G. (1998a). "Interval Estimation for the Difference between Independent Proportions: Comparison of Eleven Methods." *Statistics in Medicine* 17:873–890.

Newcombe, R. G. (1998b). "Two-Sided Confidence Intervals for the Single Proportion: Comparison of Seven Methods." *Statistics in Medicine* 17:857–872.

Newcombe, R. G., and Nurminen, M. M. (2011). "In Defence of Score Intervals for Proportions and Their Differences." *Communications in Statistics—Theory and Methods* 40:1271–1282.

Olsson, U. (1979). "Maximum Likelihood Estimation of the Polychoric Correlation Coefficient." *Psychometrika* 12:443–460.

Pirie, W. (1983). "Jonckheere Tests for Ordered Alternatives." In *Encyclopedia of Statistical Sciences*, vol. 4, edited by S. Kotz, N. L. Johnson, and C. B. Read. New York: John Wiley & Sons.

Radlow, R., and Alf, E. F. (1975). "An Alternate Multinomial Assessment of the Accuracy of the Chi-Square Test of Goodness of Fit." *Journal of the American Statistical Association* 70:811–813.

Robins, J. M., Breslow, N., and Greenland, S. (1986). "Estimators of the Mantel-Haenszel Variance Consistent in Both Sparse Data and Large-Strata Limiting Models." *Biometrics* 42:311–323.

Santner, T. J., Pradhan, V., Senchaudhuri, P., Mehta, C. R., and Tamhane, A. (2007). "Small-Sample Comparisons of Confidence Intervals for the Difference of Two Independent Binomial Proportions." *Computational Statistics and Data Analysis* 51:5791–5799.

Santner, T. J., and Snell, M. K. (1980). "Small-Sample Confidence Intervals for $p_1 - p_2$ and p_1/p_2 in 2×2 Contingency Tables." *Journal of the American Statistical Association* 75:386–394.

Sato, T. (1989). "On the Variance Estimator of the Mantel-Haenszel Risk Difference." *Biometrics* 45:1323–1324. Letter to the editor.

Schuirmann, D. J. (1987). "A Comparison of the Two One-Sided Tests Procedure and the Power Approach for Assessing the Equivalence of Average Bioavailability." *Journal of Pharmacokinetics and Biopharmaceutics* 15:657–680.

Schuirmann, D. J. (1999). "Confidence Interval Methods for Bioequivalence Testing with Binomial Endpoints." In *Proceedings of the Biopharmaceutical Section*, 227–232. Alexandria, VA: American Statistical Association.

Silvapulle, M. J. (2001). "Tests against Qualitative Interaction: Exact Critical Values and Robust Tests." *Biometrics* 57:1157–1165.

Sim, J., and Wright, C. C. (2005). "The Kappa Statistic in Reliability Studies: Use, Interpretation, and Sample Size Requirements." *Physical Therapy* 85:257–268.

Snedecor, G. W., and Cochran, W. G. (1989). *Statistical Methods*. 8th ed. Ames: Iowa State University Press.

Somers, R. H. (1962). "A New Asymmetric Measure of Association for Ordinal Variables." *American Sociological Review* 27:799–811.

Stokes, M. E., Davis, C. S., and Koch, G. G. (2012). *Categorical Data Analysis Using SAS*. 3rd ed. Cary, NC: SAS Institute Inc.

Suissa, S., and Shuster, J. J. (1985). "Exact Unconditional Sample Sizes for the 2×2 Binomial Trial." *Journal of the Royal Statistical Society, Series A* 148:317–327.

Tarone, R. E. (1985). "On Heterogeneity Tests Based on Efficient Scores." *Biometrika* 72:91–95.

Theil, H. (1972). *Statistical Decomposition Analysis*. Amsterdam: North-Holland.

Thomas, D. G. (1971). "Algorithm AS-36: Exact Confidence Limits for the Odds Ratio in a 2×2 Table." *Journal of the Royal Statistical Society, Series C* 20:105–110.

Thorlund, K., Imberger, G., Johnston, B. C., Walsh, M., Awad, T., Thabane, L., Gluud, C., et al. (2012). "Evolution of Heterogeneity (I^2) Estimates and Their 95% Confidence Intervals in Large Meta-analyses." *PLoS ONE* 7:1–8. https://doi.org/10.1371/journal.pone.0039471.

Valz, P. D., and Thompson, M. E. (1994). "Exact Inference for Kendall's S and Spearman's ρ with Extensions to Fisher's Exact Test in $r \times c$ Contingency Tables." *Journal of Computational and Graphical Statistics* 3:459–472.

Van Elteren, P. H. (1960). "On the Combination of Independent Two-Sample Tests of Wilcoxon." *Bulletin of the International Statistical Institute* 37:351–361.

Vollset, S. E., Hirji, K. F., and Elashoff, R. M. (1991). "Fast Computation of Exact Confidence Limits for the Common Odds Ratio in a Series of 2×2 Tables." *Journal of the American Statistical Association* 86:404–409.

Wilson, E. B. (1927). "Probable Inference, the Law of Succession, and Statistical Inference." *Journal of the American Statistical Association* 22:209–212.

Woolf, B. (1955). "On Estimating the Relationship between Blood Group and Disease." *Annals of Human Genetics* 19:251–253.

Xie, Q. (2013). "Agree or Disagree? A Demonstration of an Alternative Statistic to Cohen's Kappa for Measuring the Extent and Reliability of Agreement between Observers." Paper presented at FCSM Research Conference, Nov. 7–9, Washington, DC.

Yan, X., and Su, X. G. (2010). "Stratified Wilson and Newcombe Confidence Intervals for Multiple Binomial Proportions." *Statistics in Biopharmaceutical Research* 2:329–335.

Zelen, M. (1971). "The Analysis of Several 2×2 Contingency Tables." *Biometrika* 58:129–137.

Chapter 12
GAMMOD Procedure

Contents

Overview: GAMMOD Procedure **580**
PROC GAMMOD Features 580
PROC GAMMOD Compared with Other SAS Procedures 581
Using CAS Sessions and CAS Engine Librefs 583
Getting Started: GAMMOD Procedure **584**
Syntax: GAMMOD Procedure **590**
PROC GAMMOD Statement 590
BY Statement 594
CLASS Statement 594
DISPLAY Statement 594
DISPLAYOUT Statement 595
FREQ Statement 596
MODEL Statement 596
OUTPUT Statement 605
STORE Statement 607
WEIGHT Statement 607
Details: GAMMOD Procedure **607**
Missing Values 607
Thin-Plate Regression Splines 608
Generalized Additive Models 610
Model Evaluation Criteria 612
Fitting Algorithms 613
Degrees of Freedom 614
Model Inference 615
Dispersion Parameter 616
Tests for Smoothing Components 616
Multithreading 617
Optimization Algorithms 617
Displayed Output 618
ODS Table Names 621
ODS Graphics 622
Examples: GAMMOD Procedure **623**
Example 12.1: Scatter Plot Smoothing 623
Example 12.2: Nonparametric Logistic Regression 628
Example 12.3: Nonparametric Tweedie Regression 634
References **639**

Overview: GAMMOD Procedure

The GAMMOD procedure fits generalized additive models that are based on low-rank regression splines (Wood 2006) in SAS Viya.

Generalized additive models are extensions of generalized linear models. They relax the generalized linear models' assumption of linearity by allowing spline terms that characterize nonlinear dependency structures. Each spline term is constructed by the thin-plate regression spline technique (Wood 2003). A roughness penalty is applied to each spline term by a smoothing parameter that controls the balance between goodness of fit and the roughness of the spline curve. PROC GAMMOD fits models for standard distributions in the exponential family, such as the normal, Poisson, gamma, and Tweedie distributions.

PROC GAMMOD Features

PROC GAMMOD offers the following basic features:

- estimates the regression parameters of a generalized additive model that has fixed smoothing parameters by using penalized likelihood estimation
- estimates the smoothing parameters of a generalized additive model by using either the performance iteration method or the outer iteration method
- estimates the regression parameters of a generalized linear model by using maximum likelihood techniques
- tests the total contribution of each spline term based on the Wald statistic
- provides model-building syntax in the CLASS statement and effect-based parametric effects in the MODEL statement, which are used in other SAS/STAT analytic procedures (in particular, the GLM, LOGISTIC, GLIMMIX, and MIXED procedures)
- provides response-variable options
- enables you to construct a spline term by using multiple variables
- provides control options for constructing a spline term, such as fixed degrees of freedom, initial smoothing parameter, fixed smoothing parameter, smoothing parameter search range, user-supplied knot values, and so on
- provides multiple link functions for any distribution
- provides a WEIGHT statement for weighted analysis
- provides a FREQ statement for grouped analysis
- provides an OUTPUT statement to produce a data table that has predicted values and other observationwise statistics
- produces graphs via ODS Graphics

Because the GAMMOD procedure runs on SAS Cloud Analytic Services (CAS), it also does the following:

- enables you to run on a cluster of machines that distribute the data and the computations
- enables you to run in single-machine mode
- exploits all the available cores and concurrent threads (for information about how PROC GAMMOD uses threads, see the section "Multithreading" on page 102 in Chapter 2, "Shared Concepts")

PROC GAMMOD Compared with Other SAS Procedures

The GAMMOD procedure provides generalized additive modeling functionality that is comparable to that of GAMPL and GAM procedures in SAS/STAT software.

PROC GAMMOD Compared with the GAMPL Procedure

The functionality of the GAMMOD procedure closely resembles that of the GAMPL procedure, which is a high-performance SAS/STAT procedure. The GAMMOD procedure is the next generation of the GAMPL procedure, and it was developed specifically for SAS Viya. Both procedures are designed to run on a cluster of machines that distribute the data and the computations. Both procedures perform computations in multiple threads.

PROC GAMMOD provides all the features and options that PROC GAMPL supports; in addition, it supports BY processing and analytic stores. The default link functions for two distributions differ between the two procedures, as shown in Table 12.1.

Table 12.1 Default Link Function

Distribution	PROC GAMMOD	PROC GAMPL
Gamma	Log	Reciprocal
Inverse Gaussian	Log	Reciprocal square

You should expect close results between these two procedures, although exactly equivalent models are not guaranteed because their implementations are different.

PROC GAMMOD Compared with the GAM Procedure

Both the GAMMOD procedure and the GAM procedure in SAS/STAT software fit generalized additive models. However, the GAMMOD procedure uses different approaches for constructing spline basis expansions, fitting generalized additive models, and testing smoothing components. The GAMMOD procedure focuses on automatic smoothing parameter selection by using global model-evaluation criteria to find optimal models. The GAM procedure focuses on constructing models by fitting partial residuals against each smoothing term with fixed degrees of freedom. In general, you should not expect similar results from these two procedures. Table 12.2 summarizes the main differences.

Table 12.2 PROC GAMMOD Compared with PROC GAM

Functionality	PROC GAMMOD	PROC GAM
Constructing spline basis expansions	Uses thin-plate regression splines to construct basis expansions for each spline term, and each term allows multiple variables	Uses univariate or bivariate smoothing splines to construct basis expansions, and each term allows only one or two variables. Also allows loess smoothers.
Fitting generalized additive models	Fits models that have fixed smoothness terms by optimizing penalized likelihood. For models that have varying smoothness terms, PROC GAMMOD estimates smoothing parameters simultaneously by optimizing global model fit criteria such as generalized cross validation (GCV).	Fits models that have fixed smoothness terms by fitting partial residuals against each smoothing term. For models that have varying smoothness terms, PROC GAM estimates each smoothing parameter by optimizing the local GCV criterion for one spline term at a time.
Distribution families and link functions	Supports more distributions, including the negative binomial and the Tweedie families. Also supports any applicable link functions for each distribution.	Supports major distribution families and the canonical link function for each distribution
Testing smoothing components	Tests the total contribution for a spline term	Tests the existence of nonlinearity for a spline term beyond the linear trend
Model inference	A global Bayesian posterior covariance matrix is available. The confidence limits for each observation's prediction is available, in addition to componentwise confidence limits.	A local Bayesian posterior covariance matrix is available for each spline term. Only the componentwise confidence limits are available.
Model degrees of freedom	Uses the trace of the global degrees-of-freedom matrix	Uses the summation of the traces of individual smoothing matrix for each smoothing term
Multithreading scheme	Specifically designed to operate in SAS Viya and performs computations in multiple threads	Executes in a single thread on a single machine

Using CAS Sessions and CAS Engine Librefs

SAS Cloud Analytic Services (CAS) is the analytic server and associated cloud services in SAS Viya. This section describes how to create a CAS session and set up a CAS engine libref that you can use to connect to the CAS session. It assumes that you have a CAS server already available; contact your system administrator if you need help starting and terminating a server. This CAS server is identified by specifying the host on which it runs and the port on which it listens for communications. To simplify your interactions with this CAS server, the host information and port information for the server are stored as SAS option values that are retrieved automatically whenever this CAS server needs to be accessed. You can examine the host and port values for the server at your site by using the following statements:

```
proc options option=(CASHOST CASPORT);
run;
```

In addition to starting a CAS server, your system administrator might also have created a CAS session and a CAS engine libref for your use. You can define your own sessions and CAS engine librefs that connect to the CAS server as shown in the following statements:

```
cas mysess;
libname mylib cas sessref=mysess;
```

The CAS statement creates the CAS session named mysess, and the LIBNAME statement creates the mylib CAS engine libref that you use to connect to this session. It is not necessary to explicitly name the CASHOST and CASPORT of the CAS server in the CAS statement, because these values are retrieved from the corresponding SAS option values.

If you have created the mysess session, you can terminate it by using the TERMINATE option in the CAS statement as follows:

```
cas mysess terminate;
```

For more information about the CAS and LIBNAME statements, see the section "Introduction to Shared Concepts" on page 6 in Chapter 2, "Shared Concepts."

Getting Started: GAMMOD Procedure

(View the complete code for this example at https://github.com/sascommunities/doc-supplement-casstat/tree/main/Examples/gammodgs1.sas.)

This example concerns the proportions and demographic and geographic characteristics of votes that were cast in 3,107 counties in the United States in the 1980 presidential election. You can use the data set sashelp.Vote1980 directly from the Sashelp library or download it from the StatLib data sets archive (Vlachos 1998). For more information about the data set, see Pace and Barry (1997).

NOTE: Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections "Using CAS Sessions and CAS Engine Librefs" on page 7 and "Loading a SAS Data Set onto a CAS Server" on page 7 in Chapter 2, "Shared Concepts."

You can load the sashelp.Vote1980 data set into your CAS session by using your CAS engine libref with the following DATA step:

```
data mylib.Vote1980;
   set sashelp.Vote1980;
run;
```

These statements assume that your CAS engine libref is named mylib, as in the section "Using CAS Sessions and CAS Engine Librefs" on page 583, but you can substitute any appropriately defined CAS engine libref.

The data table contains 3,107 observations and seven variables. The dependent variable LogVoteRate is the logarithm transformation of the proportion of the county population who voted for any candidate. The six explanatory variables are the number of people in the county 18 years of age or older (Pop), the number of people in the county who have a 12th-grade or higher education (Edu), the number of owned housing units (Houses), the aggregate income (Income), and the scaled longitude and latitude of geographic centroids (Longitude and Latitude).

The following statements produce a plot of LogVoteRate with respect to the geographic locations Longitude and Latitude:

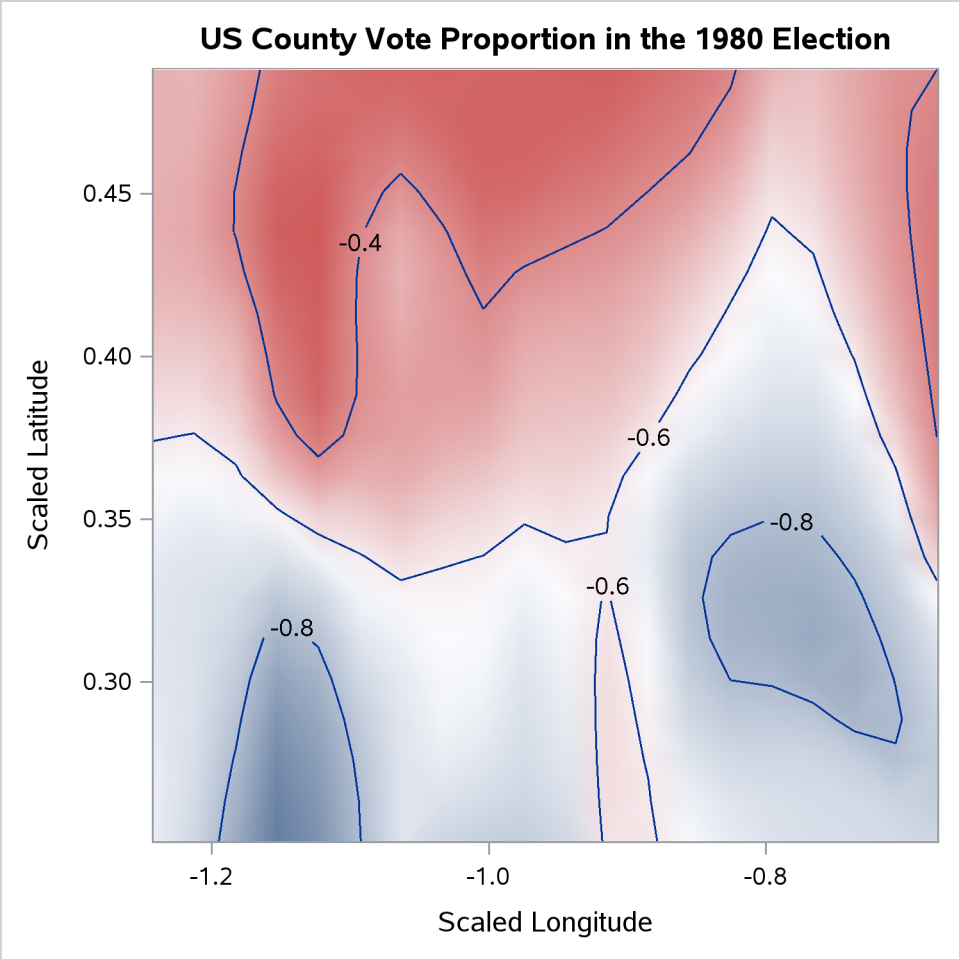
```
%let off0 = offsetmin=0 offsetmax=0
            linearopts=(thresholdmin=0 thresholdmax=0);
proc template;
   define statgraph surface;
      dynamic _title _z;
      begingraph / designwidth=defaultDesignHeight;
         entrytitle _title;
         layout overlay / xaxisopts=(&off0) yaxisopts=(&off0);
            contourplotparm z=_z y=Latitude x=Longitude / gridded=FALSE;
         endlayout;
      endgraph;
   end;
run;

proc sgrender data=mylib.Vote1980 template=surface;
   dynamic _title = 'US County Vote Proportion in the 1980 Election'
           _z     = 'LogVoteRate';
```

```
run;
```

Figure 12.1 shows the map of the logarithm transformation of the proportion of the county population who voted for any candidate in the 1980 US presidential election.

Figure 12.1 US County Vote Proportion in the 1980 Election



The objective is to explore the nonlinear dependency structure between the dependent variable and the demographic variables (Pop, Edu, Houses, and Income), in addition to the nonlinear dependency structure between the response and the Longitude and Latitude variables. The following statements use thin-plate regression splines to fit a generalized additive model:

```
ods graphics on;

proc gammod data=mylib.Vote1980 plots seed=12345;
   model LogVoteRate = spline(Pop    ) spline(Edu) spline(Houses)
                       spline(Income) spline(Longitude Latitude);
   output out=mylib.VotePred copyvars=(Longitude Latitude);
run;
```

With ODS Graphics enabled by the first statement, the PLOTS option in the PROC GAMMOD statement requests a smoothing component panel of fitted spline terms. The SEED option specifies the random seed so that you can reproduce the analysis.

The default output from this analysis is presented in Figure 12.2 through Figure 12.9.

Figure 12.2 displays the "Model Information" table. The response variable LogVoteRate is modeled by using a normal distribution whose mean is modeled by an identity link function. The GAMMOD procedure uses the performance iteration method and the generalized cross validation (GCV) criterion as the fitting criterion. PROC GAMMOD searches for the optimum smoothing parameters by using the Newton-Raphson algorithm to optimize the fitting criterion. The random number seed is set to 12,345. Random number generation is used for sampling from observations to form spline knots and truncated eigendecomposition. Changing the random number seed might yield slightly different model fits.

Figure 12.2 Model Information

The GAMMOD Procedure

Model Information	
Data Source	VOTE1980
Response Variable	LogVoteRate
Distribution	Normal
Link Function	Identity
Fitting Method	Performance Iteration
Fitting Criterion	GCV
Optimization Technique for Smoothing	Newton-Raphson
Random Number Seed	12345

Figure 12.3 displays the "Number of Observations" table. All 3,107 observations in the data table are used in the analysis. For data tables that have missing or invalid values, the number of used observations might be less than the number of observations read.

Figure 12.3 Number of Observations

Number of Observations Read	3107
Number of Observations Used	3107

Figure 12.4 displays the convergence status of the performance iteration method.

Figure 12.4 Convergence Status

The performance iteration converged after 2 steps.

Figure 12.5 shows the "Fit Statistics" table. The penalized log likelihood and the roughness penalty are displayed. You can use effective degrees of freedom to compare generalized additive models with generalized linear models that do not have spline transformations. Information criteria such as Akaike's information criterion (AIC), Akaike's bias-corrected information criterion (AICC), and Schwarz Bayesian information criterion (BIC) can also be used for comparisons. These criteria penalize the –2 log likelihood for effective degrees of freedom. The GCV criterion is used to compare against other generalized additive models or models that are penalized.

Figure 12.5 Fit Statistics

Fit Statistics	
Penalized Log Likelihood	2729.51482
Roughness Penalty	0.25787
Effective Degrees of Freedom	48.70944
Effective Degrees of Freedom for Error	3055.44725
AIC (smaller is better)	-5361.86863
AICC (smaller is better)	-5360.28467
BIC (smaller is better)	-5067.59479
GCV (smaller is better)	0.01042

.

The "Parameter Estimates" table in Figure 12.6 shows the regression parameter and dispersion parameter estimates. In this model, the intercept is the only regression parameter because (1) all variables are characterized by spline terms and no parametric effects are present and (2) the intercept absorbs the constant effect that is extracted from each spline term to make fitted splines identifiable. The dispersion parameter is estimated by maximizing the likelihood, given other model parameters.

Figure 12.6 Regression Parameter Estimates

Parameter Estimates					
Parameter	**DF**	**Estimate**	**Standard Error**	**Chi-Square**	**Pr > ChiSq**
Intercept	1	-0.576234	0.001803	102119.645	<.0001
Dispersion	1	0.010103	0.014287		

The "Estimates for Smoothing Components" table is shown in Figure 12.7. For each spline term, the effective degrees of freedom, the estimated smoothing parameter, and the corresponding roughness penalty are displayed. The table also displays additional information about spline terms, such as the number of parameters, penalty matrix rank, and number of spline knots.

Figure 12.7 Estimates for Smoothing Components

Estimates for Smoothing Components						
Component	**Effective DF**	**Smoothing Parameter**	**Roughness Penalty**	**Number of Parameters**	**Rank of Penalty Matrix**	**Number of Knots**
Spline(Pop)	7.80559	0.0398	0.0114	9	10	2000
Spline(Edu)	7.12453	0.2729	0.0303	9	10	2000
Spline(Houses)	7.20940	0.1771	0.0370	9	10	2000
Spline(Income)	5.92854	0.7498	0.0488	9	10	2000
Spline(Longitude Latitude)	18.64138	0.000359	0.1304	19	20	2000

Figure 12.8 displays the hypothesis testing results for each smoothing component. The null hypothesis for each spline term is whether the total dependency on each variable is 0. The effective degrees of freedom for fit and the effective degrees of freedom for test are displayed.

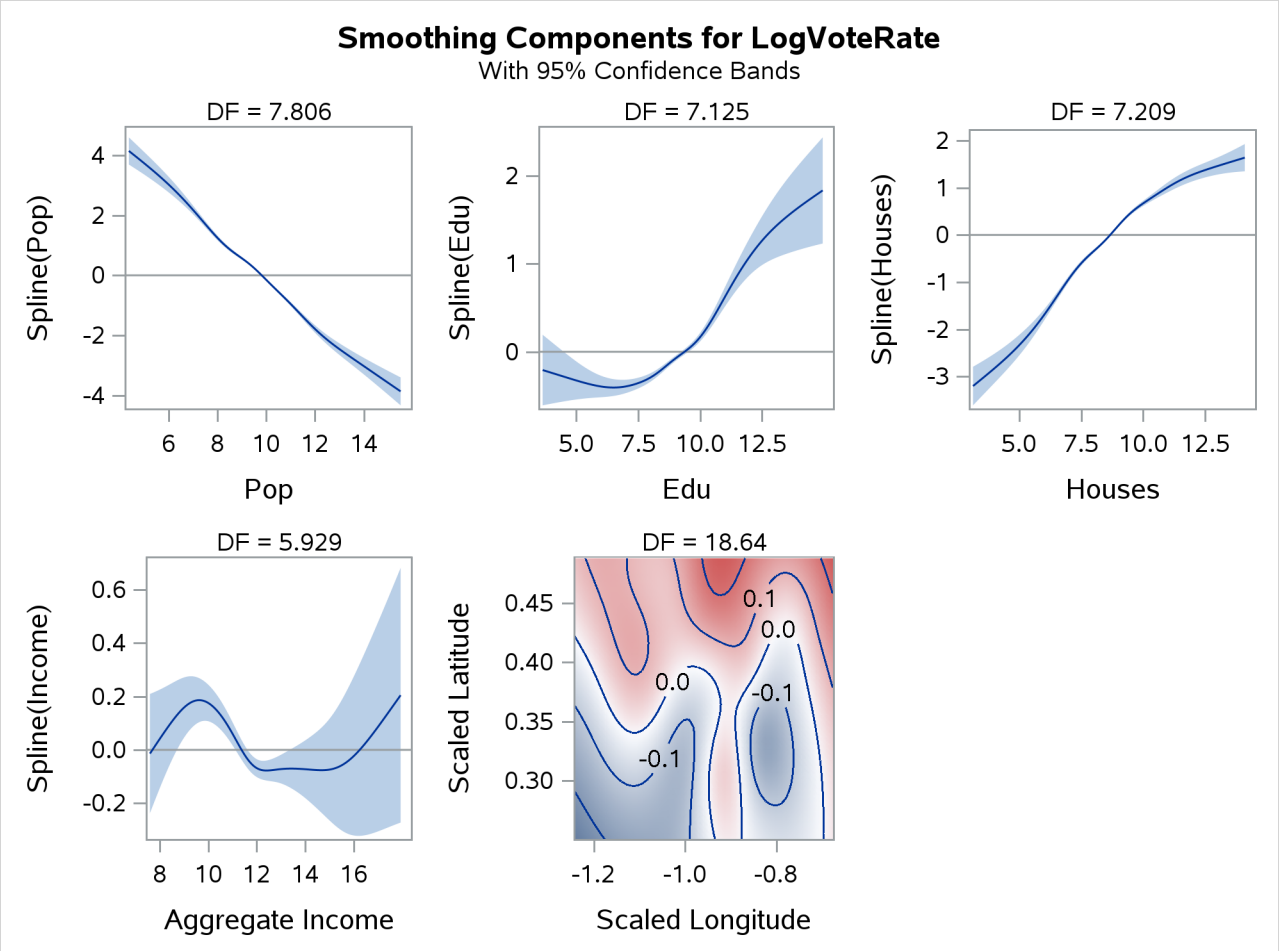
Figure 12.8 Tests for Smoothing Components

Tests for Smoothing Components

Component	Effective DF	Effective DF for Test	F Value	Pr > F
Spline(Pop)	7.80559	8	1443.64	<.0001
Spline(Edu)	7.12453	8	153.94	<.0001
Spline(Houses)	7.20940	8	1213.94	<.0001
Spline(Income)	5.92854	7	43.17	<.0001
Spline(Longitude Latitude)	18.64138	19	1619.15	<.0001

Figure 12.9 displays the "Smoothing Component Panel" for all the spline terms used in the model. It displays predicted spline curves and 95% Bayesian posterior confidence bands for each univariate spline term.

Figure 12.9 Smoothing Component Panel

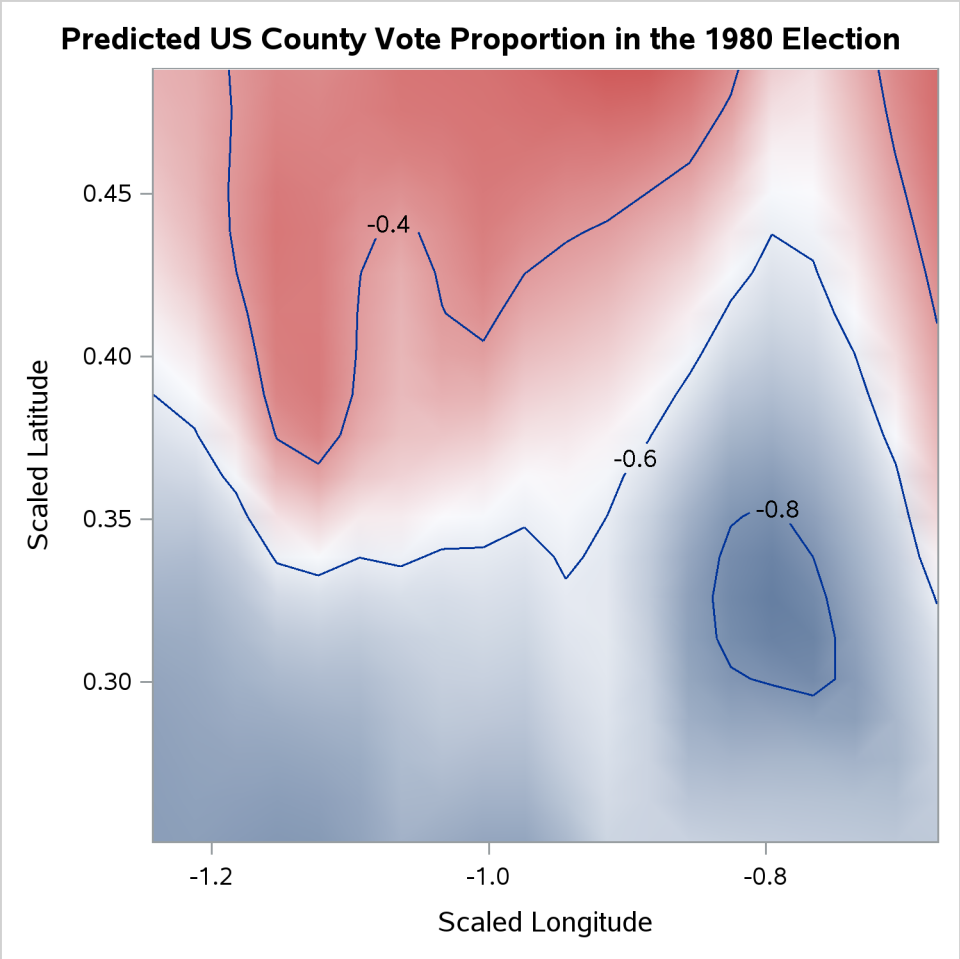


The preceding program requests the prediction for each observation by default, and the OUTPUT statement saves the results in the data table mylib.VotePred. The COPYVARS= option in the OUTPUT statement copies the specified variables (Longitude and Latitude) from the input data table, mylib.Vote1980, to the output data table, mylib.VotePred. The following run of the SGRENDER procedure produces the fitted surface of the log vote proportion in the 1980 presidential election:

```
proc sgrender data=mylib.VotePred template=surface;
   dynamic _title='Predicted US County Vote Proportion in the 1980 Election'
           _z    ='Pred';
run;
```

Figure 12.10 shows the map of predictions of the logarithm transformation of the proportion of county population who voted for any candidates in the 1980 US presidential election from the fitted generalized additive model.

Figure 12.10 Predicted US County Vote Proportion in the 1980 Election



Compared to the map of the logarithm transformations of the proportion of votes cast shown in Figure 12.1, the map of the predictions of the logarithm transformations of the proportion of votes cast has a smoother surface.

Syntax: GAMMOD Procedure

The following statements are available in the GAMMOD procedure:

PROC GAMMOD < *options* > **;**
BY *variables* **;**
CLASS *variable* < **(***options***)** >. . . < *variable* < **(***options***)** > > < / *global-options* > **;**
DISPLAY < *table-list* > < / *options* > **;**
DISPLAYOUT *table-spec-list* < / *options* > **;**
FREQ *variable* **;**
MODEL *response* < **(***response-options***)** > **=** < **PARAM(***effects***)** >
< *spline-effects* > < / *model-options* > **;**
MODEL *events* / *trials* **=** < **PARAM(***effects***)** > < *spline-effects* > < / *model-options* > **;**
OUTPUT OUT=*libref.data-table*
< **ALPHA=***number* > < **COMPONENT** > < **COPYVARS=(***variables***)** >
< *keyword* < **=***name* > >. . . < *keyword* < **=***name* > > **;**
STORE < **OUT=** >*libref.data-table* **;**
WEIGHT *variable* **;**

The PROC GAMMOD statement and a single MODEL statement are required. The CLASS statement can appear multiple times. If a CLASS statement is specified, it must precede the MODEL statement. The following sections describe the PROC GAMMOD statement and then describe the other statements in alphabetical order.

PROC GAMMOD Statement

PROC GAMMOD < *options* > **;**

The PROC GAMMOD statement invokes the procedure. Table 12.3 summarizes the available options in the PROC GAMMOD statement by function. The options are then described fully in alphabetical order.

Table 12.3 PROC GAMMOD Statement Options

Option	Description
Basic Options	
ALPHA=	Specifies a global significance level
DATA=	Specifies the input data table
SEED=	Sets the seed for pseudorandom number generation
Display Options	
ITDETAILS	Displays the “Iteration History” table
NOCLPRINT	Limits or suppresses the display of classification variable levels
NOPRINT	Suppresses ODS output
PLOTS=	Controls plots that are produced through ODS Graphics
Optimization Subject Options	
PLIKEOPTIONS	Sets optimization parameters for likelihood estimation

Table 12.3 *continued*

Option	Description
SMOOTHOPTIONS	Sets optimization parameters for smoothing parameter estimation
Tolerance Options	
SINGCHOL=	Tunes the singularity criterion for Cholesky decompositions
SINGULAR=	Tunes the general singularity criterion

You can specify the following *options* in the PROC GAMMOD statement.

ALPHA=*number*

specifies a global significance level for the hypothesis testing of smoothing components and the construction of Bayesian confidence bands of predictions. The confidence level is 1 – *number*. The value of *number* must be between 0 and 1; the default is 0.05. You can override this global significance level for Bayesian confidence bands of predictions by specifying the ALPHA= option in the OUTPUT statement.

DATA=*libref.data-table*

names the input data table for PROC GAMMOD to use. The default is the most recently created data table. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to the data, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 583.

data-table specifies the name of the input data table.

ITDETAILS

adds to the "Iteration History" table the current values of the parameter estimates and their gradients. If the optimization algorithm is used to determine at least one smoothing parameter, the table lists values for smoothing parameters. If all smoothing parameters are fixed or a parametric generalized linear model is specified, the table lists values for regression parameters. These quantities are reported only for parameters that participate in the optimization.

NOCLPRINT< =*number* **>**

suppresses the display of the "Class Level Information" table if you do not specify *number*. If you specify *number*, the values of the classification variables are displayed for only those variables whose number of levels is less than *number*. Specifying a *number* helps reduce the size of the "Class Level Information" table if some classification variables have a large number of levels.

NOPRINT

suppresses the generation of ODS output.

PLIKEOPTIONS(*optimization-parameters***)**
specifies optimization parameters for either maximum or penalized likelihood estimation. For more information about which *optimization-parameters* you can specify, see the section "Optimization Parameters" on page 593. For the performance iteration method, only the ABSFCONV=, FCONV=, and MAXITER= options are effective for determining when the performance iteration should have converged. If the distribution family involves a dispersion parameter that needs to be estimated by the maximum likelihood method, all the optimization parameters in this option are applicable.

PLOTS < (*global-plot-option***) > < = *plot-requests* < (***option***) > >**
controls the plots that are produced through ODS Graphics. When ODS Graphics is enabled, PROC GAMMOD produces by default a panel of plots of partial prediction curves or surfaces of smoothing components.

```
ods graphics on;

proc gammod plots;
   model y=spline(x1) spline(x2);
run;

ods graphics off;
```

You can specify the following *global-plot-option*, which applies to the smoothing component plots that the GAMMOD procedure generates:

UNPACK | UNPACKPANEL
suppresses paneling. By default, multiple smoothing component plots can appear in some output panels. Specify UNPACK to get each plot individually.

You can specify the following *plot-requests*:

ALL
requests that all default plots be produced.

COMPONENTS < (*component-option***) >**
plots a panel of smoothing components of the fitted model. You can specify the following *component-option*:

COMMONAXES
requests that the smoothing component plots use a common vertical axis except for bivariate contour plots. This option enables you to visually judge the relative effect size.

NONE
suppresses all plots.

SEED=*number*
specifies an integer that is used to start the pseudorandom number generator for truncated eigendecomposition and for subset sampling from observations to form knots if necessary. If you do not specify this option or if *number* ≤ 0, the seed is generated from the time of day, which is read from the computer's clock.

SINGCHOL=*number*
tunes the singularity criterion in Cholesky decomposition and matrix inversion operations. The default is 1E4 times the machine epsilon; this product is approximately 1E–12 on most computers.

SINGULAR=*number*
tunes the singularity criterion in truncated eigendecomposition to determine its convergence. The default is 1E4 times the machine epsilon; this product is approximately 1E–12 on most computers.

SMOOTHOPTIONS(*optimization-parameters***)**
specifies optimization parameters for smoothing parameter estimation. For more information about which *optimization-parameters* you can specify, see the next section. For parametric generalized linear models or generalized additive models that have fixed smoothing parameters, any optimization parameters that you specify for this option are ignored.

Optimization Parameters

You can specify *optimization-parameters* for both the PLIKEOPTIONS and SMOOTHOPTIONS options. Depending on the modeling context, some optimization parameters might have no effect. The optimization algorithm is considered to have converged when any one of the convergence criteria that are specified in *optimization-parameters* is satisfied. Table 12.4 lists the available optimization parameters for both the PLIKEOPTIONS and SMOOTHOPTIONS options.

Table 12.4 Optimization Parameters

Option	Description
ABSCONV=	Tunes the absolute function convergence criterion
ABSFCONV=	Tunes the absolute function difference convergence criterion
ABSGCONV=	Tunes the absolute gradient convergence criterion
FCONV=	Tunes the relative function difference convergence criterion
GCONV=	Tunes the relative gradient convergence criterion
MAXFUNC=	Specifies the maximum number of function evaluations in any optimization
MAXITER=	Chooses the maximum number of iterations in any optimization
MAXTIME=	Specifies the upper limit of CPU time (in seconds) for any optimization
MINITER=	Specifies the minimum number of iterations in any optimization
TECHNIQUE=	Selects the optimization technique

The *optimization-parameters* are fully described in the section "Optimization Options" on page 57 in Chapter 2, "Shared Concepts."

NOTE: By default, TECHNIQUE=NEWRAP for the performance iteration (METHOD=PERFORMANCE), and TECHNIQUE=QUANEW for the outer iteration (METHOD=OUTER).

BY Statement

BY *variables* **;**

You can specify a BY statement in PROC GAMMOD to obtain separate analyses of observations in groups that are defined by the values of the BY variables. If you specify more than one BY statement, only the last one specified is used. For more information, see the discussion of BY-group processing in the "Grouping Data" section of *SAS Programmers Guide: Essentials*.

CLASS Statement

CLASS *variable* < **(***options***)** >. . . < *variable* < **(***options***)** > > < / *global-options* > **;**

The CLASS statement names the classification variables to be used as explanatory variables in the analysis. You can list the response variable for binary models in the CLASS statement, but this is not required. Table 12.5 summarizes the values that you can use for either an *option* or a *global-option*. The options are fully documented in the section "CLASS Statement" on page 9 in Chapter 2, "Shared Concepts."

Table 12.5 CLASS Statement Options

Option	Description
DESCENDING	Reverses the sort order
MISSING	Treats missing values as valid levels
ORDER=	Specifies the sort order for the levels
PARAM=	Specifies the parameterization of the variable
REF=	Specifies the reference level of the variable

DISPLAY Statement

DISPLAY < *table-list* > < / *options* > **;**

The DISPLAY statement enables you to specify a list of display tables to display or exclude. This statement is similar to the ODS SELECT, ODS EXCLUDE, and ODS TRACE statements. However, the DISPLAY statement can improve performance when a large number of tables could be generated (such as in BY-group processing). The procedure processes the DISPLAY statement on a CAS server and thus sends only a subset of ODS tables to the SAS client. Because ODS statements are processed on a SAS client, first all the generated display tables are sent to the client, and then the client creates a subset.

If you use both DISPLAY and ODS statements together, the DISPLAY statement takes precedence over the ODS statements. Note that the ODS EXCLUDE statement processes tables that are sent to the client after they have been filtered by the DISPLAY statement. In some cases, it might appear that the ODS EXCLUDE statement is taking precedence because it can further filter the tables. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

You can specify the *table-list* as a list of table names, paths, partial pathnames, and regular expressions.

The table names that you can specify are listed in the section "ODS Table Names" on page 621. A path is a table name that is prefixed with dot-separated grouping information. For example, a SelectionSummary table that a procedure produces during a selection routine might have the path *Bygroup1.Summary.SelectionSummary*. A partial pathname does not include all groups; for example, *SelectionSummary* and *Summary.SelectionSummary* are partial pathnames for *Bygroup1.Summary.SelectionSummary*.

When you specify a table name or partial pathname, all display tables whose paths end in the specified name are selected for display or exclusion. For example, both *SelectionSummary* and *Summary.SelectionSummary* select *Bygroup1.Summary.SelectionSummary*.

A regular expression is enclosed in forward slashes (/). For example, specifying "/tions/" selects all pathnames that contain the substring "tions"; in particular, the *Bygroup1.Summary.SelectionSummary* table is selected. Specifying "!/tions/" selects all pathnames that do not contain the substring "tions"; in particular, the *Bygroup1.Summary.SelectionSummary* table is not selected.

You can specify the following *options* after a slash (/):

CASESENSITIVE
performs a case-sensitive comparison of table names in the *table-list* to display table names when tables are subsetted for display. To preserve case, you must enclose table names in the *table-list* in quotation marks.

EXCLUDE
displays all display tables except those that you specify in the *table-list*.

EXCLUDEALL
suppresses display of all tables. This option takes precedence over the other options.

TRACE
displays the display table names, labels, and paths.

DISPLAYOUT Statement

DISPLAYOUT *table-spec-list* < / *options* > **;**

The DISPLAYOUT statement enables you to create output tables from your displayed output. This statement is similar to the ODS OUTPUT statement. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

The *table-spec-list* specifies a list of output tables to create. Each entry in the list has either a *key=value* format or a *key* format:

key=value specifies *key* as the ODS table name, path, or partial pathname, and specifies *value* as the output table name.

key specifies *key* as the ODS table name and also as the output table name.

The ODS table names that you can specify are listed in the section "ODS Table Names" on page 621. You cannot specify the ODS table named OutputCasTables in the *table-spec-list*.

Table names and partial pathnames are discussed under the DISPLAY statement. The DISPLAYOUT statement does not support regular expressions.

You can specify the following *options* after a slash (/):

INCLUDEALL
creates output tables for all display tables. The name of the created output table is the same as the name of the corresponding display table. If you specify this option, the *table-spec-list* specification is ignored.

NOREPLACE
does not replace any existing output table of the same name.

REPEATED
replicates all output tables on all nodes.

FREQ Statement

FREQ *variable* **;**

The *variable* in the FREQ statement identifies a numeric variable in the input data table that contains the frequency of occurrence of each observation. PROC GAMMOD treats each observation as if it appears f times, where f is the value of the FREQ *variable* for the observation. If f is not an integer, it is truncated to an integer. If f is less than 1 or missing, the observation is not used in the analysis. When you omit the FREQ statement, each observation is assigned a frequency of 1.

MODEL Statement

MODEL *response* < **(***response-options***)** > **=** < **PARAM(***effects***)** > < *spline-effects* > < / *model-options* > **;**

MODEL *events* / *trials* **=** < **PARAM(***effects***)** > < *spline-effects* > < / *model-options* > **;**

The MODEL statement specifies the response (dependent or target) variable and the predictor (independent or explanatory) effects of the model. You can specify the response in the form of a single variable or in the form of a ratio of two variables, which are denoted *events/trials*. The first form applies to all distribution families; the second form applies only to summarized binomial response data. When you have binomial data, the *events* variable contains the number of positive responses (or events) and the *trials* variable contains the number of trials. The values of both *events* and (*trials* – *events*) must be nonnegative, and the value of *trials* must be positive. If you specify a single *response* variable that is in a CLASS statement, then the response is assumed to be binary.

You can specify parametric effects that are constructed from variables in the input data and include the effects in the parentheses of a PARAM( ) option, which can appear multiple times. For information about constructing the model effects, see the section "Specification and Parameterization of Model Effects" on page 64 in Chapter 2, "Shared Concepts."

You can specify *spline-effects* by including independent variables inside the parentheses of the SPLINE( ) option. Only continuous variables (not classification variables) can be specified in *spline-effects*. Each *spline-effect* can have at least one variable and optionally some *spline-options*. You can specify any number of *spline-effects*. The following table shows some examples.

Spline Effect Specification	Meaning
`spline(x)`	Constructs a univariate spline by using x and uses the observed data points as knots. The maximum degrees of freedom is 10. PROC GAMMOD uses an optimization algorithm to determine the optimal smoothing parameter.
`spline(x1/knots=list(1 to 10))`	Constructs a univariate spline by using x1 and a supplied list of knots from 1 to 10. PROC GAMMOD uses an optimization algorithm to determine the optimal smoothing parameter.
`spline(x2 x3/smooth=0.3)`	Constructs a bivariate spline by using x2 and x3 and a fixed smoothing parameter 0.3.
`spline(x4 x5 x6/maxdf=40)`	Constructs a trivariate spline by using x4, x5, and x6 and a maximum of 40 degrees of freedom. PROC GAMMOD uses an optimization algorithm to determine the optimal smoothing parameter.

Both parametric effects and spline effects are optional. If none are specified, a model that contains only an intercept is fitted. If only parametric effects are present, PROC GAMMOD fits a parametric generalized linear model by using the terms inside the parentheses of all PARAM( ) terms. If only spline effects are present, PROC GAMMOD fits a nonparametric additive model. If both types of effects are present, PROC GAMMOD fits a semiparametric model by using the parametric effects as the linear part of the model.

There are three sets of options in the MODEL statement. The *response-options* determine how the GAMMOD procedure models probabilities for binary data. The *spline-options* controls how each spline term forms basis expansions. The *model-options* control other aspects of model formation and inference. Table 12.6 summarizes these options, and subsequent sections describe them in detail.

Table 12.6 MODEL Statement Options

Option	Description
Response Variable Options	
DESCENDING	Reverses the response categories
EVENT=	Specifies the event category
ORDER=	Specifies the sort order
REF=	Specifies the reference category
Spline Effect Options	
DETAILS	Requests detailed spline information
DF=	Specifies the fixed degrees of freedom
INITSMOOTH=	Specifies the starting value for the smoothing parameter
KNOTS=	Specifies the knots to use for constructing the spline
M=	Specifies polynomial orders for constructing the spline
MAXDF=	Specifies the maximum degrees of freedom
MAXKNOTS=	Specifies the maximum number of knots to use for constructing the spline
MAXSMOOTH=	Specifies the upper bound for the smoothing parameter

Table 12.6 *continued*

Option	Description
MINSMOOTH=	Specifies the lower bound for the smoothing parameter
SMOOTH=	Specifies a fixed smoothing parameter
Model Options	
ALLOBS	Requests all nonmissing values of spline variables for constructing spline basis functions regardless of other model variables
CRITERION=	Specifies the model evaluation criterion
DISPERSION | PHI=	Specifies the fixed dispersion parameter
DISTRIBUTION | DIST=	Specifies the response distribution
FDHESSIAN	Requests a finite-difference Hessian for smoothing parameter selection
INITIALPHI=	Specifies the starting value of the dispersion parameter
LINK=	Specifies the link function
NORMALIZE	Requests normalized spline basis functions for model fitting
MAXPHI=	Specifies the upper bound for searching the dispersion parameter
METHOD=	Specifies the algorithm for selecting smoothing parameters
MINPHI=	Specifies the lower bound for searching the dispersion parameter
OFFSET=	Specifies the offset variable
RIDGE=	Specifies the ridge parameter
SCALE=	Specifies the method for estimating the dispersion parameter

Response Variable Options

Response variable options determine how the GAMMOD procedure models probabilities for binary data.

You can specify the following *response-options* by enclosing them in parentheses after the *response* variable.

DESCENDING

DESC

reverses the order of the response categories. If you specify both the DESCENDING and ORDER= options, PROC GAMMOD orders the response categories according to the ORDER= option and then reverses that order.

EVENT='*category***' | FIRST | LAST**

specifies the event category for the binary response model. PROC GAMMOD models the probability of the event category. This option has no effect when there are more than two response categories.

You can specify any of the following:

'*category*' specifies that observations whose value matches *category* (formatted, if a format is applied) in quotation marks represent events in the data. For example, the following statements specify that observations that have a formatted value of '1' represent events in the data. The probability that is modeled by the GAMMOD procedure is thus the probability that the variable def takes on the (formatted) value '1'.

```
proc gammod data=mylib.MyData;
   class A B C;
   model def(event ='1') = param(A B C) spline(x1 x2 x3);
run;
```

FIRST designates the first ordered category as the event.

LAST designates the last ordered category as the event.

By default, EVENT=FIRST.

ORDER=FORMATTED | FREQ | INTERNAL
specifies the sort order for the levels of the *response* variable. You can specify the following values:

FORMATTED sorts the levels by external formatted value, except for numeric variables that have no explicit format, which are sorted by their unformatted (internal) value. For numeric variables for which you have supplied no explicit format (that is, for which there is no corresponding FORMAT statement in the current PROC GAMMOD run or in the DATA step that created the data table), the levels are ordered by their internal (numeric) value. The sort order is machine-dependent.

FREQ sorts the levels by descending frequency count (levels that have the most observations come first in the order).

INTERNAL sorts the levels by unformatted value. The sort order is machine-dependent.

By default, ORDER=FORMATTED.

For more information about sort order, see the chapter about the SORT procedure in *Base SAS Procedures Guide* and the discussion of BY-group processing in *SAS Language Reference: Concepts*.

REF='*category*' **| FIRST | LAST**
specifies the reference category for the binary response model. Specifying one response category as the reference is the same as specifying the other response category as the event category. You can specify any of the following:

'*category*' specifies that observations whose value matches *category* (formatted, if a format is applied) are designated as the reference.

FIRST designates the first ordered category as the reference

LAST designates the last ordered category as the reference.

By default, REF=LAST.

Spline Effect Options

Spline effect options control how each spline term forms basis expansions.

You can specify the following *spline-effects* inside the parentheses of a SPLINE( ) term:

DETAILS
requests a detailed spline specification information table.

DF=*number*
specifies a fixed degrees of freedom. When you specify this option, no smoothing parameter selection is performed on the spline term. If *number* is not an integer, then *number* is truncated to an integer.

INITSMOOTH=*number*
specifies the starting value for a smoothing parameter. The *number* must be nonnegative.

KNOTS=*method*
specifies the *method* for supplying user-defined knot values instead of using data values for constructing basis expansions. You can use the following *methods* for supplying the knots:

LIST(*list-of-values***)**
specifies a list of values as knots for the spline construction. For a multivariate spline term, the listed values are taken as multiple row vectors, where each vector has values that are ordered by specified variables. If the last row vector of knots contains fewer values than the number of variables, then the last row vector is ignored. For example, the following specification of a spline term produces two actual knot vectors ($\boldsymbol{k}_1$ and $\boldsymbol{k}_2$) and the value 5 is ignored.

```
spline(x1 x2/knots=list(1 2 3 4 5))
```

Table 12.7 Knot Values for a Bivariate Spline with a Supplied List

	x1	x2
$\boldsymbol{k}_1$	1	2
$\boldsymbol{k}_2$	3	4

EQUAL(*n***)**
specifies the number of equally spaced interior knots for every variable in a spline term. Two boundary knots are automatically added to the knot list for each variable such that the total number of knots is $(n + 2)^d$, where d is the number of variables in the spline term. For a multivariate spline term, knot values for each variable are determined independently from the corresponding boundary values. For example, if the boundary points for x1 are 1 and 5 and the boundary points for x2 are 2 and 6, then the following specification of a spline term produces nine actual knots ($\boldsymbol{k}_1$ through $\boldsymbol{k}_9$), which consist of two boundary knots and one interior knot for each variable.

```
spline(x1 x2/knots=equal(1))
```

Table 12.8 Knot Values for a Bivariate Spline with One Interior Knot

	x1	x2
$\boldsymbol{k}_1$	1	2
$\boldsymbol{k}_2$	1	4
$\boldsymbol{k}_3$	1	6
$\boldsymbol{k}_4$	3	2
$\boldsymbol{k}_5$	3	4
$\boldsymbol{k}_6$	3	6
$\boldsymbol{k}_7$	5	2
$\boldsymbol{k}_8$	5	4
$\boldsymbol{k}_9$	5	6

M=*number*

specifies the order of the derivative in the penalty term. The *number* must be a positive integer. The default is $\max(2, \text{int}(d/2) + 1)$, where d is the number of variables in the spline term.

MAXDF=*number*

specifies the maximum *number* of degrees of freedom. When a thin-plate regression spline is formed, the specified *number* is used for constructing a low-rank penalty matrix to approximate the penalty matrix via the truncated eigendecomposition. The *number* must be greater than $\binom{m+d-1}{d}$, where m is the derivative order that is specified in the M= option, and d is the number of variables in the spline term. The default is $10 \times d$. For more information, see the section "Thin-Plate Regression Splines" on page 608.

MAXKNOTS=*number*

specifies the maximum *number* of knots if data points are used to form knots. If KNOTS=LIST(*list-of-values*) is not specified, PROC GAMMOD forms knots from unique data points. If the number of unique data points is greater than *number*, a subset of size *number* is formed by random sampling from all unique data points. The *number* cannot exceed the largest integer that can be stored on the CAS server. By default, MAXKNOTS=2000.

MAXSMOOTH=*number*

specifies the upper bound for the smoothing parameter. The default is the largest double-precision value.

MINSMOOTH=*number*

specifies the lower bound for the smoothing parameter. By default, MINSMOOTH=0.

SMOOTH=*number*

specifies a fixed smoothing parameter. When you specify this option, no smoothing parameter selection is performed on the spline term.

Model Options

You can specify the following *model-options* in the MODEL statement after a slash (/):

ALLOBS
requests that all nonmissing values of the variables be used in a spline term for constructing the spline basis functions, regardless of whether other model variables are missing.

CRITERION=*criterion*
specifies the model evaluation criterion for selecting smoothing parameters for *spline-effects*. You can specify the following values:

GACV< (FACTOR=*number* | **GAMMA=***number*) >
uses the generalized approximate cross validation (GACV) criterion to evaluate models.

GCV< (FACTOR=*number* | **GAMMA=***number*) >
uses the generalized cross validation (GCV) criterion to evaluate models.

UBRE< (FACTOR=*number* | **GAMMA=***number*) >
uses the unbiased risk estimator (UBRE) criterion to evaluate models.

The default *criterion* depends on the value of the DISTRIBUTION= option. For distributions that involve dispersion parameters, GCV is the default. For distributions without dispersion parameters, UBRE is the default. For all three criteria, you can optionally use the FACTOR= option to specify an extra tuning parameter in order to penalize more for model roughness. The value of *number* must be greater than or equal to 1. For more information about the model evaluation criteria, see the section "Model Evaluation Criteria" on page 612.

DISPERSION=*number*
PHI=*number*
specifies a fixed dispersion parameter for distributions that have a dispersion parameter. The dispersion parameter that is used in all computations is fixed at *number*; it is not estimated.

DISTRIBUTION=*keyword*
specifies the response distribution for the model. The *keywords* and their associated distributions are shown in Table 12.9.

Table 12.9 Built-In Distribution Functions

DISTRIBUTION=	Distribution Function
BINARY | BERNOULLI	Binary
BINOMIAL	Binary or binomial
GAMMA	Gamma
INVGAUSS | IG | IGAUSSIAN	Inverse Gaussian
NB | NEGBIN | NEGBINOMIAL	Negative binomial
NORMAL | GAUSSIAN | GAUSS	Normal
POISSON | POI	Poisson
TWEEDIE< (*Tweedie-options***) >**	Tweedie

When DISTRIBUTION=TWEEDIE, you can specify the following *Tweedie-options*:

INITIALP=
specifies a starting value for iterative estimation of the Tweedie power parameter.

P=
requests a fixed Tweedie power parameter.

If you do not specify a link function in the LINK= option, a default link function is used. The default link function for each distribution is shown in Table 12.10. You can use any link function shown in Table 12.11 by specifying the LINK= option. Other commonly used link functions for each distribution are shown in Table 12.10.

Table 12.10 Default and Commonly Used Link Functions

DISTRIBUTION=	Default Link Function	Other Commonly Used Link Functions
BINARY | BERNOULLI	Logit	Probit, complementary log-log, log-log
BINOMIAL	Logit	Probit, complementary log-log, log-log
GAMMA	Log	Reciprocal
INVGAUSS | IG | IGAUSSIAN	Log	Reciprocal square
NB | NEGBIN | NEGBINOMIAL	Log	
NORMAL | GAUSSIAN | GAUSS	Identity	Log
POISSON | POI	Log	
TWEEDIE	Log	

FDHESSIAN
requests that the second-order derivatives (Hessian) be computed using finite-difference approximations based on evaluation of the first-order derivatives (gradient). This option might be useful if the analytical Hessian takes a long time to compute.

INITIALPHI=*number*
specifies a starting value for iterative maximum likelihood estimation of the dispersion parameter for distributions that have a dispersion parameter.

LINK=*keyword*
specifies the link function for the model. The *keywords* and the associated link functions are shown in Table 12.11. Default and commonly used link functions for the available distributions are shown in Table 12.10.

Table 12.11 Built-In Link Functions

LINK=	Link Function	$g(\mu) = \eta =$
CLOGLOG | CLL	Complementary log-log	$\log(-\log(1-\mu))$
IDENTITY | ID	Identity	μ
INVERSE | RECIP | RECIPROCAL	Reciprocal	$1/\mu$
INV2 | POWERMINUS2 | POWMINUS2	Reciprocal square	$1/\mu^2$
LOG	Logarithm	$\log(\mu)$
LOGIT	Logit	$\log(\mu/(1-\mu))$
LOGLOG	Log-log	$-\log(-\log(\mu))$
PROBIT | NORMIT	Probit	$\Phi^{-1}(\mu)$

$\Phi^{-1}(\cdot)$ denotes the quantile function of the standard normal distribution.

MAXPHI=*number*

specifies an upper bound for maximum likelihood estimation of the dispersion parameter for distributions that have a dispersion parameter.

METHOD=OUTER | PERFORMANCE

specifies the algorithm for selecting smoothing parameters for *spline-effects*. You can specify the following values:

OUTER specifies the outer iteration method for selecting smoothing parameters. For more information about this method, see the section "Outer Iteration" on page 613.

PERFORMANCE specifies the performance iteration method for selecting smoothing parameters. For more information about this method, see the section "Performance Iteration" on page 614.

By default, METHOD=PERFORMANCE.

MINPHI=*number*

specifies a lower bound for maximum likelihood estimation of the dispersion parameter for distributions that have a dispersion parameter.

NORMALIZE

requests normalized spline basis functions for model fitting. After the regression spline basis functions are computed, each column is standardized to have a unit standard error. The corresponding penalty matrix is also scaled accordingly. This option might be helpful when you have badly scaled data.

OFFSET=*variable*

specifies a *variable* to be used as an offset to the linear predictor. An offset plays the role of an effect whose coefficient is known to be 1. The offset variable cannot appear in the CLASS statement or elsewhere in the MODEL statement. Observations that have missing values for the offset variable are excluded from the analysis.

RIDGE=*number*
allows a ridge parameter such that a diagonal matrix $\mathbf{H} : \mathbf{H}_{ii} = number$ is added to the optimization problem with respect to regression parameters:

$$\min(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})'(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + \boldsymbol{\beta}'\mathbf{S}_{\boldsymbol{\lambda}}\boldsymbol{\beta} + \boldsymbol{\beta}'\mathbf{H}\boldsymbol{\beta} \quad \text{with respect to} \quad \boldsymbol{\beta}$$

By default, RIDGE=0. Specifying a small ridge parameter might be helpful if the model matrix $\mathbf{X}'\mathbf{X} + \mathbf{S}_{\boldsymbol{\lambda}}$ is close to singular.

SCALE=DEVIANCE | MLE | PEARSON
specifies the method for estimating the scale and dispersion parameters. You can specify the following values:

DEVIANCE
estimates the dispersion parameter by using the deviance statistic.

MLE
computes the dispersion parameter by maximizing the likelihood or penalized likelihood.

PEARSON
estimates the dispersion parameter by using Pearson's statistic.

By default, SCALE=MLE.

OUTPUT Statement

OUTPUT OUT=*libref.data-table*
< **ALPHA=***number* > < **COMPONENT** > < **COPYVARS=***(variables)* >
< *keyword* < **=***name* > >. . . < *keyword* < **=***name* > > **;**

The OUTPUT statement creates a data table that contains observationwise statistics that PROC GAMMOD computes after fitting the model. In order to avoid data duplication for large data tables, the variables in the input data table are *not* included in the output data table unless you specify them in the COPYVARS= option.

The computation of the output statistics is based on the final parameter estimates. If the model fit does not converge, missing values are produced for the quantities that depend on the estimates.

You must specify the following option:

OUT=*libref.data-table*
names the output data table for PROC GAMMOD to use. You must specify this option before any other options. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to where the data table is to be stored, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 583.

data-table specifies the name of the output data table.

You can also specify the following syntax elements:

ALPHA=*number*

specifies the significance level for the construction of confidence intervals in the output data table. The confidence level is $1 - number$. The value of *number* must be between 0 and 1. By default, *number* is equal to the value of the ALPHA= option in the PROC GAMMOD statement, or 0.05 if that option is not specified.

COMPONENT

requests componentwise statistics for all spline terms if LINP, LOWER, STD, or UPPER is specified as a *keyword*.

COPYVAR=*variable*

COPYVARS=(*variables***)**

transfers one or more *variables* from the input data table to the output data table.

keyword **<=***name***>**

specifies a statistic to include in the output data table and optionally assigns a *name* to the variable. If you do not provide a *name*, the GAMMOD procedure assigns a default name based on the *keyword*.

You can specify the following *keywords* for adding statistics to the OUTPUT data table:

LINP | XBETA

requests the linear predictor $\eta = \mathbf{x}'\boldsymbol{\beta}$. For observations in which only the response variable is missing, values of the linear predictor are computed even though these observations do not affect the model fit. The default *name* is Xbeta.

LOWER

requests a lower Bayesian confidence band value for the predicted value. The default *name* is Lower.

PEARSON | PEARS | RESCHI

requests the Pearson residual, $(y - \mu)/\sqrt{\mathrm{V}(\mu)}$, where μ is the estimate of the predicted response mean and $\mathrm{V}(\mu)$ is the response distribution variance function. The default *name* is Pearson.

PREDICTED | PRED | P

requests predicted values for the response variable. For observations in which only the response variable is missing, the predicted values are computed even though these observations do not affect the model fit. The default *name* is Pred.

RESIDUAL | RESID | R

requests the raw residual, $y - \mu$, where μ is the estimate of the predicted mean. The default *name* is Residual.

STD

requests a standard error for the linear predictor. The default *name* is Std.

UPPER

requests an upper Bayesian confidence band value for the predicted value. The default *name* is Upper.

STORE Statement

STORE < **OUT=** >*libref.data-table* **;**

The STORE statement saves the context and results of the statistical analysis. The resulting analytic store has a binary file format that cannot be modified. An example of analytic store technology is to perform a time-consuming analysis and to store its results by using the STORE statement. Later you can perform specific statistical analysis tasks based on the saved results of the previous analysis, without having to fit the model again. The contents of the analytic store that PROC GAMMOD produces can be processed by the ASTORE procedure, the **`gamScore`** action in the **`gam`** action set, and the **`score`** action in the **`aStore`** action set.

libref.data-table is a two-level name, where *libref* refers to the library, and *data-table* specifies the name of the output data table. For more information about this two-level name, see the section "Using CAS Sessions and CAS Engine Librefs" on page 583.

WEIGHT Statement

WEIGHT *variable* **;**

The *variable* in the WEIGHT statement is used as a weight to perform a weighted analysis of the data. Observations that have nonpositive or missing weights are not included in the analysis. If a WEIGHT statement is not included, then all observations that are used in the analysis are assigned a weight of 1.

Details: GAMMOD Procedure

Missing Values

Any observation that has missing values for the response, frequency, weight, offset, or explanatory variables is excluded from the analysis; however, missing values are valid for response and explanatory variables if you specify the MISSING option in the CLASS statement. Observations that have a nonpositive weight or a frequency less than 1 are also excluded. For Poisson, negative binomial, and Tweedie distributions, observations that have a negative response value are excluded. For gamma and inverse Gaussian distributions, observations that have a nonpositive response value are excluded.

The estimated linear predictor and the fitted means are not computed for any observation that has missing offset or explanatory variables. However, if only the response value is missing, the linear predictor and the fitted means can be computed and output to a data table by using the OUTPUT statement.

Thin-Plate Regression Splines

The GAMMOD procedure uses thin-plate regression splines (Wood 2003) to construct spline basis expansions. The thin-plate regression splines are based on thin-plate smoothing splines (Duchon 1976, 1977). Compared to thin-plate smoothing splines, thin-plate regression splines produce fewer basis expansions and thus make direct fitting of generalized additive models possible.

Thin-Plate Smoothing Splines

Consider the problem of estimating a smoothing function f of $\mathbf{x}$ with d covariates from n observations. The model assumes

$$y_i = f(\mathbf{x}_i) + \epsilon_i, i = 1, \ldots, n$$

Then the thin-plate smoothing splines estimate the smoothing function f by minimizing the penalized least squares function:

$$\sum_{i=1}^{n}(y_i - f(\mathbf{x}_i))^2 + \lambda J_{m,d}(f)$$

The penalty term $\lambda J_{m,d}(f)$ includes the function that measures roughness on the f estimate:

$$J_{m,d}(f) = \int \cdots \int \sum_{\alpha_1+\cdots+\alpha_d=m} \frac{m!}{\alpha_1! \cdots \alpha_d!} \left(\frac{\partial^m f}{\partial {x_1}^{\alpha_1} \cdots \partial {x_d}^{\alpha_d}} \right)^2 \mathrm{d}x_1 \cdots \mathrm{d}x_d$$

The parameter m (which corresponds to the M= option for a spline effect) specifies how the penalty is applied to the function roughness. Function derivatives whose order is less than m are not penalized. The relation $2m > d$ must be satisfied.

The penalty term also includes the smoothing parameter $\lambda \in [0, \infty)$, which controls the trade-off between the model's fidelity to the data and the function smoothness of f. When $\lambda = 0$, the function estimate corresponds to an interpolation. When $\lambda \to \infty$, the function estimate becomes the least squares fit. By using the defined penalized least squares criterion and a fixed λ value, you can explicitly express the estimate of the smooth function f in the following form:

$$f_\lambda(\mathbf{x}) = \sum_{j=1}^{M} \theta_j \phi_j(\mathbf{x}) + \sum_{i}^{n} \delta_i \eta_{m,d}(\|\mathbf{x} - \mathbf{x}_i\|)$$

In the expression of $f_\lambda(\mathbf{x})$, δ_i and θ_j are coefficients to be estimated. The functions $\phi_j(\mathbf{x})$ correspond to unpenalized polynomials of $\mathbf{x}$ with degrees up to $m - 1$. The total number of these polynomials is $M = \binom{m+d-1}{d}$. The function $\eta_{m,d}$ models the extra nonlinearity besides the polynomials and is a function of the Euclidean distance r between any $\mathbf{x}$ value and an observed $\mathbf{x}_i$ value:

$$\eta_{m,d}(r) = \begin{cases} \frac{(-1)^{m+1+d/2}}{2^{2m-1}\pi^{d/2}(m-1)!(m-d/2)!} r^{2m-d} \log(r) & \text{if } d \text{ is even} \\ \frac{\Gamma(d/2-m)}{2^{2m}\pi^{d/2}(m-1)!} r^{2m-d} & \text{if } d \text{ is odd} \end{cases}$$

Define the penalty matrix $\mathbf{E}$ such that each entry $E_{ij} = \eta_{m,d}(\|\mathbf{x}_i - \mathbf{x}_j\|)$, let $\mathbf{y}$ be the vector of the response, let $\mathbf{T}$ be the matrix where each row is formed by $\phi_j(\mathbf{x})$, and let $\boldsymbol{\theta}$ and $\boldsymbol{\delta}$ be vectors of coefficients θ_j and δ_i. Then you can obtain the function estimate f_λ from the following minimization problem:

$$\min \|\mathbf{y} - \mathbf{T}\boldsymbol{\theta} - \mathbf{E}\boldsymbol{\delta}\|^2 + \lambda \boldsymbol{\delta}'\mathbf{E}\boldsymbol{\delta} \quad \text{subject to} \quad \mathbf{T}'\boldsymbol{\delta} = \mathbf{0}$$

Low-Rank Approximation

Given the representation of the thin-plate smoothing spline, the estimate of f involves as many parameters as the number of unique data points. Solving $(\boldsymbol{\theta}, \boldsymbol{\delta})$ with an optimum λ becomes difficult for large problems.

Because the matrix $\mathbf{E}$ is symmetric and nonnegative definite, the eigendecomposition can be taken as $\mathbf{E} = \mathbf{U}\mathbf{D}\mathbf{U}'$, where $\mathbf{D}$ is the diagonal matrix of eigenvalues d_i of $\mathbf{E}$, and $\mathbf{U}$ is the matrix of eigenvectors that corresponds to d_i. The truncated eigendecomposition forms $\tilde{\mathbf{E}}_k$, which is an approximation to $\mathbf{E}$ such that

$$\tilde{\mathbf{E}}_k = \mathbf{U}_k \mathbf{D}_k \mathbf{U}_k'$$

where $\mathbf{D}_k$ is a diagonal matrix that contains the k most extreme eigenvalues in descending order of absolute values: $|\tilde{d}_1| > \cdots > |\tilde{d}_k|$. $\mathbf{U}_k$ is the matrix that is formed by columns of eigenvectors that correspond to the eigenvalues in $\mathbf{D}_k$.

The approximation $\tilde{\mathbf{E}}_k$ not only reduces the dimension from $n \times n$ of $\mathbf{E}$ to $n \times k$ but also is optimal in two senses. First, $\tilde{\mathbf{E}}_k$ minimizes the spectral norm $\|\mathbf{E} - \mathbf{F}_k\|_2$ between $\mathbf{E}$ and all rank k matrices $\mathbf{F}_k$. Second, $\tilde{\mathbf{E}}_k$ also minimizes the worst possible change that is introduced by the eigenspace truncation as defined by

$$\max_{\boldsymbol{\delta} \neq \mathbf{0}} \frac{\boldsymbol{\delta}'(\mathbf{E} - \mathbf{G}_k)\boldsymbol{\delta}}{\|\boldsymbol{\delta}\|^2}$$

where $\mathbf{G}_k$ is formed by any k eigenvalues and corresponding eigenvectors. For more information, see Wood (2003).

Now given $\mathbf{E} \approx \tilde{\mathbf{E}}_k$ and $\tilde{\mathbf{E}}_k = \mathbf{U}_k \mathbf{D}_k \mathbf{U}_k'$, and letting $\boldsymbol{\delta}_k = \mathbf{U}_k' \boldsymbol{\delta}$, the minimization problem becomes

$$\min \|\mathbf{y} - \mathbf{T}\boldsymbol{\theta} - \mathbf{U}_k \mathbf{D}_k \boldsymbol{\delta}_k\|^2 + \lambda \boldsymbol{\delta}_k' \mathbf{D}_k \boldsymbol{\delta}_k \quad \text{subject to} \quad \mathbf{T}'\mathbf{U}_k \boldsymbol{\delta}_k = \mathbf{0}$$

You can turn the constrained optimization problem into an unconstrained one by using any orthogonal column basis $\mathbf{Z}$. One way to form $\mathbf{Z}$ is via the QR decomposition of $\mathbf{U}_k'\mathbf{T}$:

$$\mathbf{U}_k'\mathbf{T} = [\mathbf{Q}_1 \mathbf{Q}_2] \begin{bmatrix} \mathbf{R} \\ \mathbf{0} \end{bmatrix}$$

Let $\mathbf{Z} = \mathbf{Q}_2$. Then it is verified that

$$\mathbf{T}'\mathbf{U}_k \mathbf{Z} = \mathbf{R}'\mathbf{Q}_1'\mathbf{Q}_2 = \mathbf{0}$$

So for $\boldsymbol{\delta}_k$ such that $\mathbf{T}'\mathbf{U}_k \boldsymbol{\delta}_k = \mathbf{0}$, it is true that $\boldsymbol{\delta}_k = \mathbf{Z}\tilde{\boldsymbol{\delta}}$. Now the problem becomes the unconstrained optimization,

$$\min \|\mathbf{y} - \mathbf{T}\boldsymbol{\theta} - \mathbf{U}_k \mathbf{D}_k \mathbf{Z}\tilde{\boldsymbol{\delta}}\|^2 + \lambda \tilde{\boldsymbol{\delta}}' \mathbf{Z}' \mathbf{D}_k \mathbf{Z} \tilde{\boldsymbol{\delta}}$$

Let

$$\boldsymbol{\beta} = \begin{bmatrix} \boldsymbol{\theta} \\ \tilde{\boldsymbol{\delta}} \end{bmatrix}, \quad \mathbf{X} = [\mathbf{T} : \mathbf{U}_k \mathbf{D}_k \mathbf{Z}], \quad \text{and} \quad \mathbf{S} = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{Z}'\mathbf{D}_k\mathbf{Z} \end{bmatrix}$$

The optimization is simplified as

$$\min \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda \boldsymbol{\beta}' \mathbf{S} \boldsymbol{\beta} \quad \text{with respect to} \quad \boldsymbol{\beta}$$

Generalized Additive Models

Generalized Linear Models

All probability distributions that the GAMMOD procedure fits are members of an exponential family of distributions. For the specification of an exponential family, see the section "Exponential Family Distributions" in Chapter 14, "GENSELECT Procedure."

Table 12.12 lists and defines some common notation that is used in the context of generalized linear models and generalized additive models.

Table 12.12 Common Notation

Notation	Meaning
ℓ	Log-likelihood
ℓ_p	Penalized log-likelihood
D	Deviance
D_p	Penalized deviance
g	Link function
g^{-1}	Inverse link function
$\boldsymbol{\mu}$	Response mean $\boldsymbol{\mu} = g^{-1}(\boldsymbol{\eta})$
$\boldsymbol{\eta}$	Linear predictor $\boldsymbol{\eta} = \mathbf{X}\boldsymbol{\beta}$
ϕ	Dispersion parameter
$\mathbf{z}$	Column of adjusted response variable
$\boldsymbol{\nu}$	Column of response variance
ω_i	Prior weight for each observation
w_i	Adjusted weight for each observation
$\mathbf{W}$	Diagonal matrix of adjusted weights

The GAMMOD procedure supports the following distributions: binary, binomial, gamma, inverse Gaussian, negative binomial, normal (Gaussian), Poisson, and Tweedie.

For forms of log likelihood functions for each of the probability distributions, see the section "Log-Likelihood Functions" in Chapter 14, "GENSELECT Procedure."

Generalized Additive Models

Generalized additive models are extensions of generalized linear models (Nelder and Wedderburn 1972). For each observation that has a response Y_i and a row vector of the model matrix $\mathbf{x}_i$, both generalized additive models and generalized linear models assume the model additivity,

$$g(\mu_i) = f_1(\mathbf{x}_{i1}) + \cdots + f_p(\mathbf{x}_{ip})$$

where $\mu_i = E(Y_i)$ and Y_i is independently distributed in some exponential family. Generalized linear models further assume model linearity by $f_j(\mathbf{x}_{ij}) = \mathbf{x}_{ij}\beta_j$ for $j = 1, \ldots, p$. Generalized additive models relax the linearity assumption by allowing some smoothing functions f_j to characterize the dependency. The GAMMOD procedure constructs the smoothing functions by using thin-plate regression splines.

Consider a generalized additive model that has some linear terms $\mathbf{X}_L$ with coefficients $\boldsymbol{\beta}_L$ and p smoothing functions f_j. Each smoothing function can be constructed by thin-plate regression splines with a smoothing

parameter λ_j. Using the notations described in the section "Low-Rank Approximation" on page 609, you can characterize each smoothing function by

$$\boldsymbol{\beta}_j = \begin{bmatrix} \boldsymbol{\theta}_j \\ \tilde{\boldsymbol{\delta}}_j \end{bmatrix}, \quad \mathbf{X}_j = [\mathbf{T}_j : \mathbf{U}_{kj}\mathbf{D}_{kj}\mathbf{Z}_j], \quad \text{and} \quad \mathbf{S}_j = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{Z}_j'\mathbf{D}_{kj}\mathbf{Z}_j \end{bmatrix}$$

Notice that each smoothing function representation contains a zero-degree polynomial that corresponds to a constant. Having multiple constant terms makes the smoothing functions unidentifiable. The solution is to include a global constant term (that is, the intercept) in the model and enforce the centering constraint to each smoothing function. You can write the constraint as

$$\mathbf{1}'\mathbf{X}_j\boldsymbol{\beta}_j = 0$$

By using a similar approach as the linear constraint for thin-plate regression splines, you obtain the orthogonal column basis $\mathbf{V}_j$ via the QR decomposition of $\mathbf{X}_j'\mathbf{1}$ such that $\mathbf{1}'\mathbf{X}_j\mathbf{V}_j = 0$. Each smoothing function can be reparameterized as $\tilde{\mathbf{X}}_j = \mathbf{X}_j\mathbf{V}_j$.

Let $\mathbf{X} = [\mathbf{X}_L : \tilde{\mathbf{X}}_1 : \cdots : \tilde{\mathbf{X}}_p]$ and $\boldsymbol{\beta}' = [\boldsymbol{\beta}_L' : \boldsymbol{\beta}_1' : \cdots : \boldsymbol{\beta}_p']$. Then the generalized additive model can be represented as $g(\boldsymbol{\mu}) = \mathbf{X}\boldsymbol{\beta}$. The roughness penalty matrix is represented as a block diagonal matrix:

$$\mathbf{S}_{\boldsymbol{\lambda}} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \lambda_1\mathbf{S}_1 & \cdots & \mathbf{0} \\ & & \ddots & \\ \vdots & \vdots & \lambda_j\mathbf{S}_j & \vdots \\ & & & \ddots \\ \mathbf{0} & \mathbf{0} & \cdots & \lambda_p\mathbf{S}_p \end{bmatrix}$$

Then the roughness penalty is measured in the quadratic form $\boldsymbol{\beta}'\mathbf{S}_{\boldsymbol{\lambda}}\boldsymbol{\beta}$.

Penalized Likelihood Estimation

Given a fixed vector of smoothing parameters, $\boldsymbol{\lambda} = [\lambda_i \ldots \lambda_p]'$, you can fit the generalized additive models by the penalized likelihood estimation. In contrast to the maximum likelihood estimation, penalized likelihood estimation obtains an estimate for $\boldsymbol{\beta}$ by maximizing the penalized log likelihood,

$$\ell_p(\boldsymbol{\beta}) = \ell(\boldsymbol{\beta}) - \frac{1}{2}\boldsymbol{\beta}'\mathbf{S}_{\boldsymbol{\lambda}}\boldsymbol{\beta}$$

Any optimization technique that you can use for maximum likelihood estimation can also be used for penalized likelihood estimation. If first-order derivatives are required for the optimization technique, you can compute the gradient as

$$\frac{\partial \ell_p}{\partial \boldsymbol{\beta}} = \frac{\partial \ell}{\partial \boldsymbol{\beta}} - \mathbf{S}_{\boldsymbol{\lambda}}\boldsymbol{\beta}$$

If second-order derivatives are required for the optimization technique, you can compute the Hessian as

$$\frac{\partial^2 \ell_p}{\partial \boldsymbol{\beta}\partial \boldsymbol{\beta}'} = \frac{\partial^2 \ell}{\partial \boldsymbol{\beta}\partial \boldsymbol{\beta}'} - \mathbf{S}_{\boldsymbol{\lambda}}$$

In the gradient and Hessian forms, $\partial\ell/\partial\boldsymbol{\beta}$ and $\partial^2\ell/(\partial\boldsymbol{\beta}\partial\boldsymbol{\beta}')$ are the corresponding gradient and Hessian, respectively, for the log likelihood for generalized linear models. For more information about optimization techniques, see the section "Optimization Options" on page 57 in Chapter 2, "Shared Concepts."

Model Evaluation Criteria

Given a fixed set of smoothing parameters $\boldsymbol{\lambda}$ in which each λ_i controls the smoothness of each spline term, you can fit a generalized additive model by penalized likelihood estimation. There are infinitely many sets of smoothing parameters. In order to search optimum models, some model evaluation criteria need to be defined to quantify the model's goodness of fit. The GAMMOD procedure uses the following model evaluation criteria:

- generalized cross validation (GCV), $\mathcal{V}_g$ (Craven and Wahba 1979)
- unbiased risk estimator (UBRE), $\mathcal{V}_u$ (Craven and Wahba 1979)
- generalized approximate cross validation (GACV), $\mathcal{V}_a$ (Xiang and Wahba 1996)

Consider the optimization problem

$$\min(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})'(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})+\boldsymbol{\beta}'\mathbf{S}_{\boldsymbol{\lambda}}\boldsymbol{\beta} \quad \text{with respect to} \quad \boldsymbol{\beta}$$

The parameter estimate for $\boldsymbol{\beta}$ can be represented as

$$\hat{\boldsymbol{\beta}}=(\mathbf{X}'\mathbf{X}+\mathbf{S}_{\boldsymbol{\lambda}})^{-1}\mathbf{X}'\mathbf{y}$$

And the smoothing matrix (also called the influence matrix or hat matrix) is thus represented as

$$\mathbf{H}_{\boldsymbol{\lambda}}=\mathbf{X}(\mathbf{X}'\mathbf{X}+\mathbf{S}_{\boldsymbol{\lambda}})^{-1}\mathbf{X}'$$

With the defined smoothing matrix, you can form the model evaluation criteria as follows:

$$\begin{aligned}
\mathcal{V}_g(\boldsymbol{\lambda}) &= \frac{n\|\mathbf{y}-\mathbf{H}_{\boldsymbol{\lambda}}\mathbf{y}\|^2}{(\mathrm{tr}(\mathbf{I}-\gamma\mathbf{H}_{\boldsymbol{\lambda}}))^2} \\
\mathcal{V}_u(\boldsymbol{\lambda}) &= \frac{1}{n}\|\mathbf{y}-\mathbf{H}_{\boldsymbol{\lambda}}\mathbf{y}\|^2-\frac{2}{n}\sigma^2\mathrm{tr}(\mathbf{I}-\gamma\mathbf{H}_{\boldsymbol{\lambda}})+\sigma^2 \\
\mathcal{V}_a(\boldsymbol{\lambda}) &= \frac{1}{n}\|\mathbf{y}-\mathbf{H}_{\boldsymbol{\lambda}}\mathbf{y}\|^2\left(1+2\gamma\frac{\mathrm{tr}(\mathbf{H}_{\boldsymbol{\lambda}})}{\mathrm{tr}(\mathbf{I}-\mathbf{H}_{\boldsymbol{\lambda}})}\right)
\end{aligned}$$

In the equations, $\gamma \geq 1$ (which corresponds to the GAMMA= suboption of the CRITERION= option) is the tuning parameter that is sometimes used to enforce smoother models.

The GAMMOD procedure uses fitting algorithms that involve minimizing the model evaluation criterion with respect to unknown smoothing parameters $\boldsymbol{\lambda}$.

Fitting Algorithms

For models that assume a normally distributed response variable, you can minimize the model evaluation criteria directly by searching for optimal smoothing parameters. For models that have nonnormal distributions, you cannot use the model evaluation criteria directly because the involved statistics keep changing between iterations. The GAMMOD procedure enables you to use either of two fitting approaches to search for optimum models: the outer iteration method and the performance iteration method. The outer iteration method modifies the model evaluation criteria so that a global objective function can be minimized in order to find the best smoothing parameters. The performance iteration method minimizes a series of objective functions in an iterative fashion and then obtains the optimum smoothing parameters at convergence. For large data tables, the performance iteration method usually converges faster than the outer iteration method.

Outer Iteration

The outer iteration method is outlined in Wood (2006). The method uses modified model evaluation criteria, which are defined as follows:

$$
\begin{aligned}
\mathcal{V}_g^o(\boldsymbol{\lambda}) &= \frac{n D_{\boldsymbol{\lambda}}(\boldsymbol{\mu})}{(n-\gamma \mathrm{tr}(\mathbf{H}_{\boldsymbol{\lambda}}))^2} \\
\mathcal{V}_u^o(\boldsymbol{\lambda}) &= \frac{D_{\boldsymbol{\lambda}}(\boldsymbol{\mu})}{n} - \frac{2}{n}\sigma^2 \mathrm{tr}(\mathbf{I}-\gamma\mathbf{H}_{\boldsymbol{\lambda}}) + \sigma^2 \\
\mathcal{V}_a^o(\boldsymbol{\lambda}) &= \frac{D_{\boldsymbol{\lambda}}(\boldsymbol{\mu})}{n} + \frac{2\gamma \mathrm{tr}(\mathbf{H}_{\boldsymbol{\lambda}}) P_{\boldsymbol{\lambda}}}{n \mathrm{tr}(\mathbf{I}-\mathbf{H}_{\boldsymbol{\lambda}})}
\end{aligned}
$$

By replacing $\|\mathbf{y}-\mathbf{H}_{\boldsymbol{\lambda}}\mathbf{y}\|^2$ with model deviance $D_{\boldsymbol{\lambda}}(\boldsymbol{\mu})$, the modified model evaluation criteria relate to the smoothing parameter $\boldsymbol{\lambda}$ in a direct way such that the analytic gradient and Hessian are available in explicit forms. Pearson's statistic $P_{\boldsymbol{\lambda}}$ is used in the GACV criterion $\mathcal{V}_a^o(\boldsymbol{\lambda})$ (Wood 2008). The algorithm for the outer iteration is thus as follows:

1. Initialize smoothing parameters by taking one step of performance iteration based on adjusted response and adjusted weight except for spline terms whose initial values are specified in the INITSMOOTH= option.
2. Search for the best smoothing parameters by minimizing the modified model evaluation criteria. The optimization process stops when any of the convergence criteria that are specified in the SMOOTHOPTIONS option is met. At each optimization step:
   a) Initialize by setting initial regression parameters $\boldsymbol{\beta} = \{g(\hat{\mathbf{y}}), 0, \ldots, 0\}'$. Set the initial dispersion parameter if necessary.
   b) Search for the best regression parameters $\boldsymbol{\beta}$ by minimizing the penalized deviance D_p (or maximizing the penalized likelihood ℓ_p for negative binomial distribution). The optimization process stops when any of the convergence criteria that are specified in the PLIKEOPTIONS option is met.
   c) At convergence, evaluate derivatives of the model evaluation criteria with respect to $\boldsymbol{\lambda}$ by using $\partial D_p/\partial\boldsymbol{\beta}$, $\partial^2 D_p/(\partial\boldsymbol{\beta}\partial\boldsymbol{\beta}')$, $\partial\boldsymbol{\beta}/\partial\lambda_j$, and $\partial^2\boldsymbol{\beta}/(\partial\lambda_j\partial\lambda_k)$.

Step 2b usually converges quickly because it is essentially penalized likelihood estimation given that $D_p = 2\phi(\ell_{\max} - \ell) + \boldsymbol{\beta}\mathbf{S}_{\boldsymbol{\lambda}}\boldsymbol{\beta}'$. Step 2c contains involved computation by using the chain rule of derivatives. For more information about computing derivatives of $\mathcal{V}_g^o$ and $\mathcal{V}_u^o$, see Wood (2008, 2011).

Performance Iteration

The performance iteration method is proposed by Gu and Wahba (1991). Wood (2004) modifies and stabilizes the algorithm for fitting generalized additive models. The algorithm for the performance iteration method is as follows:

1. Initialize smoothing parameters $\boldsymbol{\lambda} = \{1, \ldots, 1\}$, except for spline terms whose initial values are specified in the INITSMOOTH= option. Set the initial regression parameters $\boldsymbol{\beta} = \{g(\bar{\mathrm{y}}), 0, \ldots, 0\}'$. Set the initial dispersion parameter if necessary.
2. Repeat the following steps until any of these three conditions is met: (1) the absolute function change in penalized likelihood is sufficiently small; (2) the absolute relative function change in penalized likelihood is sufficiently small; (3) the number of iterations exceeds the maximum iteration limit.
   a) Form the adjusted response and adjusted weight from $\boldsymbol{\mu} = g^{-1}(\boldsymbol{\eta})$. For each observation,
   $$z_i = \eta_i + (y_i - \mu_i)/\mu_i', \quad w_i = \omega_i \mu_i'^2/v_i$$
   b) Search for the best smoothing parameters for the current iteration based on valid adjusted response values and adjusted weight values.
   c) Use the smoothing parameters to construct the linear predictor and the predicted means.
   d) Obtain an estimate for the dispersion parameter if necessary.

In step 2b, you can use different optimization techniques to search for the best smoothing parameters. The Newton-Raphson optimization is efficient in finding the optimum $\boldsymbol{\lambda}$ where the first- and second-order derivatives are available. For more information about computing derivatives of $\mathcal{V}_g$ and $\mathcal{V}_u$ with respect to $\boldsymbol{\lambda}$, see Wood (2004).

Degrees of Freedom

Let $\mathbf{W}$ be the adjusted weight matrix at convergence, and let $\mathbf{S}_{\boldsymbol{\lambda}}$ be the roughness penalty matrix with selected smoothing parameters. The degrees of freedom matrix is defined as in Wood (2006):

$$\mathbf{F} = (\mathbf{X}'\mathbf{W}\mathbf{X} + \mathbf{S}_{\boldsymbol{\lambda}})^{-1}\mathbf{X}'\mathbf{W}\mathbf{X}$$

Given the adjusted response $\mathbf{z}$, the parameter estimate is shown to be $\tilde{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}\mathbf{X}'\mathbf{W}\mathbf{z}$ for the model without penalization, and the parameter estimate is $\hat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{W}\mathbf{X} + \mathbf{S}_{\boldsymbol{\lambda}})^{-1}\mathbf{X}'\mathbf{W}\mathbf{z} = \mathbf{F}\tilde{\boldsymbol{\beta}}$ with penalization. $\mathbf{F}$ is thus the matrix that projects or maps the unpenalized parameter estimates to the penalized ones.

The model degrees of freedom is given as

$$\mathrm{df} = \mathrm{tr}(\mathbf{F})$$

And the degrees of freedom for error is given as

$$\mathrm{df}_r = n - 2\mathrm{df} + \mathrm{tr}(\mathbf{FF})$$

For the *j*th spline term, the degrees of freedom for the component is defined to be the trace of the submatrix of $\mathbf{F}$ that corresponds to parameter estimates $\boldsymbol{\beta}_j$:

$$\mathrm{df}_j = \mathrm{tr}(\mathbf{F}_j)$$

The degrees of freedom for the smoothing component test of the *j*th term is defined similarly as

$$\mathrm{df}_j^t = 2\mathrm{df}_j - \mathrm{tr}((\mathbf{FF})_j)$$

Model Inference

Wahba (1983) proposes a Bayesian covariance matrix for parameter estimates $\boldsymbol{\beta}$ by interpreting a smoothing spline as a posterior mean. Nychka (1988) shows that the derived Bayesian posterior confidence limits work well from frequentist viewpoints. The Bayesian posterior covariance matrix for the parameters is

$$\mathbf{V}_{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{W}\mathbf{X} + \mathbf{S}_{\boldsymbol{\lambda}})^{-1}\sigma^2$$

The posterior distribution for $\boldsymbol{\beta}$ is thus

$$\boldsymbol{\beta}|\mathbf{y} \sim N(\hat{\boldsymbol{\beta}}, \mathbf{V}_{\boldsymbol{\beta}})$$

For a particular point whose design row is vector $\mathbf{x}$, the prediction is $\mathbf{x}\hat{\boldsymbol{\beta}}$ and the standard error is $\sqrt{\mathbf{x}\mathbf{V}_{\boldsymbol{\beta}}\mathbf{x}'}$. The Bayesian posterior confidence limits are thus

$$\left(\mathbf{x}\hat{\boldsymbol{\beta}} \pm z_{\alpha/2}\sqrt{\mathbf{x}\mathbf{V}_{\boldsymbol{\beta}}\mathbf{x}'}\right)$$

where $z_{\alpha/2}$ is the $1 - \alpha/2$ quantile of the standard normal distribution.

For the *j*th spline term, the prediction for the component is formed by $\mathbf{x}_j\hat{\boldsymbol{\beta}}$, where $\mathbf{x}_j$ is a row vector of zeros except for columns that correspond to basis expansions of the *j*th spline term. And the standard error for the component is $\sqrt{\mathbf{x}_j\mathbf{V}_{\boldsymbol{\beta}}\mathbf{x}_j'}$.

Dispersion Parameter

Some distribution families (Gaussian, gamma, inverse Gaussian, negative binomial, and Tweedie) have a dispersion parameter that you can specify in the DISPERSION= option in the MODEL statement or that you can estimate from the data. The following three suboptions for the SCALE= option in the MODEL statement correspond to three ways to estimate the dispersion parameter:

DEVIANCE
estimates the dispersion parameter by the deviance, given the regression parameter estimates:

$$\hat{\phi} = \frac{\sum_i D_i(y_i, \mu_i)}{n - \mathrm{df}}$$

MLE
estimates the dispersion parameter by maximizing the penalized likelihood, given the regression parameter estimates:

$$\hat{\phi} = \underset{\phi}{\operatorname{argmax}}\, \ell_p(\hat{\boldsymbol{\beta}}, \phi)$$

The MLE option is the only option that you can use to estimate the dispersion parameter for the negative binomial and Tweedie distributions.

PEARSON
estimates the dispersion parameter by Pearson's statistic, given the regression parameter estimates:

$$\hat{\phi} = \frac{\sum_i \omega_i (y_i - \mu_i)^2 / v_i}{n - \mathrm{df}}$$

If the dispersion parameter is estimated, it contributes one additional degree of freedom to the fitted model.

Tests for Smoothing Components

The GAMMOD procedure performs a smoothing component test on the null hypothesis $f_j = 0$ for the jth component. In contrast to the analysis of deviance that is used in the GAM procedure (which tests existence of nonlinearity for each smoothing component), the smoothing component test used in PROC GAMMOD tests for the existence of a contribution for each smoothing component.

The hypothesis test is based on the Wald statistic. Define $\mathbf{X}_j$ as the matrix of all zeros except for columns that correspond to basis expansions of the jth spline term. Then the column vector of predictions is $\hat{\mathbf{f}}_j = \mathbf{X}_j \hat{\boldsymbol{\beta}}$, and the covariance matrix for the predictions is $\mathbf{V}_j = \mathbf{X}_j \mathbf{V}_{\boldsymbol{\beta}} \mathbf{X}_j'$. The Wald statistic for testing is

$$T_r = \hat{\mathbf{f}}_j' \mathbf{V}_j^{r-} \hat{\mathbf{f}}_j = \hat{\boldsymbol{\beta}}' \mathbf{X}_j' (\mathbf{X}_j \mathbf{V}_{\boldsymbol{\beta}} \mathbf{X}_j')^{r-} \mathbf{X}_j \hat{\boldsymbol{\beta}}$$

where $\mathbf{V}_j^{r-}$ is the rank-r pseudo-inverse of $\mathbf{V}_j$. If $\mathbf{R}_j$ is the Cholesky root for $\mathbf{X}_j' \mathbf{X}_j$ such that $\mathbf{R}_j' \mathbf{R} = \mathbf{X}_j' \mathbf{X}_j$, then the test statistic can be written as

$$T_r = \hat{\boldsymbol{\beta}}' \mathbf{R}_j' (\mathbf{R}_j \mathbf{V}_{\boldsymbol{\beta}} \mathbf{R}_j')^{r-} \mathbf{R}_j \hat{\boldsymbol{\beta}}$$

Wood (2012) proposes using the df_j^t degrees of freedom for test (which is defined in the section "Degrees of Freedom" on page 614) as the rank r. Because spline terms in fitted models often have noninteger degrees of freedom, the GAMMOD procedure uses a rounded value of df_j^t as the rank:

$$r = \begin{cases} \lfloor \mathrm{df}_j^t \rfloor & \text{if } \mathrm{df}_j^t - \lfloor \mathrm{df}_j^t \rfloor <= 0.05 \text{ or } \mathrm{df}_j^t < 1 \\ \lceil \mathrm{df}_j^t \rceil & \text{otherwise} \end{cases}$$

Let K be a symmetric and nonnegative definite matrix, and let its eigenvalues be sorted as $d_1 > d_2 > \cdots$; then the rank-r pseudo-inverse of K is formed by

$$K^{r-} = U_k \begin{bmatrix} d_1^{-1} & & \\ & \ddots & \\ & & d_r^{-1} \end{bmatrix} U_k'$$

where U_k are formed by columns of eigenvectors that correspond to the r eigenvalues.

Under the null hypothesis, the Wald statistic T_r approximately follows the chi-square distribution $T_r \sim \chi_r^2$. For an observed test statistic t_r, the p-value for rejecting the null hypothesis is computed as $P(\chi_r^2 > t_r)$ if the dispersion parameter is constant, or $P(F_{r,\mathrm{df}_r} > t_r)$ with df_r error degrees of freedom if the dispersion parameter is estimated.

Be cautious when you interpret the results of the smoothing component test because p-values are computed by approximation and the test does not take the smoothing parameter selection process into account.

Multithreading

The GAMMOD procedure allocates data to different threads and calculates the likelihood function, gradient, and Hessian by accumulating the values from all threads. For more information about how PROC GAMMOD uses threads, see the section "Multithreading" on page 102 in Chapter 2, "Shared Concepts."

Optimization Algorithms

Several optimization techniques are available in PROC GAMMOD. You can choose a particular optimizer by using the TECHNIQUE= option in the PROC GAMMOD statement. Table 12.13 summarizes the optimization techniques available in PROC GAMMOD.

Table 12.13 Optimization Techniques

TECHNIQUE=	Algorithm
TRUREG	Trust region method
NEWRAP	Newton-Raphson method with line search
NRRIDG	Newton-Raphson method with ridging
QUANEW	Quasi-Newton methods
DBLDOG	Double-dogleg method
CONGRA	Conjugate gradient methods
NMSIMP	Nelder-Mead simplex method

There is no algorithm for optimizing general nonlinear functions that always finds the global optimum for a general nonlinear optimization problem in a reasonable amount of time. Because no single optimization technique is always superior to others, PROC GAMMOD provides a variety of optimization techniques that work well in various circumstances. However, you can devise problems for which none of the techniques in PROC GAMMOD can find the correct solution. Moreover, nonlinear optimization can be computationally expensive in terms of time and memory, so you must be careful when matching an algorithm to a problem. The section "Choosing an Optimization Algorithm" on page 103 in Chapter 2, "Shared Concepts," is helpful in choosing a suitable optimization algorithm.

Displayed Output

The following sections describe the output that the GAMMOD procedure produces by default. The output is organized into various tables, which are discussed in the order of their appearance.

Model Information

The "Model Information" table displays basic information about the model, such as the response variable, frequency variable, link function, and model category that the GAMMOD procedure determines based on your input and options. The "Model Information" table also displays the distribution of the data that PROC GAMMOD assumes. For information about the supported response distributions, see the DISTRIBUTION= option.

Number of Observations

The "Number of Observations" table displays the number of observations that are read from the input data table and the number of observations that are used in the analysis. If a FREQ statement is present, the sum of the frequencies read and used is displayed. If the events/trials syntax is used, the number of events and trials is also displayed.

Response Profile

The "Response Profile" table displays the ordered values from which the GAMMOD procedure determines the probability that is modeled as an event in binary models. For each response category level, the frequency that is used in the analysis is reported. You can affect the ordering of the response values by specifying *response-options* in the MODEL statement. For binary models, the note that follows the "Response Profile" table indicates which outcome is modeled as the event in binary models and which value serves as the reference category.

The "Response Profile" table is not produced for binomial (events/trials) data. You can find information about the number of events and trials in the "Number of Observations" table.

Class Level Information

The "Class Level Information" table lists the levels of every variable that is specified in the CLASS statement. You should check this information to make sure that the data are correct. You can adjust the order of the CLASS variable levels by specifying the ORDER= option in the CLASS statement. You can suppress the "Class Level Information" table completely or partially by specifying the NOCLPRINT= option in the PROC GAMMOD statement.

If the classification variables use reference parameterization, the "Class Level Information" table also displays the reference value for each variable.

Specifications for Spline(*spline-variables*)

The "Specifications for Spline(*spline-variables*)" table displays basic information (such as the number of variables, specified degrees of freedom, search range for the smoothing parameter, and so on) about how to construct a spline term that the GAMMOD procedure uses to construct basis expansions and search for the smoothing parameter. PROC GAMMOD generates the "Specifications for Spline(*spline-variables*)" table only when you specify the DETAILS option for a spline term.

Optimization Iteration History

For each iteration of the optimization, the "Iteration History" table displays the number of function evaluations (including gradient and Hessian evaluations), the value of the objective function, the change in the objective function from the previous iteration, and the absolute value of the largest (projected) gradient element. The objective function that PROC GAMMOD uses in the optimization is normalized by default to enable comparisons across data tables that have different sampling intensity.

If you specify the ITDETAILS option in the PROC GAMMOD statement, information about the parameter estimates and gradients in the course of the optimization is added to the "Iteration History" table.

For a parametric generalized linear model or for a generalized additive model that has fixed smoothing parameters, the "Iteration History" table displays information about regression parameter estimates and gradients. For a generalized additive model that has unknown smoothing parameters, the "Iteration History" table displays information about smoothing parameter estimates and gradients. If the performance iteration method is used, a column of performance iteration steps is added to the table.

Convergence Status

The convergence status table is a small ODS table that follows the "Iteration History" table in the default output. In the listing, this table appears as a message that indicates whether the optimization succeeded and which convergence criterion was met. If the optimization fails, the message indicates the reason for the failure. If you save the convergence status table to an output data table, a numeric Status variable is added that enables you to programmatically assess convergence. The values of the Status variable encode the following:

0 Convergence was achieved, or an optimization was not performed because TECHNIQUE=NONE was specified.

1 The objective function could not be improved.

2 Convergence was not achieved because of a user interrupt or because a limit (such as the maximum number of iterations or the maximum number of function evaluations) was reached. To modify these limits, see the MAXITER=, MAXFUNC=, and MAXTIME= options in the PROC GAMMOD statement.

3 Optimization failed to converge because function or derivative evaluations failed at the starting values or during the iterations or because a feasible point that satisfies the parameter constraints could not be found in the parameter space.

Fit Statistics

The "Fit Statistics" table displays a variety of likelihood-based measures of fit in addition to the model roughness measurement. All information criteria are presented in "smaller is better" form.

The calculation of the information criteria uses the following formulas, where df denotes the model degrees of freedom, f denotes the number of frequencies used, and ℓ is the log likelihood that is evaluated at the converged estimates:

$$\mathrm{AIC} = -2\ell + 2\mathrm{df}$$

$$\mathrm{AICC} = \begin{cases} -2\ell + 2\mathrm{df}\, f/(f - \mathrm{df} - 1) & \text{when } f > \mathrm{df} + 2 \\ -2\ell + 2\mathrm{df}(\mathrm{df} + 2) & \text{otherwise} \end{cases}$$

$$\mathrm{BIC} = -2\ell + \mathrm{df}\log(f)$$

If no FREQ statement is specified, then f equals n, the number of observations that are used.

Parameter Estimates

The "Parameter Estimates" table displays the regression parameter estimates, their estimated (asymptotic) standard errors, chi-square statistics, and p-values for the hypothesis that the parameter is 0.

Estimates for Smoothing Components

The "Estimates for Smoothing Components" table displays a summary of the fitted spline terms, including effective degrees of freedom, smoothing parameters, roughness penalty values, number of parameters, rank of penalty matrix, and number of knots.

Tests for Smoothing Components

The "Tests for Smoothing Components" table displays effective degrees of freedom, effective degrees of freedom for test, F/χ_2 statistics, and p-values for rejecting the hypothesis that the smoothing component has zero contribution.

Timing

The "Timing" table displays the amount of time (in seconds) that PROC GAMMOD required to perform different tasks in the analysis.

OutputCasTables Table

The OutputCasTables table is a special table that has information about each CAS table that is created during a CAS action execution. The information for each CAS table consists of the CAS table name, the caslib in which the table resides, and the number of columns and rows in the CAS table. Because this table is not a typical ODS table that contains analytical results, you cannot include it in the *table-spec-list* in the DISPLAYOUT statement.

ODS Table Names

Each table that the GAMMOD procedure creates has a name that is associated with it, and you must use this name to refer to the table when you use ODS statements. These names are listed in Table 12.14.

Table 12.14 ODS Tables Produced by PROC GAMMOD

Table Name	Description	Statement	Option
ClassInfo	Level information from the CLASS statement	CLASS	
ConvergenceStatus	Status of optimization at conclusion of optimization	Default output	
FitStatistics	Fit statistics	Default output	
LikelihoodHist	Iteration history for maximum likelihood estimation or penalized likelihood estimation	PROC GAMMOD	ITDETAILS
ModelInfo	Information about the modeling environment	Default output	
NObs	Number of observations read and used, and number of events and trials, if applicable	Default output	
OutputCasTables	See the section "OutputCasTables Table" on page 620	OUTPUT DISPLAYOUT	OUT=
ParameterEstimates	Solutions for the parameter estimates that are associated with parametric effects in the MODEL statement	Default output	
ResponseProfile	Response categories and the category that is modeled in models for binary data	Default output	
SmoothingEstimates	Information for spline terms after model fitting	Default output	
SmoothingHist	Iteration history for smoothing parameter estimation	PROC GAMMOD	ITDETAILS
SmoothingTests	Smoothing components test result	Default output	
SplineDetails	Information about spline construction and smoothing parameter search	MODEL	SPLINE(*variables* / DETAILS)
Timing	Absolute and relative times for tasks performed by the procedure	Default output	

ODS Graphics

Statistical procedures use ODS Graphics to create graphs as part of their output. ODS Graphics is described in detail in the "Statistical Graphics Using ODS" chapter in *SAS/STAT User's Guide*.

Before you create graphs, ODS Graphics must be enabled (for example, by using the ODS GRAPHICS ON statement). For more information about enabling and disabling ODS Graphics, see the section "Enabling and Disabling ODS Graphics" in that chapter.

The overall appearance of graphs is controlled by ODS styles. Styles and other aspects of using ODS Graphics are discussed in the section "A Primer on ODS Statistical Graphics" in that chapter.

When ODS Graphics is enabled, the GAMMOD procedure by default produces plots of the partial predictions for each spline term in the model. Use the PLOTS option in the PROC GAMMOD statement to control aspects of these plots.

PROC GAMMOD assigns a name to each graph that it creates by using ODS. You can use these names to refer to the graphs when using ODS. The names are listed in Table 12.15.

Table 12.15 Graphs Produced by PROC GAMMOD

ODS Graph Name	Plot Description	PLOTS= Option
SmoothingComponentPanel	Panel of multiple prediction curves	Default COMPONENTS
SmoothingComponentPlot	Unpacked prediction curves	PLOTS(UNPACK) PLOTS=COMPONENTS(UNPACK)

By default, prediction plots for each spline component are displayed in panels that contain at most six plots. If you specify more than six smoothing components, multiple panels are used. Use the PLOTS(UNPACK) option in the PROC GAMMOD statement to display these plots individually.

Examples: GAMMOD Procedure

The examples in this chapter are available in the GitHub repository located at https://github.com/sascommunities/doc-supplement-casstat.

Example 12.1: Scatter Plot Smoothing

(View the complete code for this example (gammodex1.sas) in the example repository.)

This example shows how you can use PROC GAMMOD to perform scatter plot smoothing.

The example uses the LIDAR data set (Ruppert, Wand, and Carroll 2003). This data set is used in many books and journals to illustrate different smoothing techniques. Scientists use a technique known as LIDAR (light detection and ranging), which uses laser reflections to detect chemical compounds in the atmosphere. The following DATA step creates the data table Lidar. This DATA step assumes that your CAS engine libref is named mylib, but you can substitute any appropriately defined CAS engine libref.

```
title 'Scatter Plot Smoothing';
data mylib.Lidar;
   input Range LogRatio @@;
   datalines;
390  -0.05035573    391  -0.06009706    393  -0.04190091    394   -0.0509847
396  -0.05991345    397  -0.02842392    399  -0.05958421    400  -0.03988881
402  -0.02939582    403  -0.03949445    405  -0.04764749    406     -0.06038
408  -0.03123034    409  -0.03816584    411  -0.07562269    412  -0.05001751
414   -0.0457295    415  -0.07766966    417  -0.02460641    418  -0.07133184

   ... more lines ...

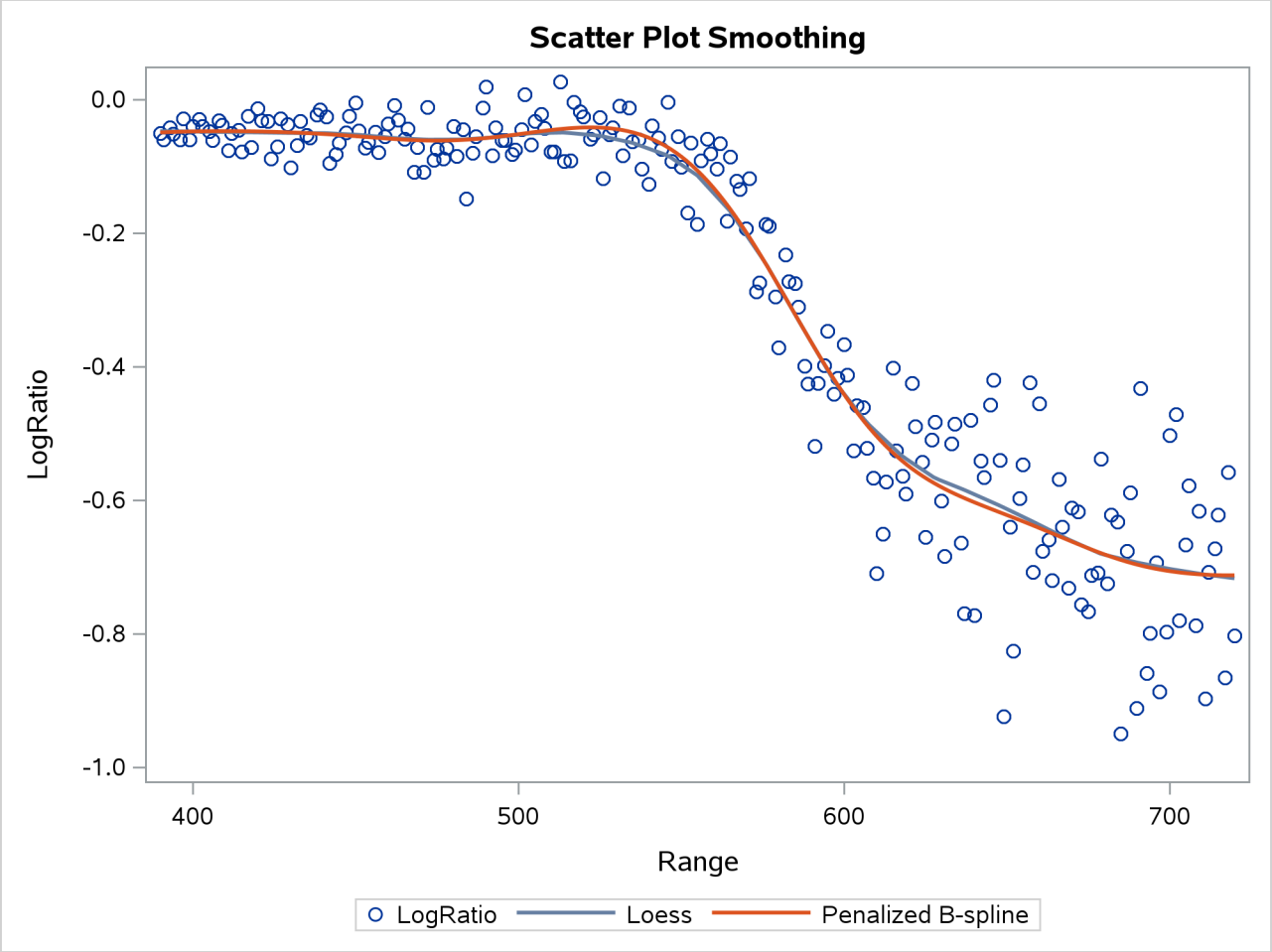
702   -0.4716702    703   -0.7801088    705   -0.6668431    706   -0.5783479
708   -0.7874522    709   -0.6156956    711   -0.8967602    712   -0.7077379
714    -0.672567    715   -0.6218413    717   -0.8657611    718    -0.557754
720   -0.8026684
;
```

In this data table, Range records the distance that light travels before it is reflected back to the source. LogRatio is the logarithm of the ratio of light that is received from two laser sources. The objective is to use scatter plot smoothing to discover the nonlinear pattern in the data. SAS provides different methods (for example, local regression) for scatter plot smoothing. You can perform scatter plot smoothing by using the SGPLOT procedure, as shown in the following statements:

```
proc sgplot data=mylib.Lidar;
   scatter  x=Range y=LogRatio;
   loess    x=Range y=LogRatio / nomarkers;
   pbspline x=Range y=LogRatio / nomarkers;
run;
```

Output 12.1.1 shows the scatter plot of Range and LogRatio and the smoothing curves that are fitted by the local regression and penalized B-splines smoothing techniques.

Output 12.1.1 Scatter Plot Smoothing



Both scatter plot smoothing techniques show a significant nonlinear structure between Range and LogRatio that cannot be easily modeled by ordinary polynomials. You can also use the GAMMOD procedure to perform scatter plot smoothing on these data, as in the following statements:

```
proc gammod data=mylib.Lidar seed=12345;
   model LogRatio = spline(Range/details);
   output out=mylib.LidarOut pred=p;
run;
```

The "Specifications for Spline(Range)" table in Output 12.1.2 displays the specifications for constructing the spline term for Range. The maximum degrees of freedom is 10, which sets the upper limit of effective degrees of freedom for the spline term to be 9 after one degree of freedom is absorbed in the intercept. The order of the derivative in the penalty is 2, which means that the unpenalized portion of the spline term involves polynomials with degrees up to 2.

Output 12.1.2 Spline Specification

Scatter Plot Smoothing

The GAMMOD Procedure

Specifications for Spline(Range)	
Number of Variables	1
Rank of Penalty Approximation	10
Order of Derivative in the Penalty	2
Maximum Number of Knots	2000

The "Fit Statistics" table in Output 12.1.3 shows the summary statistics for the fitted model.

Output 12.1.3 Fit Statistics

Fit Statistics	
Penalized Log Likelihood	251.44124
Roughness Penalty	0.00000385
Effective Degrees of Freedom	9.99989
Effective Degrees of Freedom for Error	211.00000
AIC (smaller is better)	-482.88271
AICC (smaller is better)	-481.83512
BIC (smaller is better)	-448.90148
GCV (smaller is better)	0.00654

The "Estimates for Smoothing Components" table in Output 12.1.4 shows that the effective degrees of freedom for the spline term of Range is approximately 8 after the GCV criterion is optimized with respect to the smoothing parameter. The roughness penalty is small, suggesting that there is an important contribution from the penalized part of thin-plate regression splines beyond nonpenalized polynomials.

Output 12.1.4 Estimates for Smoothing Components

Estimates for Smoothing Components						
Component	Effective DF	Smoothing Parameter	Roughness Penalty	Number of Parameters	Rank of Penalty Matrix	Number of Knots
Spline(Range)	7.99989	1.0000	3.85E-6	9	10	221

Because the optimal model is obtained by searching in a functional space that is constrained by the maximum degrees of freedom for a spline term, you might wonder whether PROC GAMMOD produces a much different model if you increase the value. The following statements fit another model in which the maximum degrees of freedom is increased to 20:

```
proc gammod data=mylib.Lidar seed=12345;
   model LogRatio = spline(Range/maxdf=20);
   output out=mylib.LidarOut2 pred=p2;
run;
```

Output 12.1.5 displays fit summary statistics for the second model. The model fit statistics from the second model are very close to the ones from the first model, indicating that the second model is not much different from the first model.

Output 12.1.5 Fit Statistics

Scatter Plot Smoothing

The GAMMOD Procedure

Fit Statistics	
Penalized Log Likelihood	250.96488
Roughness Penalty	0.05315
Effective Degrees of Freedom	10.06143
Effective Degrees of Freedom for Error	209.02182
AIC (smaller is better)	-481.86006
AICC (smaller is better)	-480.79981
BIC (smaller is better)	-447.66969
GCV (smaller is better)	0.00657

Output 12.1.6 shows that the effective degrees of freedom for the spline term of Range is slightly larger than 8, which is understandable because increasing the maximum degrees of freedom expands the functional space for model searching. Functions in the expanded space can provide a better fit to the data, but they are also penalized more because the roughness penalty value for the second model is much larger than the one for the first model. This suggests that functions in the expanded space do not help much, given the nonlinear relationship between Range and LogRatio.

Output 12.1.6 Estimates for Smoothing Components

Estimates for Smoothing Components						
Component	Effective DF	Smoothing Parameter	Roughness Penalty	Number of Parameters	Rank of Penalty Matrix	Number of Knots
Spline(Range)	8.06143	23032.9	0.0532	19	20	221

The two fitted models are both based on thin-plate regression splines, in which polynomials that have degrees higher than 2 are penalized. You might wonder whether allowing higher-order polynomials yields a much different model. The following statements fit a third spline model by penalizing polynomials that have degrees higher than 3:

```
proc gammod data=mylib.Lidar seed=12345;
   model LogRatio = spline(Range/m=3);
   output out=mylib.LidarOut3 pred=p3;
run;
```

The fit summary statistics shown in Output 12.1.7 are close to the ones from the previous two models, albeit slightly smaller.

Output 12.1.7 Fit Statistics

Scatter Plot Smoothing

The GAMMOD Procedure

Fit Statistics	
Penalized Log Likelihood	249.79779
Roughness Penalty	9.440383E-9
Effective Degrees of Freedom	10.00000
Effective Degrees of Freedom for Error	211.00000
AIC (smaller is better)	-479.59559
AICC (smaller is better)	-478.54797
BIC (smaller is better)	-445.61397
GCV (smaller is better)	0.00664

As shown in Output 12.1.8, the effective degrees of freedom for the spline term where polynomials with degrees less than 4 are allowed without penalization is 8. The roughness penalty is quite small compared to the previous two fits. This also suggests that there are important contributions from the penalized part of the thin-plate regression splines even after the nonpenalized polynomials are raised to order 3.

Output 12.1.8 Estimates for Smoothing Components

Estimates for Smoothing Components

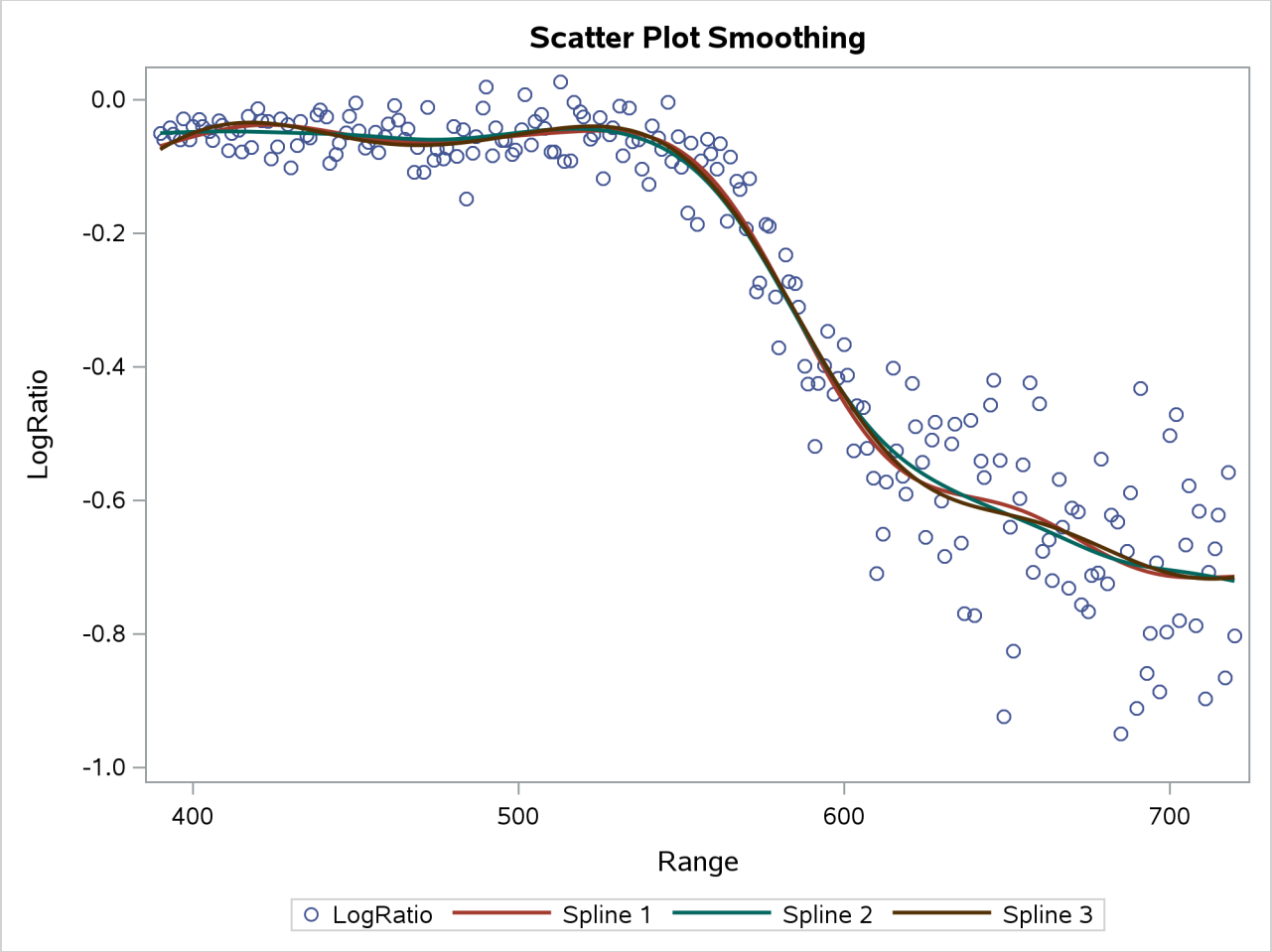
Component	Effective DF	Smoothing Parameter	Roughness Penalty	Number of Parameters	Rank of Penalty Matrix	Number of Knots
Spline(Range)	8.00000	1.0000	9.44E-9	9	10	221

The following statements use the DATA step to merge the predictions from the three scatter plot smoothing fits by PROC GAMMOD and use the SGPLOT procedure to visualize them:

```
data LidarPred;
   merge mylib.Lidar mylib.LidarOut mylib.LidarOut2 mylib.LidarOut3;
run;
proc sort data=LidarPred; by Range;run;
proc sgplot data=LidarPred;
   scatter x=Range y=LogRatio / markerattrs=GraphData1(size=7);
   series  x=Range y=p        / lineattrs  =GraphData2(thickness=2)
                               legendlabel="Spline 1";
   series  x=Range y=p2       / lineattrs  =GraphData3(thickness=2)
                               legendlabel="Spline 2";
   series  x=Range y=p3       / lineattrs  =GraphData4(thickness=2)
                               legendlabel="Spline 3";
run;
```

Output 12.1.9 displays the scatter plot smoothing fits by PROC GAMMOD under three different spline specifications.

Output 12.1.9 Scatter Plot Smoothing



Example 12.2: Nonparametric Logistic Regression

(View the complete code for this example (gammodex2.sas) in the example repository.)

This example shows how you can use PROC GAMMOD to build a nonparametric logistic regression model for a data set that contains a binary response and then use that model to classify observations.

The example uses the Pima Indians Diabetes data set, which can be obtained from the UCI Machine Learning Repository (Lichman 2013). It is extracted from a larger database that was originally owned by the National Institute of Diabetes and Digestive and Kidney Diseases. Data are for female patients who are at least 21 years old, are of Akimel O'otham (Pima Indian) heritage, and live near Phoenix, Arizona. The objective of this study is to investigate the relationship between a diabetes diagnosis and variables that represent physiological measurements and medical attributes. Some missing or invalid observations are removed from the analysis. The reduced data set contains 532 records. The following DATA step creates the data set DiabetesStudy:

```
title 'Diabetes Study';
data DiabetesStudy;
   input NPreg Glucose Pressure Triceps BMI Pedigree Age Diabetes Test@@;
   datalines;
 6  148   72   35   33.6  0.627  50  1  1    1   85   66   29   26.6  0.351  31  0  1
 1   89   66   23   28.1  0.167  21  0  0    3   78   50   32    31  0.248  26  1  0
 2  197   70   45   30.5  0.158  53  1  0    5  166   72   19   25.8  0.587  51  1  1
 0  118   84   47   45.8  0.551  31  1  1    1  103   30   38   43.3  0.183  33  0  1
 3  126   88   41   39.3  0.704  27  0  0    9  119   80   35    29  0.263  29  1  0

   ... more lines ...

 1  128   48   45   40.5  0.613  24  1  0    2  112   68   22   34.1  0.315  26  0  0
 1  140   74   26   24.1  0.828  23  0  1    2  141   58   34   25.4  0.699  24  0  1
 7  129   68   49   38.5  0.439  43  1  1    0  106   70   37   39.4  0.605  22  0  0
 1  118   58   36   33.3  0.261  23  0  1    8  155   62   26    34  0.543  46  1  0
;
```

The data set contains nine variables, including the binary response variable Diabetes. Table 12.16 describes the variables.

Table 12.16 Variables in the DiabetesStudy Data Set

Variable	Description
NPreg	Number of pregnancies
Glucose	Two-hour plasma glucose concentration in an oral glucose tolerance test
Pressure	Diastolic blood pressure (mm Hg)
Triceps	Triceps skin fold thickness (mm)
BMI	Body mass index (weight in kg/(height in m)^2)
Pedigree	Diabetes pedigree function
Age	Age (years)
Diabetes	0 if test negative for diabetes, 1 if test positive
Test	0 for training role, 1 for test

The Test variable splits the data set into training and test subsets. The training observations (whose Test value is 0) hold approximately 59.4% of the data. To build a model that is based on the training data and evaluate its performance by predicting the test data, you use the following statements to create a new variable, Result, whose value is the same as that of the Diabetes variable for a training observation and is missing for a test observation:

```
data mylib.DiabetesStudy;
   set DiabetesStudy;
   Result = Diabetes;
   if Test=1 then Result=.;
run;
```

A typical starting point of your analysis for modeling the binary response might be a parametric logistic regression model. But it is restricted in the sense that all variables affect the response in strictly linear fashion. If you are uncertain that a variable is an important factor and its contribution is linear in predicting the response, you might want to choose a nonparametric logistic regression model to fit the data. You can use

PROC GAMMOD to form a nonparametric model by including the spline transformation of each explanatory variable, as shown in the following statements:

```
proc gammod data=mylib.DiabetesStudy seed=12345;
   model Result(event='1') = spline(NPreg)    spline(Glucose)
                             spline(Pressure) spline(Triceps)
                             spline(BMI)      spline(Pedigree)
                             spline(Age) / dist=binary;
run;
```

Because the response variable Result is binary, the DIST=BINARY option in the MODEL statement specifies a binary distribution for the response variable. By default, PROC GAMMOD models the probability of the first ordered response category, which is a negative diabetes testing result in this case. The EVENT= option specifically requests that PROC GAMMOD model the probability of positive diabetes testing results. The "Response Profile" table in Output 12.2.1 shows the frequencies of the response in both categories.

Output 12.2.1 Response Profile

Diabetes Study

The GAMMOD Procedure

Response Profile

Ordered Value	Result	Total Frequency
1	0	208
2	1	122

Probability modeled is Result = 1.

Output 12.2.2 lists the summary statistics from the nonparametric logistic regression model, which include spline transformations of all variables.

Output 12.2.2 Fit Statistics

Fit Statistics	
Penalized Log Likelihood	-138.57851
Roughness Penalty	1.208734E-8
Effective Degrees of Freedom	16.00000
Effective Degrees of Freedom for Error	314.28286
AIC (smaller is better)	309.15702
AICC (smaller is better)	310.89504
BIC (smaller is better)	369.94250
UBRE (smaller is better)	0.02357

The "Tests for Smoothing Components" table in Output 12.2.3 shows approximate tests results. Although some spline terms are significant, others are not. The null testing hypothesis is whether the total contribution from a variable is 0. So you can form a reduced model by removing those nonsignificant spline terms from the model. In this case, spline transformations for NPreg, Pressure, BMI, and Triceps are dropped from the model because their p-values are larger than the 0.1 nominal level.

Output 12.2.3 Tests for Smoothing Components

Tests for Smoothing Components				
Component	Effective DF	Effective DF for Test	Chi-Square	Pr > ChiSq
Spline(NPreg)	1.00000	1	0.0186	0.8916
Spline(Glucose)	1.00000	1	44.5949	<.0001
Spline(Pressure)	1.00000	1	2.8310	0.0925
Spline(Triceps)	1.00000	1	1.2128	0.2708
Spline(BMI)	1.00000	1	5.8112	0.0159
Spline(Pedigree)	1	1	6.1108	0.0134
Spline(Age)	9	9	19.0721	0.0246

The following statements use PROC GAMMOD to fit a reduced nonparametric logistic regression model. The OUTPUT statement requests predicted probabilities for both training and test data. The COPYVARS option in the OUTPUT statement requests that the Diabetes and Test variables also be included in the output data table so that you can use them to identify test observations and compute misclassification errors.

```
ods graphics on;
proc gammod data=mylib.DiabetesStudy plots seed=12345;
   model Result(event='1') = spline(Glucose)
                             spline(Pedigree) spline(Age) / dist=binary;
   output out=mylib.DiabetesStudyOut copyvars=(Diabetes Test);
run;
```

Output 12.2.4 shows the summary statistics from the reduced nonparametric logistic regression model. The values of the information criteria are better than those of the parametric logistic regression model.

Output 12.2.4 Fit Statistics

Diabetes Study

The GAMMOD Procedure

Fit Statistics	
Penalized Log Likelihood	-149.85766
Roughness Penalty	2.85613
Effective Degrees of Freedom	8.05241
Effective Degrees of Freedom for Error	320.61183
AIC (smaller is better)	312.96402
AICC (smaller is better)	313.41826
BIC (smaller is better)	343.55588
UBRE (smaller is better)	-0.00230

In the "Estimates for Smoothing Components" table in Output 12.2.5, PROC GAMMOD reports that the effective degrees of freedom value for spline transformations of Glucose is quite close to 1. This suggests a strictly linear form for Glucose. For Pedigree, the degrees of freedom value demonstrates a moderate amount of nonlinearity. For Age, the degrees of freedom value is much larger than 1. The measure suggests a nonlinear pattern in the dependency of the response on Age.

Output 12.2.5 Estimates for Smoothing Components

Estimates for Smoothing Components

Component	Effective DF	Smoothing Parameter	Roughness Penalty	Number of Parameters	Rank of Penalty Matrix	Number of Knots
Spline(Glucose)	1.00000	9.636E17	2.66E-14	9	10	110
Spline(Pedigree)	1.51071	0.4383	0.5086	9	10	283
Spline(Age)	4.54170	69.8812	2.3475	9	10	42

The "Tests for Smoothing Components" table in Output 12.2.6 shows that all spline transformations are significant in predicting diabetes testing results.

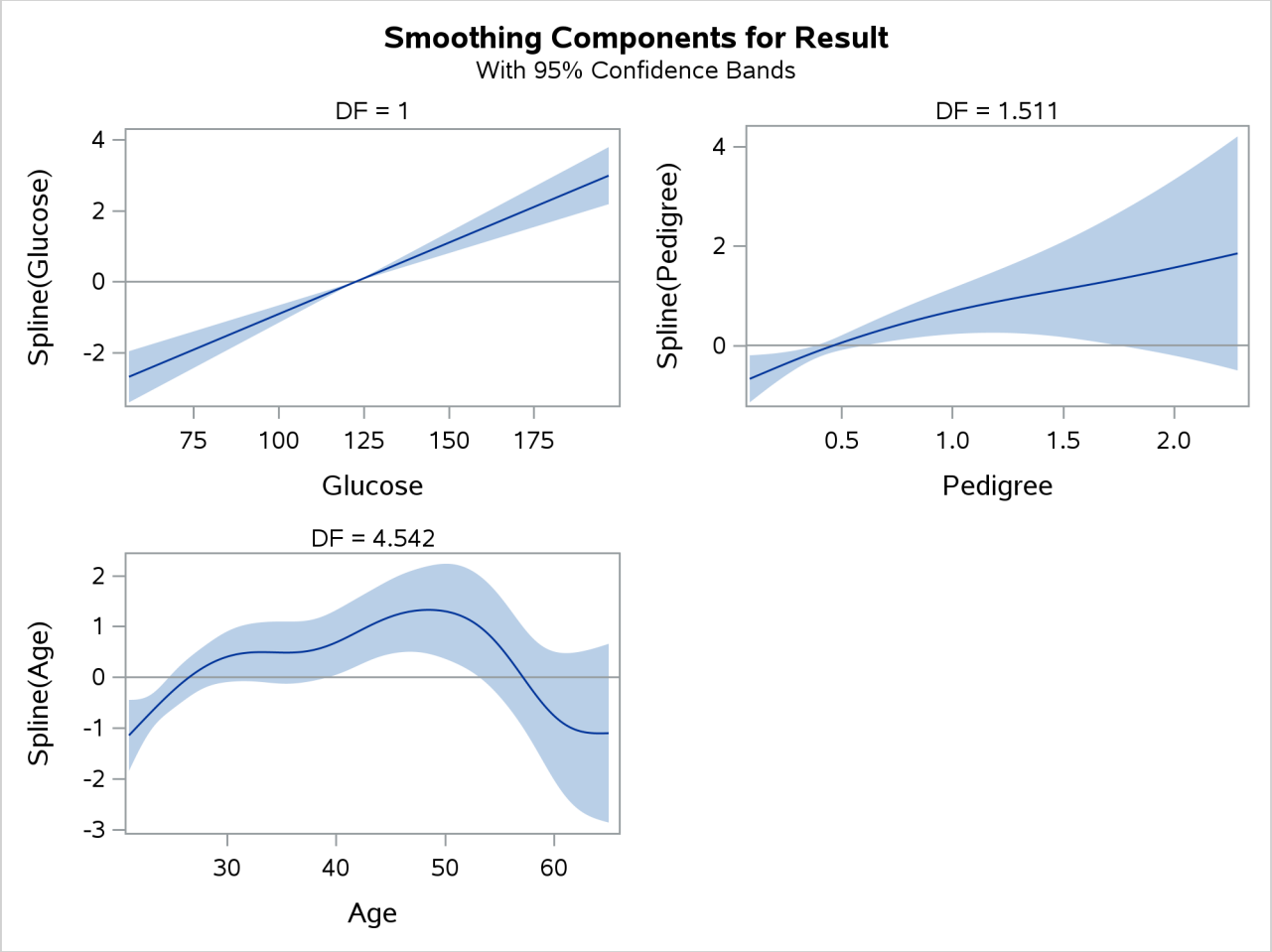
Output 12.2.6 Tests for Smoothing Components

Tests for Smoothing Components

Component	Effective DF	Effective DF for Test	Chi-Square	Pr > ChiSq
Spline(Glucose)	1.00000	1	53.0363	<.0001
Spline(Pedigree)	1.51071	2	9.9354	0.0070
Spline(Age)	4.54170	6	23.0661	0.0008

The smoothing component panel (which is produced by the PLOTS option and is shown in Output 12.2.7) visualizes the spline transformations for the four variables in addition to 95% Bayesian curvewise confidence bands. For Glucose, the spline transformation is almost a straight line. For Pedigree, the spline transformation shows a slightly nonlinear trend. For Age, the dependency is obviously nonlinear.

Output 12.2.7 Smoothing Components Panel



The following statements compute the misclassification error on the test data from the reduced nonparametric logistic regression model that PROC GAMMOD produces:

```
data test;
   set mylib.DiabetesStudyOut(where=(Test=1));
   if ((Pred>0.5 & Diabetes=1) | (Pred<0.5 & Diabetes=0))
   then Error=0;
   else Error=1;
run;

proc freq data=test;
   tables Diabetes*Error/nocol norow;
run;
```

Output 12.2.8 shows the misclassification errors for observations in the test set and observations of each response category.

Output 12.2.8 Crosstabulation Table for Test Set Prediction

Diabetes Study

The FREQ Procedure

Frequency Percent	Table of Diabetes by Error		
		Error	
Diabetes	0	1	Total
0	130 64.36	17 8.42	147 72.77
1	35 17.33	20 9.90	55 27.23
Total	165 81.68	37 18.32	202 100.00

Example 12.3: Nonparametric Tweedie Regression

(View the complete code for this example (gammodex4.sas) in the example repository.)

This example illustrates how you can use the GAMMOD procedure to fit a nonparametric regression model with spline effects for a response variable that has a Tweedie distribution.

You can view this model as a generalized linear model because the Tweedie distribution belongs to the exponential family. The Tweedie distribution is particularly useful for modeling response variables that are continuous for positive values and take the value 0 with a positive probability. For example, in the insurance industry the Tweedie distribution is often assumed for claims, which are positive for customers who have filed claims and are 0 for other customers. You can use the Tweedie distribution to model such response variables without special transformations.

The GENSELECT procedure can fit Tweedie regression models that incorporate linear effects for the covariates. However, as demonstrated in this example, the assumption of linearity can be too restrictive, and you might want to use the GAMMOD procedure to explore the nonlinear dependency structures in the data.

The following DATA step simulates a response variable Y by using a Tweedie distribution and four continuous predictors. Each continuous predictor is sampled independently from the uniform distribution $U(0,1)$. The linear predictor η for the true model is formed by applying additive nonlinear transformations to x_1, x_2, and x_3:

$$\eta = \frac{2\sin(\pi x_1) + 0.8\exp(2x_2) + 0.2x_3^{11}(10(1-x_3))^6 + 10(10x_3)^3(1-x_3)^{10}}{20}$$

The predictor x_4 is a nuisance parameter that does not enter the model. The parameters for the Tweedie distribution are $\phi = 0.4$ and $p = 1.5$, and the link function is $\log\mu = \eta$, where the expected value is $\mathrm{E}(Y) = \mu$ and the variance is $\mathrm{Var}(Y) = \phi\mu^p$. The power parameter p controls the variance of the distribution. When p is between 1 and 2, as in this example, a Tweedie random variable can be generated from a compound Poisson distribution (Smyth 1996).

```
title 'Nonparametric Tweedie Model';
%let phi=0.4;
%let power=1.5;
```

```
data mylib.one;
   do i=1 to 1000;

      /* Sample the predictors */
      x1=ranuni(1);
      x2=ranuni(1);
      x3=ranuni(1);
      x4=ranuni(1);

      /* Apply nonlinear transformations to predictors */
      f1=2*sin(3.14159265*x1);
      f2=exp(2*x2)*0.8;
      f3=0.2*x3**11*(10*(1-x3))**6+10*(10*x3)**3*(1-x3)**10;
      xb=f1+f2+f3;
      xb=xb/20;
      mu=exp(xb);

      /* Compute parameters of compound Poisson distribution */
      lambda=mu**(2-&power)/(&phi*(2-&power));
      alpha=(2-&power)/(&power-1);
      gamma=&phi*(&power-1)*(mu**(&power-1));

      /* Simulate the response */
      rpoi=ranpoi(1,lambda);
      if rpoi=0 then y=0;
      else do;
         y=0;
         do j=1 to rpoi;
            y=y+rangam(1,alpha);
         end;
         y=y*gamma;
      end;
      output;
   end;
run;
```

You can use PROC GENSELECT to fit a parametric model for Y by using the following statements:

```
proc genselect data=mylib.one;
   model y=x1 x2 x3 x4/dist=tweedie;
run;
```

Equivalently, you can also use PROC GAMMOD to fit a parametric linear model by using the PARAM option in the MODEL statement, as shown in the following statements.

```
proc gammod data=mylib.one seed=1234;
   model y=param(x1 x2 x3 x4)/dist=tweedie;
run;
```

The "Fit Statistics" table in Output 12.3.1 shows the summary statistics for the fitted parametric Tweedie model.

Output 12.3.1 Fit Statistics

Nonparametric Tweedie Model

The GAMMOD Procedure

Fit Statistics	
Penalized Log Likelihood	-1178.69507
Roughness Penalty	0
Effective Degrees of Freedom	7.00000
Effective Degrees of Freedom for Error	993.00000
AIC (smaller is better)	2371.39014
AICC (smaller is better)	2371.50305
BIC (smaller is better)	2405.74443

The "Parameter Estimates" table in Output 12.3.2 shows the estimates for regression parameters in addition to dispersion and power parameters (ϕ and p).

Output 12.3.2 Parameter Estimates

Parameter Estimates					
Parameter	DF	Estimate	Standard Error	Chi-Square	Pr > ChiSq
Intercept	1	0.441832	0.067334	43.0565	<.0001
x1	1	-0.043725	0.064965	0.4530	0.5009
x2	1	0.202242	0.064216	9.9188	0.0016
x3	1	-0.253920	0.066625	14.5250	0.0001
x4	1	-0.061098	0.066040	0.8559	0.3549
Dispersion	1	0.416001	0.016987		
Power	1	1.524596	0.048675		

If you are uncertain about the relationship between the linked mean for the fitted Tweedie model and the covariates, you can add spline terms to the model to explore possible nonlinear dependence. The following statements fit a nonparametric Tweedie model and produce a plot of fitted smoothing components.

```
proc gammod data=mylib.one seed=1234 plots;
   model y=spline(x1) spline(x2) spline(x3) spline(x4)/dist=tweedie;
run;
```

The "Fit Statistics" table in Output 12.3.3 shows the summary statistics for the fitted nonparametric Tweedie model. Compared to the parametric model, the nonparametric model has more effective degrees of freedom, but smaller AIC and AICC values. The penalized likelihood value is larger for the nonparametric model. These statistics indicate that the nonparametric model provides a better fit.

Output 12.3.3 Fit Statistics

Nonparametric Tweedie Model

The GAMMOD Procedure

Fit Statistics	
Penalized Log Likelihood	-1157.17502
Roughness Penalty	2.62230
Effective Degrees of Freedom	14.11920
Effective Degrees of Freedom for Error	983.62098
AIC (smaller is better)	2339.96615
AICC (smaller is better)	2340.39965
BIC (smaller is better)	2409.25974
GCV (smaller is better)	0.39845

The "Regression Parameter Estimates" table in Output 12.3.4 lists the estimate for the intercept, in addition to the estimates for the dispersion and power parameters for the Tweedie model. Both variance parameters are close to the true values.

Output 12.3.4 Regression Parameter Estimates

Parameter Estimates					
Parameter	DF	Estimate	Standard Error	Chi-Square	Pr > ChiSq
Intercept	1	0.356825	0.018329	379.0079	<.0001
Dispersion	1	0.400459	0.518084		
Power	1	1.501771	1.514454		

The "Estimates for Smoothing Components" table in Output 12.3.5 shows the fitted information for the spline terms in the model. The degrees of freedom value and the roughness penalty value suggest some moderate nonlinear relationship between the linked mean and the three covariates x_1, x_2, and x_3. The spline term for the nuisance parameter x_4 has a linear form.

Output 12.3.5 Estimates for Smoothing Components

Estimates for Smoothing Components						
Component	Effective DF	Smoothing Parameter	Roughness Penalty	Number of Parameters	Rank of Penalty Matrix	Number of Knots
Spline(x1)	2.10969	0.9560	0.5659	9	10	1000
Spline(x2)	2.38340	0.5604	0.3814	9	10	1000
Spline(x3)	5.62608	0.0145	1.6750	9	10	1000
Spline(x4)	1.00002	133078	3.662E-6	9	10	1000

The "Tests for Smoothing Components" table in Output 12.3.6 shows the approximate Wald test for the four spline terms. The spline term for the nuisance predictor x_4 is not significant.

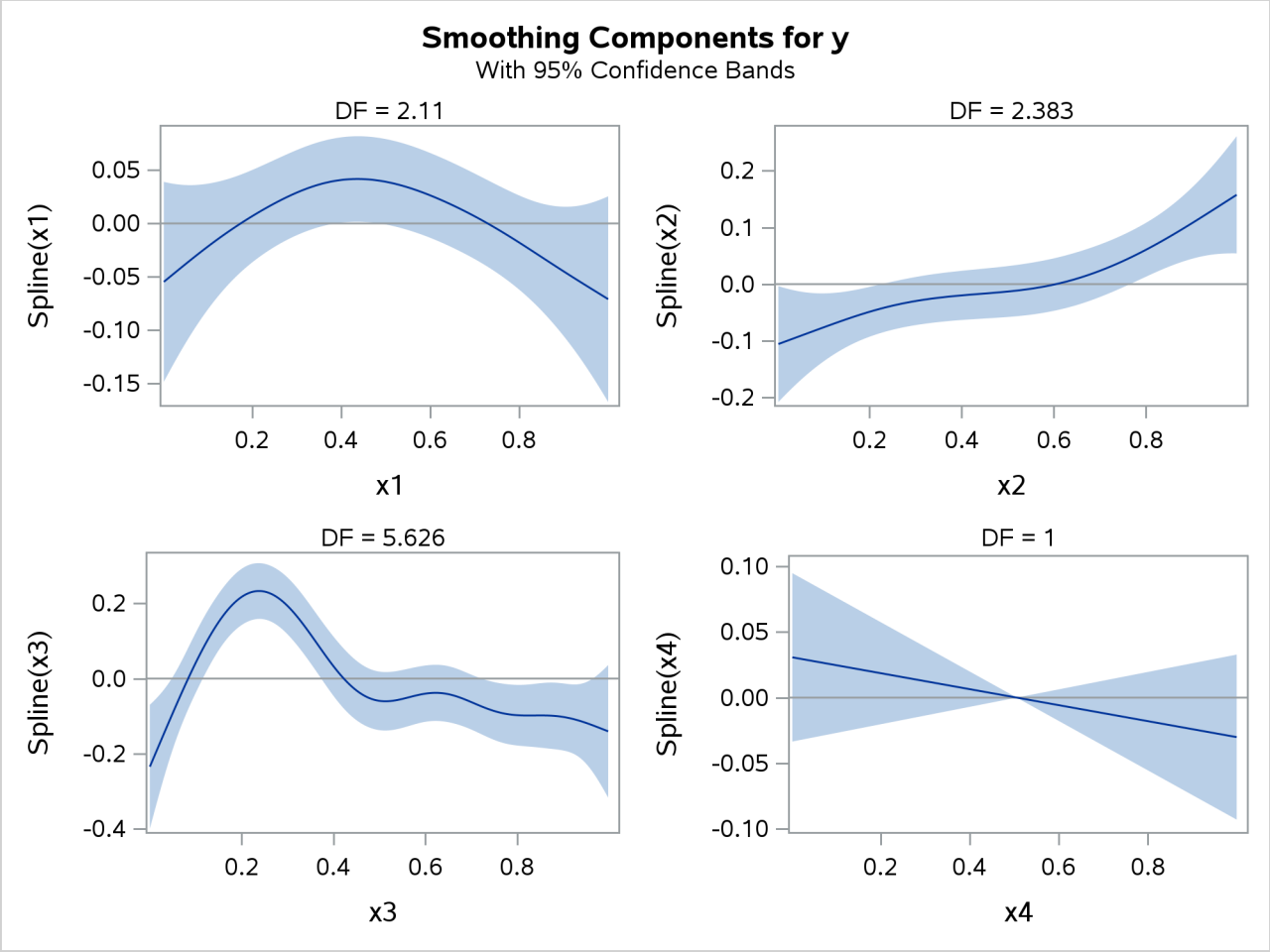
Output 12.3.6 Tests for Smoothing Components

Tests for Smoothing Components

Component	Effective DF	Effective DF for Test	F Value	Pr > F
Spline(x1)	2.10969	3	4.76	0.0027
Spline(x2)	2.38340	3	13.64	<.0001
Spline(x3)	5.62608	7	49.02	<.0001
Spline(x4)	1.00002	1	0.88	0.3484

Output 12.3.7 displays the "Smoothing Component Panel" for all the spline terms used in the model. For x_1, x_2, and x_3, the fitted curves are reasonably smooth and close to the true functions. For x_4, the spline plot shows a strictly linear fit, with 95% Bayesian confidence bands covering the horizontal line at 0. This reinforces the conclusion that x_4 does not contribute to the model and can be removed from further analysis.

Output 12.3.7 Smoothing Component Panel



References

Craven, P., and Wahba, G. (1979). "Smoothing Noisy Data with Spline Functions." *Numerical Mathematics* 31:377–403.

Duchon, J. (1976). "Fonctions-spline et espérances conditionnelles de champs gaussiens [Spline functions and conditional expectations of Gaussian fields]." *Annales scientifiques de l'Université de Clermont-Ferrand 2, Série Mathématique* 14:19–27.

Duchon, J. (1977). "Splines Minimizing Rotation-Invariant Semi-norms in Sobolev Spaces." In *Constructive Theory of Functions of Several Variables*, edited by W. Schempp and K. Zeller, 85–100. New York: Springer-Verlag.

Gu, C., and Wahba, G. (1991). "Minimizing GCV/GML Scores with Multiple Smoothing Parameters via the Newton Method." *SIAM Journal on Scientific Computing* 12:383–398.

Lichman, M. (2013). "UCI Machine Learning Repository." School of Information and Computer Sciences, University of California, Irvine. http://archive.ics.uci.edu/ml.

Nelder, J. A., and Wedderburn, R. W. M. (1972). "Generalized Linear Models." *Journal of the Royal Statistical Society, Series A* 135:370–384.

Nychka, D. (1988). "Bayesian Confidence Intervals for Smoothing Splines." *Journal of the American Statistical Association* 83:1134–1143.

Pace, R. K., and Barry, R. (1997). "Quick Computation of Spatial Autoregressive Estimators." *Geographical Analysis* 29:232–247.

Ruppert, D., Wand, M. P., and Carroll, R. J. (2003). *Semiparametric Regression*. Cambridge: Cambridge University Press.

Smyth, G. K. (1996). "Regression Analysis of Quantity Data with Exact Zeros." In *Proceedings of the Second Australia-Japan Workshop on Stochastic Models in Engineering, Technology, and Management*, edited by R. J. Wilson, S. Osaki, and D. N. P. Murthy, 572–580. Queensland, Australia: Technology Management Centre, University of Queensland.

Vlachos, P. (1998). "StatLib—Datasets Archive." http://lib.stat.cmu.edu/datasets/.

Wahba, G. (1983). "Bayesian 'Confidence Intervals' for the Cross Validated Smoothing Spline." *Journal of the Royal Statistical Society, Series B* 45:133–150.

Wood, S. (2003). "Thin Plate Regression Splines." *Journal of the Royal Statistical Society, Series B* 65:95–114.

Wood, S. (2004). "Stable and Efficient Multiple Smoothing Parameter Estimation for Generalized Additive Models." *Journal of the American Statistical Association* 99:673–686.

Wood, S. (2006). *Generalized Additive Models*. Boca Raton, FL: Chapman & Hall/CRC.

Wood, S. (2008). "Fast Stable Direct Fitting and Smoothness Selection for Generalized Additive Models." *Journal of the Royal Statistical Society, Series B* 70:495–518.

Wood, S. (2011). "Fast Stable Restricted Maximum Likelihood and Marginal Likelihood Estimation of Semiparametric Generalized Linear Models." *Journal of the Royal Statistical Society, Series B* 73:3–36.

Wood, S. (2012). "On *p*-Values for Smooth Components of an Extended Generalized Additive Model." *Biometrika* 1–8. http://biomet.oxfordjournals.org/content/early/2012/10/18/biomet.ass048.abstract.

Xiang, D., and Wahba, G. (1996). "A Generalized Approximate Cross Validation for Smoothing Splines with Non-Gaussian Data." *Statistica Sinica* 6:675–692.

Chapter 13
GAMSELECT Procedure

Contents

Overview: GAMSELECT Procedure **642**
PROC GAMSELECT Features 642
PROC GAMSELECT Compared with Other SAS Procedures 643
Using CAS Sessions and CAS Engine Librefs 645
Getting Started: GAMSELECT Procedure **646**
Syntax: GAMSELECT Procedure **651**
PROC GAMSELECT Statement 651
BY Statement 654
CLASS Statement 654
DISPLAY Statement 654
DISPLAYOUT Statement 655
FREQ Statement 656
MODEL Statement 656
OUTPUT Statement 663
PARTITION Statement 665
SELECTION Statement 665
STORE Statement 669
WEIGHT Statement 669
Programming Statements 669
Details: GAMSELECT Procedure **671**
Missing Values 671
Penalized B-Splines 671
Natural Cubic Splines 673
Generalized Additive Models 673
Model Selection Methods 674
Displayed Output 677
ODS Table Names 680
ODS Graphics 681
Examples: GAMSELECT Procedure **682**
Example 13.1: Testing Model Fit with Partitioned Data 682
Example 13.2: Options to Control Boosting Model Fit 686
Example 13.3: Options to Control Shrinkage Selection 696
References **700**

Overview: GAMSELECT Procedure

The GAMSELECT procedure fits and performs model selection for generalized additive models in SAS Viya.

Generalized additive models are extensions of generalized linear models. They relax the generalized linear models' assumption of linearity by allowing spline terms that characterize nonlinear dependency structures. PROC GAMSELECT supports model selection by componentwise functional gradient descent (Friedman 2001; Bühlmann and Hothorn 2007) and the shrinkage method by using the penalized likelihood with sparsity-inducing penalties (Meier, Van de Geer, and Bühlmann 2009). It fits models for standard distributions in the exponential family, such as the normal, Poisson, gamma, and binary distributions.

PROC GAMSELECT Features

PROC GAMSELECT offers the following basic features:

- estimates the regression parameters of a generalized additive model by using penalized likelihood estimation or componentwise functional gradient descent
- enables model building (variable selection) through the SELECTION statement
- provides model-building syntax in the CLASS statement and effect-based parametric effects in the MODEL statement, which are used in other SAS/STAT analytic procedures (in particular, the GLM, LOGISTIC, GLIMMIX, and MIXED procedures)
- provides response-variable options
- provides control options for constructing a spline term
- provides multiple link functions for any distribution
- provides a WEIGHT statement for weighted analysis
- provides a FREQ statement for grouped analysis
- provides an OUTPUT statement to produce a data table that contains predicted values and other observationwise statistics
- provides a PARTITION to divide data into training, validation, and test data roles
- produces graphs via ODS Graphics
- allows programming statements so that you can create variables on the fly for model effects and splines, frequency, offset, response, and weight

Because the GAMSELECT procedure runs on SAS Cloud Analytic Services (CAS), it also does the following:

- enables you to run on a cluster of machines that distribute the data and the computations
- enables you to run in single-machine mode
- exploits all the available cores and concurrent threads (for information about how PROC GAMSELECT uses threads, see the section "Multithreading" on page 102 in Chapter 2, "Shared Concepts")

PROC GAMSELECT Compared with Other SAS Procedures

In SAS Viya, both the GAMSELECT and GAMMOD procedures fit generalized additive models. The GAMSELECT procedure and the GRADBOOST procedure both fit predictive models by using methods that are based on a generic functional gradient descent, or boosting, algorithm (Friedman 2001).

PROC GAMSELECT Compared with the GAMMOD Procedure

Both the GAMSELECT and GAMMOD procedures in SAS Visual Statistics have been developed specifically for SAS Viya. Both procedures are designed to run on a cluster of machines that distribute the data and the computations. Both procedures perform computations in multiple threads. However, the GAMMOD and GAMSELECT procedures use different approaches for constructing spline basis expansions and fitting generalized additive models. PROC GAMSELECT fits and performs model selection for generalized additive models but does not support model inference. In general, you should not expect similar results from these two procedures. Table 13.1 summarizes the main differences.

Table 13.1 PROC GAMSELECT Compared with PROC GAMMOD

Functionality	PROC GAMSELECT	PROC GAMMOD
Constructing spline basis expansions	Uses B-spline basis expansions for the boosting method and natural cubic splines for the shrinkage method. The boosting method supports univariate and bivariate spline terms with nonnegative degrees.	Uses thin-plate regression splines to construct basis expansions for each spline term, and each term allows multiple variables
Fitting generalized additive models	Fits models and performs model selection by boosting and shrinkage methods. The boosting method uses a componentwise functional gradient descent algorithm for models with parametric effects and spline terms. The shrinkage method optimizes a penalized likelihood with sparsity-inducing penalties to control model complexity and smoothness. Penalty terms can be either fixed or varied.	Fits models that have fixed smoothness terms by optimizing penalized likelihood. For models that have varying smoothness terms, PROC GAMMOD estimates smoothing parameters simultaneously by optimizing global model fit criteria such as generalized cross validation (GCV).

Table 13.1 *continued*

Functionality	PROC GAMSELECT	PROC GAMMOD
Model degrees of freedom and inference	Does not compute model degrees of freedom or perform inference	Uses the trace of the global degrees-of-freedom matrix. A global Bayesian posterior covariance matrix is available. The confidence limits for each observation's prediction are available, in addition to componentwise confidence limits.
Multithreading scheme	Specifically designed to operate in SAS Viya and performs computations in multiple threads	Specifically designed to operate in SAS Viya and performs computations in multiple threads

PROC GAMSELECT Compared with the GRADBOOST Procedure

Both the GAMSELECT and GRADBOOST procedures fit predictive models by using a generic functional gradient descent, or boosting, algorithm (Friedman 2001). However, how the procedures implement the boosting algorithms and the types of predictive models that they fit differ substantially.

The GAMSELECT procedures uses a componentwise implementation of the generic functional gradient descent algorithm. The effects in the model can be parametric effects, univariate spline terms, or bivariate spline terms. The parameter estimates for each component are additive across the iterations of the boosting algorithm. As a result, the predictive model that is fit by the boosting method in PROC GAMSELECT is a generalized additive model that consists of parametric effects and spline terms selected at the boosting iterations. For more information about the componentwise boosting algorithm that PROC GAMSELECT implements, see the section "Boosting" on page 674.

The boosting algorithm that is implemented by the GRADBOOST procedure fits a decision tree at each iteration. The tree can consist of both interval and nominal predictors. The decision trees that are fit at each iteration are not additive in the estimated parameters. As a result, the predictive model that is fit by PROC GRADBOOST is a tree ensemble. For more information about the boosting algorithm that PROC GRADBOOST implements, see the section "Boosting" (Chapter 17, *SAS Viya: Machine Learning Procedures*).

Using CAS Sessions and CAS Engine Librefs

SAS Cloud Analytic Services (CAS) is the analytic server and associated cloud services in SAS Viya. This section describes how to create a CAS session and set up a CAS engine libref that you can use to connect to the CAS session. It assumes that you have a CAS server already available; contact your system administrator if you need help starting and terminating a server. This CAS server is identified by specifying the host on which it runs and the port on which it listens for communications. To simplify your interactions with this CAS server, the host information and port information for the server are stored as SAS option values that are retrieved automatically whenever this CAS server needs to be accessed. You can examine the host and port values for the server at your site by using the following statements:

```
proc options option=(CASHOST CASPORT);
run;
```

In addition to starting a CAS server, your system administrator might also have created a CAS session and a CAS engine libref for your use. You can define your own sessions and CAS engine librefs that connect to the CAS server as shown in the following statements:

```
cas mysess;
libname mylib cas sessref=mysess;
```

The CAS statement creates the CAS session named mysess, and the LIBNAME statement creates the mylib CAS engine libref that you use to connect to this session. It is not necessary to explicitly name the CASHOST and CASPORT of the CAS server in the CAS statement, because these values are retrieved from the corresponding SAS option values.

If you have created the mysess session, you can terminate it by using the TERMINATE option in the CAS statement as follows:

```
cas mysess terminate;
```

For more information about the CAS and LIBNAME statements, see the section "Introduction to Shared Concepts" on page 6 in Chapter 2, "Shared Concepts."

Getting Started: GAMSELECT Procedure

(View the complete code for this example at https://github.com/sascommunities/doc-supplement-casstat/tree/main/Examples/gamselectgs1.sas.)

NOTE: Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections "Using CAS Sessions and CAS Engine Librefs" on page 7 and "Loading a SAS Data Set onto a CAS Server" on page 7 in Chapter 2, "Shared Concepts."

Given a response with many potential predictors, this example shows how you can use the GAMSELECT procedure not only to find which are the few predictors that really do affect the response, but also to determine what functional form that effect takes.

The following DATA step creates the data table getStarted, which consists of 20,000 observations on a continuous response variable (y) and 90 continuous variables (x1–x90) in the current CAS engine libref named mylib:

```
data mylib.getStarted;
  drop i j w;

  array x{90};
  call streaminit(4321);
  pi=constant("pi");

  do i=1 to 20000;
      u = ranuni(12345);
      do j=1 to dim(x);
         w    = ranuni(12345);
         x{j} = (w + u)/2;
      end;
      f1 = x1;
      f2 = (2*x2-1)**2;
      f3 = sin(2 * pi * x3)/(2-sin(2*pi*x3));
      f4 = 0.1*sin(2*pi*x4) + 0.2*cos(2*pi*x4)+0.3*sin(2*pi*x4)**2
         + 0.4*cos(2*pi*x4)**3+0.5*sin(2*pi*x4)**3;
      linp = 5*f1+3*f2+4*f3+6*f4;
      y =  linp + 1.32 * rand("Normal");
      output;
   end;
run;
```

These statements assume that your CAS engine libref is named mylib, but you can substitute any appropriately defined CAS engine libref. For the simulation, only the variables x1–x4 affect the outcome y, and the variables x1–x90 are all positively correlated.

The following statements fit a nonparametric model by using spline transformations of the variables x1–x10:

```
ods graphics on;
proc gamselect data= mylib.getStarted plots=all;
   model y = spline(x1 ) spline(x2 ) spline(x3 ) spline(x4 ) spline(x5 )
             spline(x6 ) spline(x7 ) spline(x8 ) spline(x9 ) spline(x10)
             ;
```

```
   selection method=boosting;
run;
```

The MODEL statement and the SELECTION statement are required when you use PROC GAMSELECT. The output from this analysis is presented in Figure 13.1 through Figure 13.6.

Figure 13.1 displays the "Model Information" table. The GAMSELECT procedure uses the boosting method to fit and select the model. For more information about the boosting selection method, see the section "Boosting" on page 674. The response variable y is modeled using a normal distribution whose mean is modeled by an identity link function.

Figure 13.1 Model Information

The GAMSELECT Procedure

Model Information	
Data Source	GETSTARTED
Response Variable	y
Distribution	Normal
Link Function	Identity
Selection Method	Boosting
Random Number Seed	2816737677

Figure 13.2 displays the "Number of Observations" table. All 20,000 observations in the data table are used in the analysis. For data tables that have missing or invalid values, the number of observations that are used might be less than the number of observations that are read.

Figure 13.2 Number of Observations

Number of Observations Read	20000
Number of Observations Used	20000

Figure 13.3 displays the "Iteration Summary" table. The boosting algorithm executed 500 iterations. The selected model corresponds to the final boosting iteration and includes four effects.

Figure 13.3 Iteration Summary

Iteration Summary	
Iterations Executed	500
Selected Iteration	500
Number of Selected Effects	4

Figure 13.4 displays the "Selected Effects" table. The table displays the effects that are included in the selected model, the first iteration at which an effect entered the model, and the number of iterations at which the effect was selected. The four spline terms that are constructed from the variables x1–x4 are selected for the final model.

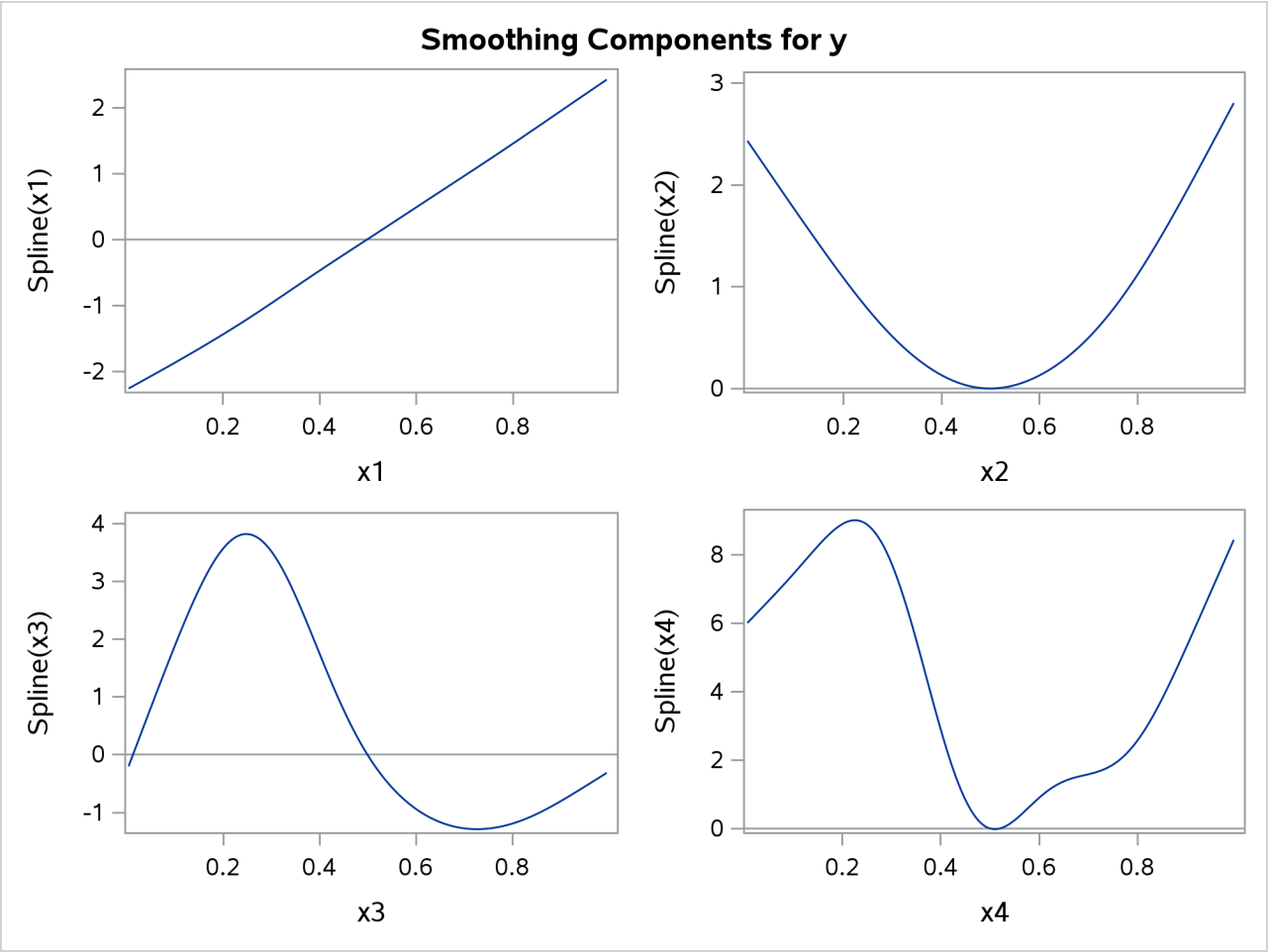
Figure 13.4 Selected Effects

Selected Effects

Effect	Entry Iteration	Times Selected
Spline(x4)	1	293
Spline(x3)	9	113
Spline(x1)	34	54
Spline(x2)	36	40

Figure 13.5 displays the smoothing component panel for the spline terms in the selected model. It displays predicted spline curves for each of the spline terms in the selected model.

Figure 13.5 Smoothing Component Panel



Finally, the procedure displays the table shown in Figure 13.6, which shows the amount of time (in seconds) that PROC GAMSELECT required to perform the various tasks in the analysis.

Figure 13.6 Procedure Timing

Task Timing		
Task	**Seconds**	**Percent**
Parsing	0.02	1.21%
Reading and Levelizing Data	0.01	0.66%
Creating Basis Expansions	0.04	1.77%
Fitting Model	1.99	96.07%
Producing Graphics	0.00	0.05%
Miscellaneous	0.01	0.26%
Cleaning Up	0.00	0.02%
Total	2.07	100.00%

The following code uses the macro function SplinePrefixList to specify a model for the variable y by using spline transformations of all 90 explanatory variables, x1–x90:

```
%macro SplinePrefixList(prefix,n);
   %do i = 1 %to &n;
      spline(&prefix.&i)
      %end;
%mend;


ods graphics on;
proc gamselect data= mylib.getStarted plots=all;
   model y = %SplinePrefixList(x,90);
   selection method=boosting;
run;
```

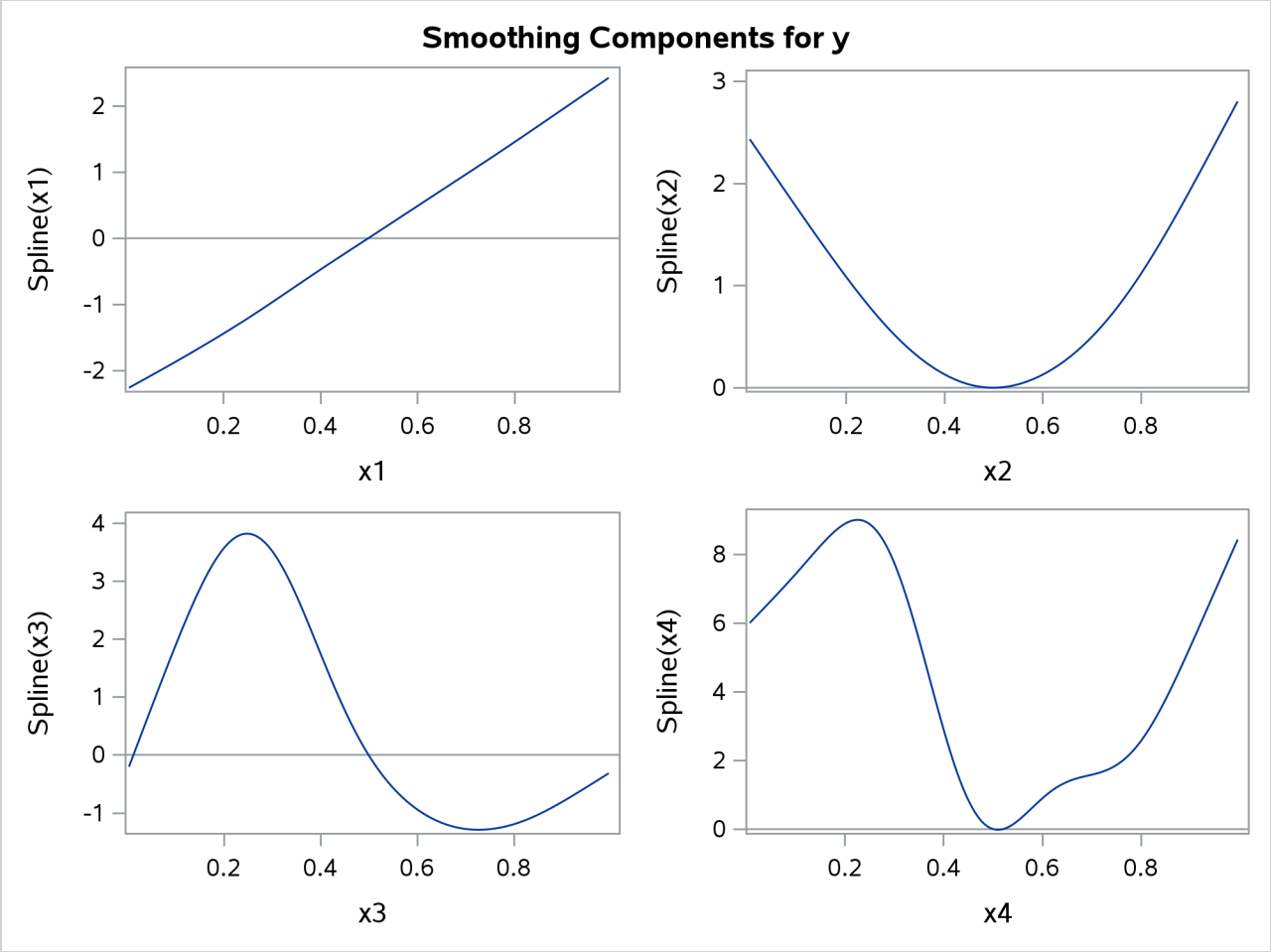
Figure 13.7 and Figure 13.8 show that the analysis that is run using all 90 predictors selects the same model that includes only the four spline terms constructed from the variables x1–x4.

Figure 13.7 Selected Effects

The GAMSELECT Procedure

Selected Effects		
Effect	**Entry Iteration**	**Times Selected**
Spline(x4)	1	293
Spline(x3)	9	113
Spline(x1)	34	54
Spline(x2)	36	40

Figure 13.8 Smoothing Component Panel



Syntax: GAMSELECT Procedure

The following statements are available in the GAMSELECT procedure:

PROC GAMSELECT < *options* > **;**
BY *variables* **;**
CLASS *variable* < **(***options***)** > . . . < *variable* < **(***options***)** > > < / *global-options* > **;**
DISPLAY < *table-list* > < / *options* > **;**
DISPLAYOUT *table-spec-list* < / *options* > **;**
FREQ *variable* **;**
MODEL *response* < **(***response-options***)** > **=** < **PARAM(***effects***)** >
< **SPLINE(***variable* < / *spline-options* >**)** >
< **SPLINE(***variable1 variable2* < / *spline-options* >**)** > < / *model-options* > **;**
MODEL *events* / *trials* **=** < **PARAM(***effects***)** >
< **SPLINE(***variable* < / *spline-options* >**)** >
< **SPLINE(***variable1 variable2* < / *spline-options* >**)** > < / *model-options* > **;**
OUTPUT OUT=*libref.data-table*
< **COMPONENT** >< **COMPVARS=(***variables***)** > < **COPYVARS=(***variables***)** >
< *keyword* < **=***name* > >. . . < *keyword* < **=***name* > > **;**
PARTITION *partition-options* **;**
SELECTION < **METHOD=***method* < **(***method-options***)** > > **;**
STORE < **OUT=** >*libref.data-table* **;**
WEIGHT *variable* **;**
Programming statements **;**

The PROC GAMSELECT statement, one MODEL statement, and one SELECTION statement are required. The CLASS statement can appear multiple times. If a CLASS statement is specified, it must precede the MODEL statement. The following sections describe the PROC GAMSELECT statement and then describe the other statements in alphabetical order.

PROC GAMSELECT Statement

PROC GAMSELECT < *options* > **;**

The PROC GAMSELECT statement invokes the procedure. Table 13.2 summarizes the available options in the PROC GAMSELECT statement by function. The options are then described fully in alphabetical order.

Table 13.2 PROC GAMSELECT Statement Options

Option	Description
Basic Options	
DATA=	Specifies the input data table
SEED=	Sets the seed for pseudorandom number generation
Display Options	
ITDETAILS	Displays the “Iteration Details” table
NOCLPRINT	Limits or suppresses the display of classification variable levels

Table 13.2 *continued*

Option	Description
PLOTS=	Controls plots that are produced through ODS Graphics
Tolerance Options	
SINGCHOL=	Tunes the singularity criterion for Cholesky decompositions
SINGULAR=	Tunes the general singularity criterion

You can specify the following *options* in the PROC GAMSELECT statement.

DATA=*libref.data-table*
names the input data table for PROC GAMSELECT to use. The default is the most recently created data table. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to the data, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 645.

data-table specifies the name of the input data table.

ITDETAILS
generates the "Iteration Details" table.

NOCLPRINT< =*number* **>**
suppresses the display of the "Class Level Information" table if you do not specify *number*. If you specify *number*, the values of the classification variables are displayed for only those variables whose number of levels is less than *number*. Specifying a *number* helps reduce the size of the "Class Level Information" table if some classification variables have a large number of levels.

PLOTS < (*global-plot-option***) > < =** *plot-requests* **< (***option***) > >**
controls the plots that are produced through ODS Graphics. When ODS Graphics is enabled, PROC GAMSELECT produces by default a panel of plots of partial prediction curves or surfaces of smoothing components. For example:

```
ods graphics on;

proc gamselect plots;
   model y=spline(x1) spline(x2);
   selection;
run;

ods graphics off;
```

You can specify the following *global-plot-option*, which applies to the smoothing component plots that the GAMSELECT procedure generates:

UNPACK

UNPACKPANEL

suppresses paneling. By default, multiple smoothing component plots can appear in some output panels. Specify this option to display each plot individually.

You can specify the following *plot-requests*:

ALL

produces all default plots.

COMPONENTS < (*component-option***) >**

plots a panel of smoothing components of the fitted model. You can specify the following *component-option*:

COMMONAXES

requests that the smoothing component plots, except for bivariate contour plots, use a common vertical axis. This option enables you to visually judge the relative effect size.

NONE

suppresses all plots.

SEED=*number*

specifies an integer to use to start the pseudorandom number generator for subset sampling from observations to form knots if necessary, for random cross validation if the boosting method is used for model selection, and for random data partitioning if a separate random number seed is not supplied. If you do not specify this option or if *number* ≤ 0, the seed is generated by reading the time of day from the computer's clock. If you specify this option with a nonzero value and you also specify the PARTITION statement without specifying the random number seed there, then the same random number seed is used to partition the input data. Note that the random sampling of observations for cross validation and data partitioning depends on how the data are distributed across machines and threads. Therefore, even if you specify the same seed, results might differ between runs of PROC GAMSELECT.

SINGCHOL=*number*

tunes the singularity criterion in Cholesky decomposition and matrix inversion operations. The default *number* is 1E4 times the machine epsilon; this product is approximately 1E–12 on most computers.

SINGULAR=*number*

tunes the singularity criterion that is used to solve ridge regressions in searching upper bounds of the sparsity parameter and the smoothness parameter, if you specify the shrinkage method of model selection. The default *number* is 1E4 times the machine epsilon; this product is approximately 1E–12 on most computers.

BY Statement

BY *variables* **;**

You can specify a BY statement in PROC GAMSELECT to obtain separate analyses of observations in groups that are defined by the values of the BY variables. If you specify more than one BY statement, only the last one specified is used. For more information, see the discussion of BY-group processing in the "Grouping Data" section of *SAS Programmers Guide: Essentials*.

CLASS Statement

CLASS *variable* < **(***options***)** >. . . < *variable* < **(***options***)** > > < / *global-options* > **;**

The CLASS statement names the classification variables to be used as explanatory variables in the analysis. You can list the response variable for binary models in the CLASS statement, but this is not required. Table 13.3 summarizes the values that you can use for either an *option* or a *global-option*. The options are fully documented in the section "CLASS Statement" on page 9 in Chapter 2, "Shared Concepts."

Table 13.3 CLASS Statement Options

Option	Description
DESCENDING	Reverses the sort order
MISSING	Treats missing values as valid levels
ORDER=	Specifies the sort order for the levels
PARAM=	Specifies the parameterization of the variable
REF=	Specifies the reference level of the variable

DISPLAY Statement

DISPLAY < *table-list* > < / *options* > **;**

The DISPLAY statement enables you to specify a list of display tables to display or exclude. This statement is similar to the ODS SELECT, ODS EXCLUDE, and ODS TRACE statements. However, the DISPLAY statement can improve performance when a large number of tables could be generated (such as in BY-group processing). The procedure processes the DISPLAY statement on a CAS server and thus sends only a subset of ODS tables to the SAS client. Because ODS statements are processed on a SAS client, first all the generated display tables are sent to the client, and then the client creates a subset.

If you use both DISPLAY and ODS statements together, the DISPLAY statement takes precedence over the ODS statements. Note that the ODS EXCLUDE statement processes tables that are sent to the client after they have been filtered by the DISPLAY statement. In some cases, it might appear that the ODS EXCLUDE statement is taking precedence because it can further filter the tables. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

You can specify the *table-list* as a list of table names, paths, partial pathnames, and regular expressions.

The table names that you can specify are listed in the section "ODS Table Names" on page 680. A path is a table name that is prefixed with dot-separated grouping information. For example, a SelectionSummary table that a procedure produces during a selection routine might have the path *Bygroup1.Summary.SelectionSummary*. A partial pathname does not include all groups; for example, *SelectionSummary* and *Summary.SelectionSummary* are partial pathnames for *Bygroup1.Summary.SelectionSummary*.

When you specify a table name or partial pathname, all display tables whose paths end in the specified name are selected for display or exclusion. For example, both *SelectionSummary* and *Summary.SelectionSummary* select *Bygroup1.Summary.SelectionSummary*.

A regular expression is enclosed in forward slashes (/). For example, specifying "/tions/" selects all pathnames that contain the substring "tions"; in particular, the *Bygroup1.Summary.SelectionSummary* table is selected. Specifying "!/tions/" selects all pathnames that do not contain the substring "tions"; in particular, the *Bygroup1.Summary.SelectionSummary* table is not selected.

You can specify the following *options* after a slash (/):

CASESENSITIVE
performs a case-sensitive comparison of table names in the *table-list* to display table names when tables are subsetted for display. To preserve case, you must enclose table names in the *table-list* in quotation marks.

EXCLUDE
displays all display tables except those that you specify in the *table-list*.

EXCLUDEALL
suppresses display of all tables. This option takes precedence over the other options.

TRACE
displays the display table names, labels, and paths.

DISPLAYOUT Statement

DISPLAYOUT *table-spec-list* < / *options* > **;**

The DISPLAYOUT statement enables you to create output tables from your displayed output. This statement is similar to the ODS OUTPUT statement. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

The *table-spec-list* specifies a list of output tables to create. Each entry in the list has either a *key=value* format or a *key* format:

key=value specifies *key* as the ODS table name, path, or partial pathname, and specifies *value* as the output table name.

key specifies *key* as the ODS table name and also as the output table name.

The ODS table names that you can specify are listed in the section "ODS Table Names" on page 680. You cannot specify the ODS table named OutputCasTables in the *table-spec-list*.

Table names and partial pathnames are discussed under the DISPLAY statement. The DISPLAYOUT statement does not support regular expressions.

You can specify the following *options* after a slash (/):

INCLUDEALL
creates output tables for all display tables. The name of the created output table is the same as the name of the corresponding display table. If you specify this option, the *table-spec-list* specification is ignored.

NOREPLACE
does not replace any existing output table of the same name.

REPEATED
replicates all output tables on all nodes.

FREQ Statement

FREQ *variable* **;**

The *variable* in the FREQ statement identifies a numeric variable in the input data table that contains the frequency of occurrence of each observation. PROC GAMSELECT treats each observation as if it appears f times, where f is the value of the FREQ *variable* for the observation. If f is not an integer, it is truncated to an integer. If f is less than 1 or missing, the observation is not used in the analysis. When you omit the FREQ statement, each observation is assigned a frequency of 1.

MODEL Statement

MODEL *response* < **(***response-options***)** > **=** < **PARAM(***effects***)** >
< **SPLINE(***variable* < / *spline-options* >**)** >
< **SPLINE(***variable1 variable2* < / *spline-options* >**)** > < / *model-options* > **;**

MODEL *events* / *trials* **=** < **PARAM(***effects***)** >
< **SPLINE(***variable* < / *spline-options* >**)** >
< **SPLINE(***variable1 variable2* < / *spline-options* >**)** > < / *model-options* > **;**

The MODEL statement specifies the response (dependent or target) variable and the predictor (independent or explanatory) effects of the model. You can specify the response as a single variable or as a ratio of two variables, which are denoted as *events/trials*. The first form applies to all distribution families; the second form applies only to summarized binomial response data. When you have binomial data, the *events* variable contains the number of positive responses (or events), and the *trials* variable contains the number of trials. The values of both *events* and (*trials* – *events*) must be nonnegative, and the value of *trials* must be positive. If you specify a single *response* variable in a CLASS statement, then the response is assumed to be binary. For more information about the *response-options*, see the section "Response Variable Options" on page 658.

You can use the PARAM(*effects*) option to specify parametric effects that are constructed from variables in the input data; this option can appear multiple times. For information about constructing the model effects, see the section "Specification and Parameterization of Model Effects" on page 64 in Chapter 2, "Shared Concepts."

You can use the SPLINE(*variable*) or SPLINE(*variable1 variable2*) option to specify nonparametric spline effects that are constructed from variables in the input data. Only continuous variables (not classification

variables) can be specified in spline effects. For more information about the options that enable you to control the construction of spline effects, see the section "Spline Effect Options" on page 659.

If only spline effects are present, PROC GAMSELECT builds a nonparametric additive model. If both spline and parametric effects are present, PROC GAMSELECT builds a semiparametric model by using the parametric effects as the linear part of the model. If only parametric effects are present, PROC GAMSELECT uses the boosting method to build a parametric generalized linear model by using the terms inside the parentheses of all PARAM(*effects*) options. The shrinkage method requires spline terms.

Three sets of options are available in the MODEL statement. The *response-options* determine how the GAMSELECT procedure models probabilities for binary data. The *spline-options* controls how each spline term forms basis expansions. The *model-options* control other aspects of model formation and inference. Table 13.4 summarizes these options, and the sections after the table describe them in detail.

Table 13.4 MODEL Statement Options

Option	Description
Response Variable Options	
DESCENDING	Reverses the response categories
EVENT=	Specifies the event category
ORDER=	Specifies the sort order
REF=	Specifies the reference category
Spline Effect Options	
DEGREE=	Specifies the degree of the spline effect
DETAILS	Produces detailed spline information
DF=	Specifies the fixed degrees of freedom
DIFFORDER=	Specifies the order of the difference penalty for a B-spline
KNOTS=	Specifies the knots to use for constructing the spline
LOWEREXKNOTS=	Specifies the lower exterior knots
MAXKNOTS=	Specifies the maximum number of knots to use for constructing the spline
SMOOTH=	Specifies a fixed smoothing parameter
UPPEREXKNOTS=	Specifies the upper exterior knots
WEIGHT1=	Specifies the sparsity penalty strength
WEIGHT2=	Specifies the smoothness penalty strength
Model Options	
ALLOBS	Uses values of spline variables from all data roles to construct spline basis functions
DISPERSION=	Specifies the fixed dispersion parameter
DISTRIBUTION=	Specifies the response distribution
INITIALPHI=	Specifies the starting value of the dispersion parameter
LINK=	Specifies the link function
MAXPHI=	Specifies the upper bound for searching the dispersion parameter
MINPHI=	Specifies the lower bound for searching the dispersion parameter
OFFSET=	Specifies the offset variable

Response Variable Options

Response variable options determine how the GAMSELECT procedure models probabilities for binary data.

You can specify the following *response-options* by enclosing them in parentheses after the *response* variable:

DESCENDING
DESC
reverses the order of the response categories. If you specify both the DESCENDING and ORDER= options, PROC GAMSELECT orders the response categories according to the ORDER= option and then reverses that order.

EVENT='*category***' | FIRST | LAST**
specifies the event category for the binary response model. PROC GAMSELECT models the probability of the event category. This option has no effect when there are more than two response categories.

You can specify any of the following values:

'*category*' specifies that observations whose value matches *category* (formatted, if a format is applied) in quotation marks represent events in the data. For example, the following statements specify that observations that have a formatted value of '1' represent events in the data. The probability that is modeled by the GAMSELECT procedure is thus the probability that the variable def takes the (formatted) value '1'.

```
proc gamselect data=mylib.MyData;
   class A B C;
   model def(event ='1') = param(A B) spline(x1) spline(x2);
   selection;
run;
```

FIRST designates the first ordered category as the event.

LAST designates the last ordered category as the event.

By default, EVENT=FIRST.

ORDER=FORMATTED | FREQ | INTERNAL
specifies the sort order for the levels of the *response* variable. You can specify the following values:

FORMATTED sorts the levels by external formatted value, except for numeric variables that have no explicit format, which are sorted by their unformatted (internal) value. For numeric variables for which you have supplied no explicit format (that is, for which there is no corresponding FORMAT statement in the current PROC GAMSELECT run or in the DATA step that created the data table), the levels are ordered by their internal (numeric) value. The sort order is machine-dependent.

FREQ sorts the levels by descending frequency count (levels that have the most observations come first in the order).

INTERNAL sorts the levels by unformatted value. The sort order is machine-dependent.

By default, ORDER=FORMATTED.

For more information about sort order, see the chapter about the SORT procedure in *Base SAS Procedures Guide* and the discussion of BY-group processing in *SAS Language Reference: Concepts*.

REF='*category***' | FIRST | LAST**
specifies the reference category for the binary response model. Specifying one response category as the reference is the same as specifying the other response category as the event category. You can specify any of the following values:

'*category*'	specifies that observations whose value matches *category* (formatted, if a format is applied) are designated as the reference.
FIRST	designates the first ordered category as the reference.
LAST	designates the last ordered category as the reference.

By default, REF=LAST.

Spline Effect Options

Spline effects are specified by the SPLINE(*variable* / < *spline-options* >) or the SPLINE(*variable1 variable2* / < *spline-options* >) option. Each SPLINE term defines one spline effect. The boosting selection method can take both types of specifications. The shrinkage selection method can only take SPLINE(*variable* / < *spline-options* >). You can specify any number of SPLINE(*variables*) options.

Spline effect options control how each spline term forms basis expansions. For more information about the construction and estimation of spline terms, see the sections "Penalized B-Splines" on page 671 and "Natural Cubic Splines" on page 673.

You can specify the following *spline-options*:

DEGREE=*number*
specifies the degree of the spline transformation, where *number* must be a nonnegative integer. This option is not applicable when the shrinkage method is requested. By default, DEGREE=3.

DETAILS
produces a detailed spline specification information table.

DF=*number*
specifies the fixed degrees of freedom for the spline at each boosting iteration. This option is applicable only when the boosting selection method is requested. The SMOOTH= option is ignored when you specify the degrees of freedom. By default, DF=4 for univariate spline terms and DF=6 for bivariate spline terms.

DIFFORDER=*number*
specifies the order of the difference penalty for a B-spline. This option is applicable only when the boosting selection method is requested. By default, DIFFORDER=2.

KNOTS=*method*
specifies the *method* to use for supplying user-defined knot values for constructing basis expansions. You can specify the following *methods*:

EQUAL(*n*)
specifies the number of equally spaced interior knots to use for every variable in a spline term. Boundary knots are automatically added to the knot list for each variable.

LIST(*list-of-values*)
specifies a list of values as knots for the spline construction. For a multivariate spline term, the listed values are taken as multiple row vectors, where each vector has values that are ordered by specified variables. If the last row vector of knots contains fewer values than the number of variables, then the last row vector is ignored. For example, the following specification of a spline term produces two actual knot vectors ($\boldsymbol{k}_1$ and $\boldsymbol{k}_2$), displayed in Table 13.5, and the value 5 is ignored.

```
spline(x1 x2/knots=list(1 2 3 4 5))
```

Table 13.5 Knot Values for a Bivariate Spline with a Supplied List

	x1	x2
$\boldsymbol{k}_1$	1	2
$\boldsymbol{k}_2$	3	4

By default, the boosting method constructs basis expansions by using 20 evenly spaced interior knots for univariate spline terms, 10 evenly spaced interior knots for each variable in bivariate spline term, and evenly spaced boundary knots. By default, the shrinkage method constructs natural cubic spline basis expansions by using interior knots at unique data points and duplicated exterior knots. You can use the MAXKNOTS= option to specify the maximum number of knots if unique data points are used as interior knots.

LOWEREXKNOTS=*number*
specifies the lower exterior knots.

MAXKNOTS=*number*
specifies the maximum *number* of knots if data points are used to form knots. The boosting method does not support knot construction from unique data points. For the shrinkage method, if KNOTS=LIST(*list-of-values*) is not specified, PROC GAMSELECT forms knots from unique data points. If the number of unique data points is greater than *number*, a subset of size *number* is formed by random sampling from all unique data points. The *number* cannot exceed the largest integer that can be stored on the CAS server. By default, MAXKNOTS=20 if the shrinkage selection method is requested.

SMOOTH=*number*
specifies a fixed smoothing parameter. This option is applicable only when the boosting selection method is requested. This option is ignored when you specify the DF= option.

UPPEREXKNOTS=*number*
specifies the upper exterior knots.

WEIGHT1=*number*
specifies a sparsity penalty strength parameter. When you specify this option, a spline term might receive more or less sparsity penalty than other terms. This option is applicable only when the shrinkage selection method is requested. By default, WEIGHT1=1.

WEIGHT2=*number*
specifies a smoothness penalty strength parameter. When you specify this option, a spline term might receive more or less smoothness penalty than other terms. This option is applicable only when the shrinkage selection method is requested. By default, WEIGHT2=1.

Model Options

You can specify the following *model-options* in the MODEL statement after a slash (/):

ALLOBS
uses values of spline variables from all data roles for constructing the spline basis functions. By default, only observations that have the training data role are used to construct the spline basis functions.

DISPERSION=*number*

PHI=*number*
specifies a fixed dispersion parameter for distributions that have a dispersion parameter. The dispersion parameter that is used in all computations is fixed at *number*; it is not estimated.

DISTRIBUTION=*keyword*
specifies the response distribution for the model. The *keywords* and their associated distributions are shown in Table 13.6.

Table 13.6 Built-In Distribution Functions

DISTRIBUTION=	Distribution Function
BINARY | BERNOULLI	Binary
BINOMIAL	Binary or binomial
GAMMA	Gamma
INVGAUSS | IG | IGAUSSIAN	Inverse Gaussian
NORMAL | GAUSSIAN | GAUSS	Normal
POISSON | POI	Poisson

If you do not specify a link function in the LINK= option, a default link function is used. The default link function for each distribution is shown in Table 13.7. You can use any link function shown in Table 13.8 by specifying the LINK= option. Other commonly used link functions for each distribution are shown in Table 13.7.

Table 13.7 Default and Commonly Used Link Functions

DISTRIBUTION=	Default Link Function	Other Commonly Used Link Functions
BINARY | BERNOULLI	Logit	Probit, complementary log-log, log-log
BINOMIAL	Logit	Probit, complementary log-log, log-log
GAMMA	Log	Reciprocal
INVGAUSS | IG | IGAUSSIAN	Log	Reciprocal square
NORMAL | GAUSSIAN | GAUSS	Identity	Log
POISSON | POI	Log	

INITIALPHI=*number*

specifies a starting value for iterative maximum likelihood estimation of the dispersion parameter for distributions that have a dispersion parameter.

LINK=*keyword*

specifies the link function for the model. The *keywords* and the associated link functions are shown in Table 13.8. Default and commonly used link functions for the available distributions are shown in Table 13.7.

Table 13.8 Built-In Link Functions

LINK=	Link Function	$g(\mu) = \eta =$
CLOGLOG | CLL	Complementary log-log	$\log(-\log(1-\mu))$
IDENTITY | ID	Identity	μ
INVERSE | RECIP | RECIPROCAL	Reciprocal	$1/\mu$
INV2 | POWERMINUS2 | POWMINUS2	Reciprocal square	$1/\mu^2$
LOG	Logarithm	$\log(\mu)$
LOGIT	Logit	$\log(\mu/(1-\mu))$
LOGLOG	Log-log	$-\log(-\log(\mu))$
PROBIT | NORMIT	Probit	$\Phi^{-1}(\mu)$

$\Phi^{-1}(\cdot)$ denotes the quantile function of the standard normal distribution.

MAXPHI=*number*

specifies an upper bound for maximum likelihood estimation of the dispersion parameter for distributions that have a dispersion parameter.

MINPHI=*number*

specifies a lower bound for maximum likelihood estimation of the dispersion parameter for distributions that have a dispersion parameter.

OFFSET=*variable*
specifies a *variable* to use as an offset to the linear predictor. An offset plays the role of an effect whose coefficient is known to be 1. The offset variable cannot appear in the CLASS statement or elsewhere in the MODEL statement. Observations that have missing values for the offset variable are excluded from the analysis. For the boosting method, the offset variable is used to set the initial model estimate.

OUTPUT Statement

OUTPUT OUT=*libref.data-table*
< **COMPONENT** > < **COMPVARS=(***variables***)** > < **COPYVARS=(***variables***)** >
< *keyword* < **=***name* > >. . . < *keyword* < **=***name* > > **;**

The OUTPUT statement creates a data table that contains observationwise statistics that PROC GAMSELECT computes after fitting the model. In order to avoid data duplication for large data tables, the variables in the input data table are *not* included in the output data table unless you specify them in the COPYVARS= option.

The computation of the output statistics is based on the final parameter estimates. If the model fit does not converge, missing values are produced for the quantities that depend on the estimates.

You must specify the following option:

OUT=*libref.data-table*
names the output data table for PROC GAMSELECT to use. You must specify this option before any other options. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to where the data table is to be stored, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 645.

data-table specifies the name of the output data table.

You can also specify the following syntax elements:

COMPONENT
produces componentwise statistics for all spline terms if LINP is specified as a *keyword*.

COMPVAR | COMPUTEDVAR=*variable*

COMPVARS | COMPUTEDVARS=(*variables***)**
transfers one or more *variables* that are computed from programming statements to the output data table.

COPYVAR=*variable*

COPYVARS=(*variables***)**
transfers one or more *variables* from the input data table to the output data table.

keyword **<** **=***name* **>**
specifies a statistic to include in the output data table and optionally assigns a *name* to the variable. If you do not provide a *name*, PROC GAMSELECT assigns a default name based on the *keyword*.

You can specify the following *keywords* for adding statistics to the OUTPUT data table:

XBETA< **=***name* **>**

LINP< **=***name* **>**
requests the linear predictor $\eta = \mathbf{x}'\boldsymbol{\beta}$. For observations in which only the response variable is missing, values of the linear predictor are computed even though these observations do not affect the model fit. The default *name* is Xbeta.

PEARSON< **=***name* **>**

PEARS< **=***name* **>**

RESCHI< **=***name* **>**
computes the Pearson residual, $(y-\mu)/\sqrt{\mathrm{V}(\mu)}$, where μ is the estimate of the predicted response mean and $\mathrm{V}(\mu)$ is the response distribution variance function. The default *name* is Pearson.

PRED< **=***name* **>**

PREDICTED< **=***name* **>**

P< **=***name* **>**
computes predicted values for the response variable. For observations in which only the response variable is missing, the predicted values are computed even though these observations do not affect the model fit. The default *name* is Pred.

RESIDUAL< **=***name* **>**

RESID< **=***name* **>**

R< **=***name* **>**
computes the raw residual, $y - \mu$, where μ is the estimate of the predicted mean. The default *name* is Residual.

ROLE< **=***name* **>**
specifies the numeric variable that indicates the role played by each observation in fitting the model. The default *name* is _ROLE_. Table 13.9 shows how this variable is interpreted for each observation.

Table 13.9 Role Interpretation

Value	Observation Role
0	Not used
1	Training
2	Validation
3	Testing

If you do not partition the input data by specifying a PARTITION statement, then the role variable value is 1 for observations that are used in fitting the model and 0 for observations that have at least one missing or invalid value for the response, regressor, frequency, or weight variables.

PARTITION Statement

PARTITION *partition-option* **;**

The PARTITION statement specifies how observations in the input data set are logically partitioned into disjoint subsets for model training, validation, and testing. For more information, see the section "Using Validation and Test Data" on page 101 in Chapter 2, "Shared Concepts." Either you can designate a variable in the input data table and a set of formatted values of that variable to determine the role of each observation, or you can specify proportions to use for randomly assigning observations to each role.

You must specify exactly one of the following *partition-options*:

FRACTION(<TEST=*fraction***> <VALIDATE=***fraction***> <SEED=***number***>)**
randomly assigns specified proportions of the observations in the input data table to the roles. You specify the proportions for testing and validation by using the TEST= and VALIDATE= suboptions. If you specify both the TEST= and VALIDATE= suboptions, then the sum of the specified fractions must be less than 1 and the remaining fraction is the proportion of the observations assigned to the training role. The SEED= option specifies an integer that is used to start the pseudorandom number generator for random partitioning of data for training, testing, and validation. If you do not specify SEED=*number* or if *number* is less than or equal to 0, the seed is generated by reading the time of day from the computer's clock.

ROLE=*variable* **(<TEST='***value***'> <TRAIN='***value***'> <VALIDATE='***value***'>)**
ROLEVAR=*variable* **(<TEST='***value***'> <TRAIN='***value***'> <VALIDATE='***value***'>)**
names the *variable* in the input data table whose values are used to assign roles to each observation. This *variable* cannot also appear as an analysis variable in other statements or options. The TEST=, TRAIN=, and VALIDATE= suboptions specify the formatted values of this variable that are used to assign observation roles. If you do not specify the TRAIN= suboption, then all observations whose role is not determined by the TEST= or VALIDATE= suboption are assigned to the training role.

SELECTION Statement

SELECTION <**METHOD=***method* <**(***method-options***)**>> **;**

The SELECTION statement performs model selection by examining whether spline effects should be kept in the model.

You can specify the following *methods* in the SELECTION statement:

METHOD=*method* **<(***method-options***)>**
specifies the method to use to select the model. You can also specify *method-options* that apply to the specified method by enclosing them in parentheses after the *method*.

The following *methods* are available and are explained in detail in the section "Model Selection Methods" on page 674. By default, METHOD=BOOSTING.

BOOSTING specifies the boosting method.

SHRINKAGE specifies the shrinkage method.

Because of the intrinsic difference between the boosting method and the shrinkage method, the two selection methods have two different sets of options that enable you to control the selection process. You can specify the following *method-options* for the boosting method.

CHOOSE=CV | VALIDATE
specifies the criterion to use to select the final model. If you specify the STOPHORIZON= option, the criterion that you specify in the CHOOSE= option is also used to evaluate early termination of the selection process. If you omit the CHOOSE= option, the average square error for the training data is used to select the final model.

You can specify the following values:

CV | CROSSVALIDATION specifies k-fold cross validation of the average square error. You can use the INDEX= or KFOLD= option to specify the partition folds.

VALIDATE specifies the average square error for the validation data.

INDEX=*variable*
names the *variable* in the input data table whose values are used to assign observations to partition folds for cross validation. This option is applicable only if you specify the CHOOSE=CV option. The number of folds equals the number of unique levels of the *variable*.

KFOLD=*number*
specifies the *number* of partition folds in the random cross validation process. This option is applicable only if you specify the CHOOSE=CV option. The *number* must be greater than 2. By default, KFOLD=5.

LINPINMEM=*number*
specifies the maximum *number* of observations for which the linear predictor values are stored in memory when you are training the model. The *number* must be positive. By default, LINPINMEM=500000.

MAXITER=*number*
specifies the maximum number of iterations for the boosting method. By default, MAXITER=500.

STEPSIZE=*number*
LEARNINGRATE=*number*
specifies the step size to use for the boosting algorithm, where *number* must be between 0 and 1. By default, STEPSIZE=0.1.

STOPHORIZON=*number*
specifies the number of consecutive iterations in which the performance measured by the criterion that you specify in the CHOOSE= option must deteriorate in order to terminate the selection process. If you specify the STOPTOL= option, a relative change in the criterion is evaluated. By default, an absolute difference in this criterion is evaluated. If you omit the CHOOSE= option, the average square error for the training data is used.

STOPTOL=*number*

STOPTOLERANCE=*number*

specifies the *number* to use as the tolerance for evaluating the relative change in the CHOOSE= option. This option is applicable only when you specify the STOPHORIZON= option. The *number* must be nonnegative.

You can specify the following *method-options* for the shrinkage method.

ADMMABSEPS=*number*

specifies the absolute convergence criterion for the alternating direction method of multipliers (ADMM) algorithm that is used to compute the solution. By default, ADMMABSEPS=1E–6.

ADMMMAXITER=*number*

specifies the maximum number of iterations that the ADMM algorithm can take at each step of solving a reweighted additive model. By default, ADMMMAXITER=10000.

ADMMRELEPS=*number*

specifies the relative convergence criterion for the ADMM algorithm that is used to compute the solution. By default, ADMMRELEPS=1E–4.

ADMMRHO=*number*

specifies the tuning parameter to use in the ADMM algorithm to control the balance between the primal residual and the dual residual. By default, *number* is computed from the input data.

ADMMRHOADAPTIVE < =NO | YES >

specifies whether to use an adaptive mechanism to update the ADMM tuning parameter in the first several steps. By default, ADMMRHOADAPTIVE=YES.

DISTRIBUTEDSEARCH < =NO | YES >

specifies whether to use distributed mode to evaluate different regularization parameter candidates in a grid environment. This mode distributes training data to workers so that each worker has a full copy of the training data and performs model fitting independently. Using this option can eliminate the cost of communication over the grid and thus improve performance. However, distributed mode requires more memory space for each worker. This option is not applicable when you are in single-machine mode. By default, DISTRIBUTEDSEARCH=YES.

LAMBDA1=*number*

L1=*number*

sets a fixed nonnegative value to control the sparsity penalty.

LAMBDA2=*number*

L2=*number*

sets a fixed nonnegative value to control the smoothness penalty.

LAMBDA3=*number*

L3=*number*

sets a fixed nonnegative value to control the generalized ridge penalty.

MAXITER=*number*

specifies the maximum number of iterations that generalized additive model fitting can take by solving reweighted additive models at each iteration. By default, MAXITER=500.

MAXLAMBDA1=*number*

MAXL1=*number*

sets the maximum value to start with to search the optimal sparsity penalty parameter. By default, *number* is computed from the input data.

MAXLAMBDA2=*number*

MAXL2=*number*

sets the maximum value to start with to search the optimal smoothness penalty parameter. By default, *number* is computed from the input data.

MAXLAMBDA3=*number*

MAXL3=*number*

sets the maximum value to start with to search the optimal generalized ridge penalty parameter. By default, MAXLAMBDA3=0.

NUMLAMBDA1=*number*

NUML1=*number*

sets the total number of candidates to try in searching the optimal sparsity penalty parameter. By default, NUMLAMBDA1=20.

NUMLAMBDA2=*number*

NUML2=*number*

sets the total number of candidates to try in searching the optimal smoothness penalty parameter. By default, NUMLAMBDA2=10.

NUMLAMBDA3=*number*

NUML3=*number*

sets the total number of candidates to try in searching the optimal generalized ridge parameter. If you specify the MAXLAMBDA3= option but not the LAMBDA3= option, then by default NUMLAMBDA3=10.

RHOLAMBDA1=*number*

RHOL1=*number*

sets the scaling factor for the sparsity penalty parameter when computing a new candidate from its previous candidate attempt. For example, if the maximum sparsity penalty parameter is λ_1^*, then PROC GAMSELECT first computes the solution at λ_1^*, and in the following steps it computes solutions at $\lambda_1^*\rho_1$, $\lambda_1^*\rho_1^2$, and so on. By default, RHOLAMBDA1=0.8.

RHOLAMBDA2=*number*

RHOL2=*number*

sets the scaling factor for the smoothness penalty parameter when computing a new candidate from its previous candidate attempt. For example, if the maximum smoothness penalty parameter is λ_2^*, then PROC GAMSELECT first computes the solution at λ_2^*, and in the following steps it computes solutions at $\lambda_2^*\rho_2$, $\lambda_2^*\rho_2^2$, and so on. By default, RHOLAMBDA2=0.5.

RHOLAMBDA3=*number*

RHOL3=*number*

sets the scaling factor for the generalized ridge penalty parameter when computing a new candidate from its previous candidate attempt. For example, if the maximum generalized ridge penalty parameter is λ_3^*, then PROC GAMSELECT first computes the solution at λ_3^*, and in the following steps it computes solutions at $\lambda_3^*\rho_3$, $\lambda_3^*\rho_3^2$, and so on. If you specify the MAXLAMBDA3= option but not the LAMBDA3= option, then by default RHOLAMBDA3=0.5.

STORE Statement

STORE < **OUT=** >*libref.data-table* **;**

The STORE statement saves the context and results of the statistical analysis. The resulting analytic store has a binary file format that cannot be modified. An example of analytic store technology is to perform a time-consuming analysis and to store its results by using the STORE statement. Later you can perform specific statistical analysis tasks based on the saved results of the previous analysis, without having to fit the model again. The contents of the analytic store can be processed by the ASTORE procedure, the **`gamScore`** action in the **`gam`** action set, and the **`score`** action in the **`aStore`** action set.

libref.data-table is a two-level name, where *libref* refers to the library, and *data-table* specifies the name of the output data table. For more information about this two-level name, see the section "Using CAS Sessions and CAS Engine Librefs" on page 645.

WEIGHT Statement

WEIGHT *variable* **;**

The *variable* in the WEIGHT statement is used as a weight to perform a weighted analysis of the data. Observations that have nonpositive or missing weights are not included in the analysis. The values of the WEIGHT variable can be nonintegral and are not truncated. When you omit the WEIGHT statement, all observations that are used in the analysis are assigned a weight of 1.

Programming Statements

A number of programming statements are available in the GAMSELECT procedure that enable you to create computed variables on the fly. For example, you can compute model effects, weights, frequency, and other variables. This section lists these programming statements and also documents the differences between them and programming statements available in the SAS DATA step. The syntax of programming statements that is used in PROC GAMSELECT is identical to that used in the NLMIXED procedure (see Chapter 89, "The NLMIXED Procedure" (*SAS/STAT User's Guide*) and the MODEL procedure (see the *SAS/ETS User's Guide*). Most of the programming statements available in the DATA step are also available in PROC GAMSELECT. See *SAS DATA Step Statements: Reference* for a description of SAS programming statements.

You can specify the following programming statements:

ARRAY *arrayname* < **[** *dimensions* **]** > < **$** > < *variables-and-constants* >**;**
CALL *name* < **(***expression* < **,** *expression . . .* > **)** >**;**
DO < *variable* **=** *expression* < **TO** *expression* > < **BY** *expression* > >
< **,** *expression* < **TO** *expression* > < **BY** *expression* > > . . .
< **WHILE** *expression* > < **UNTIL** *expression* >**;**
END;
GOTO *statement-label***;**
IF *expression***;**
IF *expression* **THEN** *program-statement***;**
ELSE *program-statement***;**
variable **=** *expression***;**
variable **+** *expression***;**
LINK *statement-label***;**
RETURN;
SELECT < **(***expression***)** >**;**
STOP;
SUBSTR(*variable***,** *index***,** *length***) =** *expression***;**
WHEN (*expression***)** *program-statement***;**
OTHERWISE *program-statement***;**

For the most part, the SAS programming statements available in PROC GAMSELECT work the same as they do in the SAS DATA step, as documented in *SAS Programmers Guide: Essentials*. However, there are several differences:

- The DO statement does not allow a character index variable. Thus

```
do i = 1,2,3;
```

  is supported. However, the following statement is not supported:

```
do i = 'A','B','C';
```

- The LAG function is not supported.
- The WHEN and OTHERWISE statements enable you to specify more than one target statement. That is, DO/END groups are not necessary for multiple WHEN statements. For example, the following syntax is valid:

```
select;
   when (exp1) stmt1;
               stmt2;
   when (exp2) stmt3;
               stmt4;
end;
```

The LINK statement is used in a program to jump immediately to the label *statement_label* and to continue program execution at that point.

When coding your programming statements, you should avoid defining variables that begin with an underscore (_), because they might conflict with internal variables that PROC GAMSELECT creates. You should avoid operations that would directly modify the variables that appear in the input table, because such operations are not supported for the input data.

You cannot use the programming statements to create a computed variable and then add it to the CLASS statement so that it is used as a classification variable. If you have already specified a variable in the CLASS statement, you should also avoid using it in the programming statements to compute variables with the expectation that the computation would use its levelization results.

Details: GAMSELECT Procedure

Missing Values

Any observation that has missing values for the response, frequency, weight, offset, or explanatory variables is excluded from the analysis; however, missing values are valid for response and explanatory variables if you specify the MISSING option in the CLASS statement. Observations that have a nonpositive weight or a frequency less than 1 are also excluded. For Poisson distribution, observations that have a negative response value are excluded. For gamma and inverse Gaussian distributions, observations that have a nonpositive response value are excluded.

The estimated linear predictor and the fitted means are not computed for any observation that has missing offset or explanatory variables. However, if only the response value is missing, the linear predictor and the fitted means can be computed and output to a data table by using the OUTPUT statement.

Penalized B-Splines

You can use penalized B-splines (Eilers and Marx 1996) to fit a smooth curve, $f_j(\mathbf{x}_j)$, through a scatter plot of points (u_i, x_{ij}), $i = 1, \ldots, n$. The penalized B-spline fit applies a difference penalty to coefficients for adjacent elements of a B-spline basis expansion $\mathbf{B}_j$ of $\mathbf{x}_j$. For more information about B-spline basis expansions, see the section "B-Spline Basis" on page 30 in Chapter 2, "Shared Concepts" and also see De Boor (1978).

Let $\boldsymbol{\beta}$ denote the coefficients for the basis expansion $\mathbf{B}$. The first-order difference for the ℓth element of $\boldsymbol{\beta}$ is $\Delta\beta_\ell = \beta_\ell - \beta_{\ell-1}$. Higher-order differences are defined recursively by $\Delta^o\beta_\ell = \Delta\left(\Delta^{o-1}\beta_\ell\right)$. For a B-spline expansion with degree d and k interior knots, the penalized B-spline fit is obtained by solving

$$\min_{\boldsymbol{\beta}} \frac{1}{2}\sum_{i=1}^{n} w_i(u_i - \mathbf{B}_{ij}\boldsymbol{\beta})^2 + \frac{\lambda}{2}\sum_{\ell=o+1}^{d+k+1}\left(\Delta^o\beta_\ell\right)^2$$

where λ is a nonnegative smoothing parameter and w_i are observation weights.

As described in Eilers and Marx (1996), a second-order difference penalty is related to a penalty on the second derivative of the estimated function. Compared to penalties on the derivatives of a function, the difference

penalty leads to a simpler minimization problem by omitting terms that involve overlapping elements of the B-spline basis functions.

The difference penalty can be written in terms of a $(d + 1 + k - o) \times (d + 1 + k)$ matrix $\mathbf{D}$. The nonzero elements of $\mathbf{D}$, order 1, are (1 –1); $\mathbf{D}$, order 2 are (1 –2 1); $\mathbf{D}$, order 3 (the default) are (1 –3 3 –1); $\mathbf{D}$, order 4 are (1 –4 6 –4 1); and so on. The nonzero elements for each order are made from the nonzero elements from the preceding order by subtraction: $\mathbf{d}'_{i+1} = (\mathbf{d}'_i \;\; 0) - (0 \;\; \mathbf{d}'_i)$. For example, the nonzero elements for the order 3 penalty are obtained as follows:

$$
\begin{array}{rcccc}
 & 1 & -1 & 0 & \\
- & 0 & 1 & -1 & \\
\hline
 & 1 & -2 & 1 & 0 \\
- & 0 & 1 & -2 & 1 \\
\hline
 & 1 & -3 & 3 & -1
\end{array}
$$

Within an order, the first nonzero element of row i is in column i—that is, each row of $\mathbf{D}$ is made from the preceding row by shifting the nonzero elements to the right by one position. For example, when $k = 4$ knots, order $o = 3$, and degree $d = 3$, $\mathbf{D}$ is the following matrix:

$$
\begin{bmatrix}
1 & -3 & 3 & -1 & 0 & 0 & 0 & 0 \\
0 & 1 & -3 & 3 & -1 & 0 & 0 & 0 \\
0 & 0 & 1 & -3 & 3 & -1 & 0 & 0 \\
0 & 0 & 0 & 1 & -3 & 3 & -1 & 0 \\
0 & 0 & 0 & 0 & 1 & -3 & 3 & -1
\end{bmatrix}
$$

The penalized B-spline transformation $\hat{\mathbf{u}}$ of $\mathbf{u}$ is a linear smoother $\hat{\mathbf{u}} = \mathbf{H_j u}$, where the hat matrix $\mathbf{H_j}$ is defined as

$$\mathbf{H_j} = \mathbf{B}_j \left(\mathbf{B}_j^T \mathbf{W} \mathbf{B}_j + \lambda \mathbf{D}^T \mathbf{D} \right)^{-1} \mathbf{B}_j^T \mathbf{W}$$

where $\mathbf{W}$ is a diagonal matrix of the observation weights w_i. The degrees of freedom for the penalized B-spline equals the trace of $\mathbf{H_j}$.

You can extend penalized B-splines to bivariate spline terms by using the approach described in Marx and Eilers (2005). A two-dimensional surface is created by taking the outer product of the B-spline basis expansions for each variable. The boosting method uses the same degree d and number of interior knots k for each B-spline basis expansion. A vectorized representation of the outer product leads to a spline expansion $\mathbf{B}_{ij}$ of dimension $1 \times (d + k + 1)^2$ for $\mathbf{x}_{ij}$.

For the bivariate spline term, the penalty matrix $\mathbf{P}$ is of the form

$$\mathbf{P} = \mathbf{I} \otimes \mathbf{D}^T \mathbf{D} + \mathbf{D}^T \mathbf{D} \otimes \mathbf{I}$$

where $\otimes$ denotes the Kronecker product, $\mathbf{I}$ denotes the identity matrix, and $\mathbf{D}$ is the $(d+1+k-o) \times (d+1+k)$ difference matrix for order o. The matrices $\mathbf{I} \otimes \mathbf{D}^T \mathbf{D}$ and $\mathbf{D}^T \mathbf{D} \otimes \mathbf{I}$ apply the difference penalty to the coefficients $\boldsymbol{\beta}$ that correspond to the rows and columns of the two-dimensional representation of the spline basis surface. The boosting method assumes that the same difference order o is used in both components of the penalty matrix $\mathbf{P}$.

The penalized B-spline fit for a bivariate spline with nonnegative smoothing parameter λ is also a linear smoother with hat matrix $\mathbf{H_j}$ defined as

$$\mathbf{H_j} = \mathbf{B}_j \left(\mathbf{B}_j^T \mathbf{W} \mathbf{B}_j + \lambda \mathbf{P} \right)^{-1} \mathbf{B}_j^T \mathbf{W}$$

Natural Cubic Splines

A natural cubic spline is the function $f(\cdot)$ that minimizes the penalized residual sum of squares

$$\mathbf{RSS}(f,\lambda) = \sum_{i=1}^{n}(y_i - f(x_i))^2 + \lambda \int (f''(t))^2 \mathrm{d}t$$

where λ is the smoothing parameter that controls the trade-off between a model's goodness of fit and its complexity. The unknown smooth function $f(x)$ is constructed by using cubic polynomials that are smoothly connected at all unique data points. Additional constraints at the boundaries enforce linear functions beyond the boundary knots. Natural cubic splines can be represented by both the truncated power function basis and the B-spline basis. For more information, see the section "Natural Cubic Spline Basis" on page 32 in Chapter 2, "Shared Concepts."

For the shrinkage selection method, the GAMSELECT procedure uses a similar spline representation to approximate the unknown smooth function *f*, but with fewer interior knots, so that its computation is more efficient for models that have multiple spline terms. The results are still close to natural cubic splines. This technique is introduced in O'Sullivan (1986). Because of this resemblance, the technique is referred to as natural cubic splines hereafter in this chapter.

On the basis of the B-spline representation, Wand and Ormerod (2008) derive the explicit formula for the penalty matrix

$$\mathbf{\Omega} = \mathbf{B}_L^T \mathbf{W} \mathbf{B}_L$$

where $\mathbf{B}_L$ is the linear B-spline basis based on the same set of knots and evaluated at an augmented set of points constructed from the used knots, and $\mathbf{W}$ is a diagonal matrix with diagonal values that is constructed from the same augmented set of points. For more information, see Wand and Ormerod (2008).

Generalized Additive Models

All probability distributions that the GAMSELECT procedure fits are members of the exponential family of distributions. For the specification of an exponential family, see the section "Exponential Family Distributions" on page 747 in Chapter 14, "GENSELECT Procedure." The GAMSELECT procedure supports the binary, binomial, gamma, inverse Gaussian, normal (Gaussian), and Poisson distributions.

For forms of log-likelihood functions for each of the probability distributions, see the section "Log-Likelihood Functions" on page 754 in Chapter 14, "GENSELECT Procedure."

Generalized additive models (GAMs) are extensions of generalized linear models (Nelder and Wedderburn 1972). For each observation that has a response Y_i and a row vector of the model matrix $\mathbf{x}_i$, both GAMs and generalized linear models assume the model additivity,

$$g(\mu_i) = f_1(\mathbf{x}_{i1}) + \cdots + f_p(\mathbf{x}_{ip})$$

where $\mu_i = E(Y_i)$ and Y_i is independently distributed in some exponential family. Generalized linear models further assume model linearity by $f_j(\mathbf{x}_{ij}) = \mathbf{x}_{ij}\beta_j$ for $j = 1, \ldots, p$. GAMs relax the linearity assumption by allowing spline functions.

For the boosting method, PROC GAMSELECT fits the model by performing a series of boosting iterations. At each iteration, a separate fit is obtained for each effect and one effect is selected for an update. The final model is composed of the effects that are selected across the boosting iterations.

For the shrinkage method, PROC GAMSELECT fits models by estimating all parameters simultaneously. Based on the spline basis construction mechanism, the model is unidentifiable in general because each spline term contains an intercept. To address this, PROC GAMSELECT transforms each spline design matrix into $\tilde{\mathbf{B}}_j$ to satisfy $\mathbf{1}'\mathbf{B}_j\boldsymbol{\beta}_j = 0$. The penalty matrix that corresponds to each spline term is also transformed. For simplicity, hereafter in the chapter the transformed spline design matrix is referred to as $\mathbf{B}_j$, and the transformed penalty matrix is referred to as $\boldsymbol{\Omega}_j$. For more information, see the section "Generalized Additive Models" on page 610 in Chapter 12, "GAMMOD Procedure."

Model Selection Methods

The model selection methods are specified using the METHOD= option in the SELECTION statement. The following methods are available. By default, the boosting method is used.

Boosting

The boosting selection method uses a componentwise implementation of a generic functional gradient descent, or boosting, algorithm (Friedman, Hastie, and Tibshirani 2000; Friedman 2001; Bühlmann and Hothorn 2007). Component functions f_j are defined for each parametric effect and spline term in the MODEL statement. Spline terms can have at most two variables for the boosting selection method.

Table 13.10 defines some notation that is used here to describe the componentwise boosting algorithm that PROC GAMSELECT implements.

Table 13.10 Boosting Notation

Notation	Meaning
ℓ	Log likelihood
$\hat{f}^m$	Estimate for f at mth boosting iteration
$\hat{f}_j^m$	Estimate for jth component function f_j at mth boosting iteration
$\mathbf{x}_j$	Predictors for the jth component function f_j
γ_j^m	Fit for jth component at mth boosting iteration
ν	Step size for the boosting algorithm
m_{end}	Number of boosting iterations executed
M	Maximum number of boosting iterations

The componentwise boosting algorithm is as follows:

1. Initialize $\hat{f}^0$ and $m = 1$.
2. Repeat the following steps until the maximum number of iterations M has been reached or an early stopping criterion is met:
   a) Compute $u_i = -\frac{\partial}{\partial f}\ell(y_i, f)|_{\hat{f}^{m-1}(\mathbf{x}_i)}, i = 1\ldots, n.$

b) For each component function f_j, $j = 1, \ldots, p$, obtain an estimated fit $\hat{\gamma}_j^m$ for the data $(u_i, \mathbf{x}_{ij})$, $i = 1 \ldots, n$.

c) Select the component $j^* = \underset{j=1,\ldots,p}{\operatorname{argmin}} \sum_{i=1}^{n} w_i \left(u_i - \hat{\gamma}_j^m(\mathbf{x}_{ij})\right)^2$.

d) Update the component functions by

$$\hat{f}_j^m = \begin{cases} \hat{f}_j^{m-1} + \nu\gamma_j^m & \text{if } j = j^* \\ \hat{f}_j^{m-1} & \text{if } j \neq j^* \end{cases}$$

e) Set $m = m + 1$ and check the stopping criterion.

3. Choose an iteration $m^* \leq m_{\text{end}}$ and set $\hat{f} = \hat{f}^{m^*}$.

The initial function $\hat{f}^0$ is set to the value of the offset variable that you specify in the OFFSET= option or by fitting an intercept-only model to the response y if no offset variable is specified. You can use the MAXITER= option in the SELECTION statement to specify the maximum number of boosting iterations. By default, MAXITER=500.

For component functions f_j that correspond to parametric effects, the functions γ_j^m are linear models that are estimated by solving

$$\min_{\boldsymbol{\beta}_j} \sum_{i=1}^{n} w_i \left(u_i - \mathbf{x}_{ij}\boldsymbol{\beta}_j\right)^2$$

Each linear model $\gamma_j^m = \mathbf{x}_{ij}\boldsymbol{\beta}_j$ includes an intercept term.

For component functions f_j that correspond to spline terms, the functions γ_j^m are estimated using penalized B-splines. For more information about penalized B-splines, see the section "Penalized B-Splines" on page 671.

By default, the penalized B-splines are of degree 3, and a difference penalty of order 2 is used. The smoothing parameter values are selected by default, so a univariate spline term has 4 degrees of freedom and a bivariate spline has 6 degrees of freedom. For more information about options that enable you to control how each spline term is constructed, see the section "Spline Effect Options" on page 659. Note that the degrees of freedom are defined for the penalized B-spline γ_j^m fit at each boosting iteration m. The degrees of freedom do not apply to the component functions f_j^m.

If you specify a value in the STOPHORIZON=*number* option in the SELECTION statement, then PROC GAMSELECT uses the criterion specified in the CHOOSE= option in the same statement to evaluate early termination of the selection process. Let CR_m denote the value of the criterion at iteration m. The selection process terminates early if $\mathrm{CR}_{m-1} < \mathrm{CR}_m$ for *number* consecutive iterations. If you specify the STOPTOL=*number* option in the SELECTION statement, a relative change in the criterion is evaluated and the selection process terminates early if

$$\left|\frac{\mathrm{CR}_{m-1} - \mathrm{CR}_m}{\mathrm{CR}_{m-1}}\right| < \tau$$

for *number* consecutive iterations, where $\tau =$*number*. By default, the average square error for the training data is used as the criterion.

When the boosting iterations finish, there are $m_{\text{end}}+1$ models $\hat{f}^m$. If you specify a criterion in the CHOOSE= option, PROC GAMSELECT selects the model $\hat{f} = \hat{f}^{m^*}$ that minimizes the criterion. By default, the procedure returns the model that minimizes the average square error on the training data. The effects in the final model are effects that were selected for at least one boosting iteration $m < m^*$.

Shrinkage

This section discusses the shrinkage selection method.

Sparsity-inducing norms have been explored by many authors as a technique for selecting generalized additive models. Because nonlinear terms in additive models are typically represented by splines and thus are naturally grouped, the L_2 norm used in group LASSO (Yuan and Lin 2006) and its generalizations are often used to perform additive model selection. For different approaches, see Lin and Zhang (2006), Cantoni, Flemming, and Ronchetti (2006), Ravikumar et al. (2009), Meier, Van de Geer, and Bühlmann (2009), Marra and Wood (2011), Raskutti, Wainwright, and Yu (2012), Suzuki and Sugiyama (2013), and Chouldechova and Hastie (2015). For a review of this field, see Amato, Antoniadis, and De Feis (2016).

Based on ideas in Meier, Van de Geer, and Bühlmann (2009), Suzuki and Sugiyama (2013), and Amato, Antoniadis, and De Feis (2016), the shrinkage method that is used in the GAMSELECT procedure performs a grid search of tuning parameters $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \lambda_3)$ and evaluates different models on the basis of their fits, given any fixed $\boldsymbol{\lambda}$ value. For each fixed $\boldsymbol{\lambda}$, PROC GAMSELECT fits the model by minimizing the penalized negative log-likelihood function

$$-\frac{\ell(\boldsymbol{\beta})\phi_d}{n} + \lambda_1 \sum_j \phi_j \sqrt{\boldsymbol{\beta}_j' \mathbf{K}_j \boldsymbol{\beta}_j} + \lambda_2 \sum_j \psi_j \sqrt{\boldsymbol{\beta}_j' \boldsymbol{\Omega}_j \boldsymbol{\beta}_j} + \lambda_3 \sum_j \boldsymbol{\beta}_j' \boldsymbol{\Omega}_j \boldsymbol{\beta}_j$$

where $\mathbf{K}_j = \mathbf{B}_j' \mathbf{B}_j / n$, and $\boldsymbol{\Omega}_j = \int (f_j'')^2 \mathrm{d}x$. The two terms $\lambda_1 \sum_j \phi_j \sqrt{\boldsymbol{\beta}_j' \mathbf{K}_j \boldsymbol{\beta}_j}$ and $\lambda_2 \sum_j \psi_j \sqrt{\boldsymbol{\beta}_j' \boldsymbol{\Omega}_j \boldsymbol{\beta}_j}$ are sparsity-inducing norms, which are generalizations of the L_2 norm. ϕ_j and ψ_j are two sets of control parameters that you can use to enforce stronger or weaker penalties for each individual spline term. ϕ_d is the dispersion parameter for the appropriate distribution in the exponential family. Because ϕ_d cancels out the denominator in the likelihood function, the optimization can focus on regression parameters only.

The first term can force $\boldsymbol{\beta}_j = \mathbf{0}$ if λ_1 is sufficiently large, and the second term can force $\boldsymbol{\beta}_j$ to be equivalent to a parametric solution ($\boldsymbol{\beta}_j' \boldsymbol{\Omega}_j \boldsymbol{\beta}_j = \mathbf{0}$) if λ_2 is sufficiently large. The third term, $\lambda_3 \sum_j \boldsymbol{\beta}_j' \boldsymbol{\Omega}_j \boldsymbol{\beta}_j$, serves as an optional penalty (similar to ridge regression) to make the solution smoother.

The solution for a fixed set of tuning parameters can be found by iteratively minimizing penalized weighted additive models that are based on pseudo-response z_i and pseudo-weights w_i. This is similar to the performance iteration algorithm used in the GAMMOD procedure. For more information, see "Performance Iteration" on page 614 in Chapter 12, "GAMMOD Procedure." The objective function for the minimization problem at each performance iteration is

$$\begin{aligned} O_{\boldsymbol{\lambda}}(\boldsymbol{\beta}|z, \mathbf{W}) = \frac{1}{2n}(z - \sum_j \mathbf{B}_j \boldsymbol{\beta}_j)' \mathbf{W} (z - \sum_j \mathbf{B}_j \boldsymbol{\beta}_j) &+ \lambda_1 \sum_j \phi_j \sqrt{\boldsymbol{\beta}_j' \mathbf{K}_j \boldsymbol{\beta}_j} \\ &+ \lambda_2 \sum_j \psi_j \sqrt{\boldsymbol{\beta}_j' \boldsymbol{\Omega}_j \boldsymbol{\beta}_j} \\ &+ \lambda_3 \sum_j \boldsymbol{\beta}_j' \boldsymbol{\Omega}_j \boldsymbol{\beta}_j \end{aligned}$$

To solve the optimization problem that involves two nonsmooth convex penalties operating on the same parameter vector, PROC GAMSELECT uses the alternating direction method of multipliers (ADMM) (Boyd et al. 2011) to break it into two subproblems.

Note that the objective function that is controlled by three tuning parameters $(\lambda_1, \lambda_2, \lambda_3)$ provides a general framework for performing additive model selection. Special cases of the objective correspond to different

model selection approaches. For example, $(\lambda_1, 0, 0)$ is close to Ravikumar et al. (2009), $(\lambda_1, \lambda_2, 0)$ is close to Koltchinskii and Yuan (2010), and $(0, \lambda_2, \lambda_3)$ is close to Tomioka and Suzuki (2009). The shrinkage selection method in PROC GAMSELECT enables you to specify a fixed value for any of the three tuning parameters while performing a grid search on the rest.

The shrinkage selection algorithm is as follows:

1. Pick a candidate $\boldsymbol{\lambda}$ value and perform the following steps:
   a) Initialize $f_j^{(0)} = 0$ for $j = 1, \ldots, p$.
   b) Repeat the following steps until the maximum differences in parameters over all spline terms have been reached or the maximum number of iterations has been reached.
      i. Recompute a pseudo-response $\boldsymbol{z}$ and pseudo-weights $\mathbf{W}$ on the basis of previous fits.
      ii. Use the corresponding partial residual to fit the parametric part, including the intercept.
      iii. Use the corresponding partial residual to optimize $\boldsymbol{O}_{\boldsymbol{\lambda}}(\boldsymbol{\beta}|\boldsymbol{z}_r, \mathbf{W})$.
   c) Obtain an estimate for the dispersion parameter if necessary.
2. Evaluate the model fit from the validation data or the training data. Update the selection if a better model is found.

Displayed Output

The following sections describe the output that PROC GAMSELECT produces. The output is organized into various tables, which are discussed in order of their appearance.

Model Information

The "Model Information" table displays basic information about the model, such as the response variable, the frequency variable, the link function, and the model category that the PROC GAMSELECT selected on the basis of your input and options. The "Model Information" table also displays the distribution of the data that is assumed by the procedure. The selection method and random number seed are also displayed.

Class Level Information

The "Class Level Information" table lists the levels of every variable that is specified in the CLASS statement. You should check this information to make sure that it is correct. You can adjust the order of the CLASS variable levels by specifying the ORDER= option in the CLASS statement. You can suppress the "Class Level Information" table completely or partially by specifying the NOCLPRINT= option in the PROC GAMSELECT statement.

If the classification variables use reference parameterization, the "Class Level Information" table also displays the reference value for each variable.

Number of Observations

The "Number of Observations" table displays the number of observations that are read from the input data table and the number of observations that are used in the analysis. If a FREQ statement is present, the sum of the frequencies that are read and used is displayed. If the events/trials syntax is used, the number of events and trials is also displayed.

Response Profile

The "Response Profile" table displays the ordered values from which PROC GAMSELECT determines the probability that is modeled as an event in binary models. For each response category level, the frequency that is used in the analysis is reported. You can affect the ordering of the response values by specifying *response-options* in the MODEL statement. For binary models, the note that follows this table indicates which outcome is modeled as the event in binary models and which value serves as the reference category.

The "Response Profile" table is not produced for binomial (events/trials) data. You can find information about the number of events and trials in the "Number of Observations" table.

Specifications for Spline(*spline-variables*)

The "Specifications for Spline(*spline-variables*)" table displays basic information (such as the number of variables, specified degrees of freedom, boundary knots and interior knots, and so on) about how to construct a spline term that PROC GAMSELECT uses to construct basis expansions. The procedure generates this table only when you specify the DETAILS option in the MODEL statement for a spline term.

Convergence Status

The convergence status table is a small ODS table that precedes the "Iteration Summary" table in the default output. In the listing, this table appears as a message that indicates whether the selected model that corresponds to the optimal regularization parameter converged. If the selected model fails to converge, the message indicates the reason for the failure. If you save the convergence status table to an output data table, a numeric Status variable is added that enables you to programmatically assess convergence. The values of the Status variable have the following meanings:

0 The selected model converged at the selected regularization parameters. The fitting of all other regularization parameters also succeeded.

1 The selected model converged at the selected regularization parameters. The fitting of some regularization parameters did not succeed.

2 Convergence was not achieved within the specified number of iterations. To modify the limit, see the MAXITER= option in the SELECTION statement.

3 The selected model did not converge.

PROC GAMSELECT generates this table only when you specify the shrinkage selection method.

Iteration Summary

For the boosting selection method, the "Iteration Summary" table displays the number of boosting iterations executed, the iteration that is selected by the criterion specified in the CHOOSE= option as the final model, and the number of effects in the final model.

For the shrinkage selection method, this table displays the number of penalty parameter trials, the number of reweighted iterations for the selected model, and the number of selected effects.

Shrinkage Regularization Parameter Summary

The "Shrinkage Regularization Parameter Summary" table displays the upper bounds of regularization parameters that are determined from the input data or specified by you, and the chosen parameters after PROC GAMSELECT performs the selection. The procedure generates this table only when you specify the shrinkage selection method.

Selected Effects

For the boosting selection method, the "Selected Effects" table displays the effects that are included in the final model, the first iteration at which an effect entered the model, and the number of iterations at which the effect was selected.

For the shrinkage selection method, this table displays which effects or spline components were included in the final model.

Fit Statistics

The "Fit Statistics" table displays the average square errors (ASEs) that are computed on the basis of the final selected model. If you specify a PARTITION statement, the table displays the values for each role along with statistics for comparing the training, validation, and testing results. If you use the boosting selection method and specify cross validation as the model selection criterion, this table also displays the cross validation ASEs for the selected model.

Parameter Estimates

The "Parameter Estimates" table displays the regression parameter estimates.

Iteration Details

For the boosting selection method, the "Iteration Details" table displays the effect that is selected at each iteration and the fit statistic values at each iteration. The table also includes columns for the coefficients of the effect selected at each iteration, scaled by the step size; these columns are not displayed by default but are accessible if you save the table as an output data set by specifying the ODS OUTPUT statement or if you save the table as an output data table by using the DISPLAYOUT statement.

Spline Regularization

The "Spline Regularization" table displays the sparsity penalty, smoothness penalty, and generalized ridge penalty for the selected spline terms in the final model. PROC GAMSELECT generates this table only when you specify the shrinkage selection method.

Timing

The "Timing" table displays the amount of time (in seconds) that PROC GAMSELECT required to perform different tasks in the analysis.

OutputCasTables Table

The OutputCasTables table is a special table that has information about each CAS table that is created during a CAS action execution. The information for each CAS table consists of the CAS table name, the caslib in which the table resides, and the number of columns and rows in the CAS table. Because this table is not a typical ODS table that contains analytical results, you cannot include it in the *table-spec-list* in the DISPLAYOUT statement.

ODS Table Names

Each table that the GAMSELECT procedure creates has a name that is associated with it. You must use this name to refer to the table when you use ODS statements. Table 13.11 lists the name of each table and a short description of its contents.

Table 13.11 ODS Tables Produced by PROC GAMSELECT

Table Name	Description	Statement	Option
ClassInfo	Level information from the CLASS statement	CLASS	
ConvergenceStatus	Status of the regularization parameter search at the conclusion of the optimization	SELECTION	METHOD=SHRINKAGE
FitStatistics	Fit statistics	Default output	
IterationDetails	Detailed information about iterations in the selection process	PROC GAMSELECT	ITDETAILS
IterationSummary	Summary information about iterations in selection methods	Default output	
ModelInfo	Information about the modeling environment	Default output	
NObs	Number of observations read and used, and number of events and trials, if applicable	Default output	
OutputCasTables	See the section "OutputCasTables Table" on page 680	OUTPUT DISPLAYOUT	OUT=

Table 13.11 *continued*

Table Name	Description	Statement	Option
ParameterEstimates	Solutions for the parameter estimates that are associated with parametric effects in the MODEL statement	Default output	
Regularization	Information about range and final choice of regularization parameters	SELECTION	METHOD=SHRINKAGE
ResponseProfile	Response categories and the category that is modeled in models for binary data	Default output	
SelectedEffects	List of effects and spline terms selected for the model	SELECTION	
SplineDetails	Information about spline construction and smoothing parameter search	MODEL	SPLINE(*variables* / DETAILS)
SplineRegularization	List of the spline regularization penalties for the selected model	SELECTION	METHOD=SHRINKAGE
Timing	Absolute and relative times for tasks performed by the procedure	Default output	

ODS Graphics

Statistical procedures use ODS Graphics to create graphs as part of their output. ODS Graphics is described in detail in the "Statistical Graphics Using ODS" chapter in *SAS/STAT User's Guide*.

Before you create graphs, ODS Graphics must be enabled (for example, by using the ODS GRAPHICS ON statement). For more information about enabling and disabling ODS Graphics, see the section "Enabling and Disabling ODS Graphics" in that chapter.

The overall appearance of graphs is controlled by ODS styles. Styles and other aspects of using ODS Graphics are discussed in the section "A Primer on ODS Statistical Graphics" in that chapter.

When ODS Graphics is enabled, the GAMSELECT procedure by default produces plots of the partial predictions for each spline term in the model. Use the PLOTS option in the PROC GAMSELECT statement to control aspects of these plots.

PROC GAMSELECT assigns a name to each graph that it creates by using ODS. You can use these names to refer to the graphs when using ODS. The names are listed in Table 13.12.

Table 13.12 Graphs Produced by PROC GAMSELECT

ODS Graph Name	Plot Description	PLOTS= Option
SmoothingComponentPanel	Panel of multiple prediction curves	Default COMPONENTS
SmoothingComponentPlot	Unpacked prediction curves	PLOTS(UNPACK) PLOTS=COMPONENTS(UNPACK)

By default, prediction plots for each spline component are displayed in panels that contain at most six plots. If you specify more than six smoothing components, multiple panels are used. To display these plots individually, use the PLOTS(UNPACK) option in the PROC GAMSELECT statement.

Examples: GAMSELECT Procedure

The examples in this chapter are available in the GitHub repository located at `https://github.com/sascommunities/doc-supplement-casstat`.

Example 13.1: Testing Model Fit with Partitioned Data

(View the complete code for this example (gamselex1.sas) in the example repository.)

The Sashelp.JunkMail data set comes from a study that classifies whether an email is junk email (coded as 1) or not (coded as 0). The data were collected by Hewlett-Packard Labs and donated by George Forman. The data table, which is specified in the following DATA step, contains 4,601 observations, with 2 binary variables and 57 continuous explanatory variables. The response variable, Class, is a binary indicator of whether an email is considered spam or not. The partitioning variable, Test, is a binary indicator that is used to divide the data into training and testing parts. The 57 explanatory variables are continuous variables that represent frequencies of some common words and characters and lengths of uninterrupted sequences of capital letters in emails.

```
data mylib.JunkMail;
   set Sashelp.JunkMail;
run;
```

In the following program, the PARTITION statement divides the data into two parts. The training data have a Test value of 0 and contain about two-thirds of the data; the rest of the data are used to evaluate the fit. The macro function SplineVList is defined to specify spline terms by using the default construction.

```
%macro SplineVList(vars);
   %let i = 1;
   %do %while (%length(%scan(&vars,&i)));
      spline(%scan(&vars,&i))
      %let i = %eval(&i + 1);
   %end;
%mend;
```

```
proc gamselect data=mylib.JunkMail;
   model Class(event='1')= %SplineVList(Make Address All _3d Our Over Remove
         Internet Order Mail Receive Will People Report Addresses Free Business
         Email You Credit Your Font _000 Money HP HPL George _650 Lab Labs
         Telnet _857 Data _415 _85 Technology _1999 Parts PM Direct CS Meeting
         Original Project RE Edu Table Conference Semicolon Paren Bracket
         Exclamation Dollar Pound CapAvg CapLong CapTotal)
   / dist=binary allobs;
   partition rolevar=Test(train='0' test='1');
   selection method=boosting(maxIter=2000);
run;
```

The ALLOBS option specifies that all observations be used to construct the spline basis functions. By default, only observations that have the training data role are used to construct the spline basis functions. By default, observations with the validation or test role that have spline variable values outside the range of values that is used to construct the spline basis functions are omitted from the analysis.

The "Number of Observations" and "Response Profile" tables in Output 13.1.1 are divided into training and testing columns.

Output 13.1.1 Partitioned Counts

The GAMSELECT Procedure

Number of Observations Read	4601
Number of Observations Used	4601
Number of Observations Used for Training	3065
Number of Observations Used for Validation	0
Number of Observations Used for Testing	1536

Response Profile

Ordered Value	Class	Total Frequency	Training Frequency	Validation Frequency	Test Frequency
1	0	2788	1847	0	941
2	1	1813	1218	0	595

Probability modeled is Class = 1.

The "Iteration Summary" table in Output 13.1.2 shows that the boosting algorithm executed 2,000 iterations and selected the model that corresponds to the final iteration with 26 effects.

Output 13.1.2 Iteration Summary

Iteration Summary	
Iterations Executed	2000
Selected Iteration	2000
Number of Selected Effects	26

The fit statistics for the selected model are displayed in the "Fit Statistics" table in Output 13.1.3.

Output 13.1.3 Partitioned Fit Statistics

Fit Statistics	
ASE (Train)	0.05450
ASE (Test)	0.06042
Misclassification Rate (Train)	0.06623
Misclassification Rate (Test)	0.06771

These statistics are computed for both the training and testing data. The average square error (ASE) and the misclassification rate should be similar between the two groups when the training data are representative of the testing data; for this model, the values of these statistics seem similar between the two disjoint subsets.

The "Selected Effects" table in Output 13.1.4 shows the effects that are included in the selected model, the iteration at which the effect first entered the model, and the number of iterations at which the effect was selected for an update.

Output 13.1.4 Selected Effects

Selected Effects		
Effect	Entry Iteration	Times Selected
Spline(Your)	1	24
Spline(Dollar)	2	127
Spline(Exclamation)	12	126
Spline(Free)	22	84
Spline(Remove)	25	113
Spline(HP)	80	229
Spline(_000)	81	80
Spline(Money)	103	75
Spline(Our)	105	89
Spline(CapLong)	134	75
Spline(George)	189	223
Spline(Internet)	218	64
Spline(CapTotal)	312	68
Spline(_1999)	323	127
Spline(Edu)	341	117
Spline(Font)	436	67
Spline(Business)	509	56
Spline(Meeting)	527	86
Spline(RE)	786	47
Spline(Over)	956	35
Spline(Project)	1283	35
Spline(Address)	1319	24
Spline(Will)	1618	14
Spline(Semicolon)	1645	12
Spline(Conference)	1984	1
Spline(CapAvg)	1986	2

For comparison, the following program uses PROC LOGSELECT to fit a parametric model by using forward selection and the same set of independent variables:

```
proc logselect data=mylib.JunkMail;
   model Class(event='1')=Make Address All _3d Our Over Remove Internet Order
         Mail Receive Will People Report Addresses Free Business Email You
         Credit Your Font _000 Money HP HPL George _650 Lab Labs Telnet _857
         Data _415 _85 Technology _1999 Parts PM Direct CS Meeting Original
         Project RE Edu Table Conference Semicolon Paren Bracket Exclamation
         Dollar Pound CapAvg CapLong CapTotal;
   partition rolevar=Test(train='0' test='1');
   selection method=forward;
run;
```

The "Selected Effects" table in Output 13.1.5 displays the 24 effects in the model that is selected by PROC LOGSELECT. The models that are selected by PROC GAMSELECT and PROC LOGSELECT have 22 variables in common.

Output 13.1.5 PROC LOGSELECT Selected Effects

The LOGSELECT Procedure

Selection Details

Selected Effects:	Intercept Our Over Remove Internet Order Will Free Business You Your Font _000 Money HP George Parts Meeting RE Edu Semicolon Exclamation Dollar CapAvg CapLong

The "Fit Statistics" table for the model that PROC LOGSELECT fits is shown in Output 13.1.6. The ASE and misclassification rates for both the training and test data are smaller for the model that PROC GAMSELECT fits (Output 13.1.3).

Output 13.1.6 PROC LOGSELECT Fit Statistics

Fit Statistics		
Description	**Training**	**Testing**
-2 Log Likelihood	1242.59491	823.68742
AIC (smaller is better)	1292.59491	873.68742
AICC (smaller is better)	1293.02268	874.54835
SBC (smaller is better)	1443.28998	1007.11085
Average Square Error	0.05659	0.06353
-2 Log L (Intercept-only)	4118.98701	2050.73514
R-Square	0.60877	0.55016
Max-rescaled R-Square	0.82359	0.74661
McFadden's R-Square	0.69833	0.59835
Misclassification Rate	0.07471	0.07813
Difference of Means	0.75122	0.73431

Example 13.2: Options to Control Boosting Model Fit

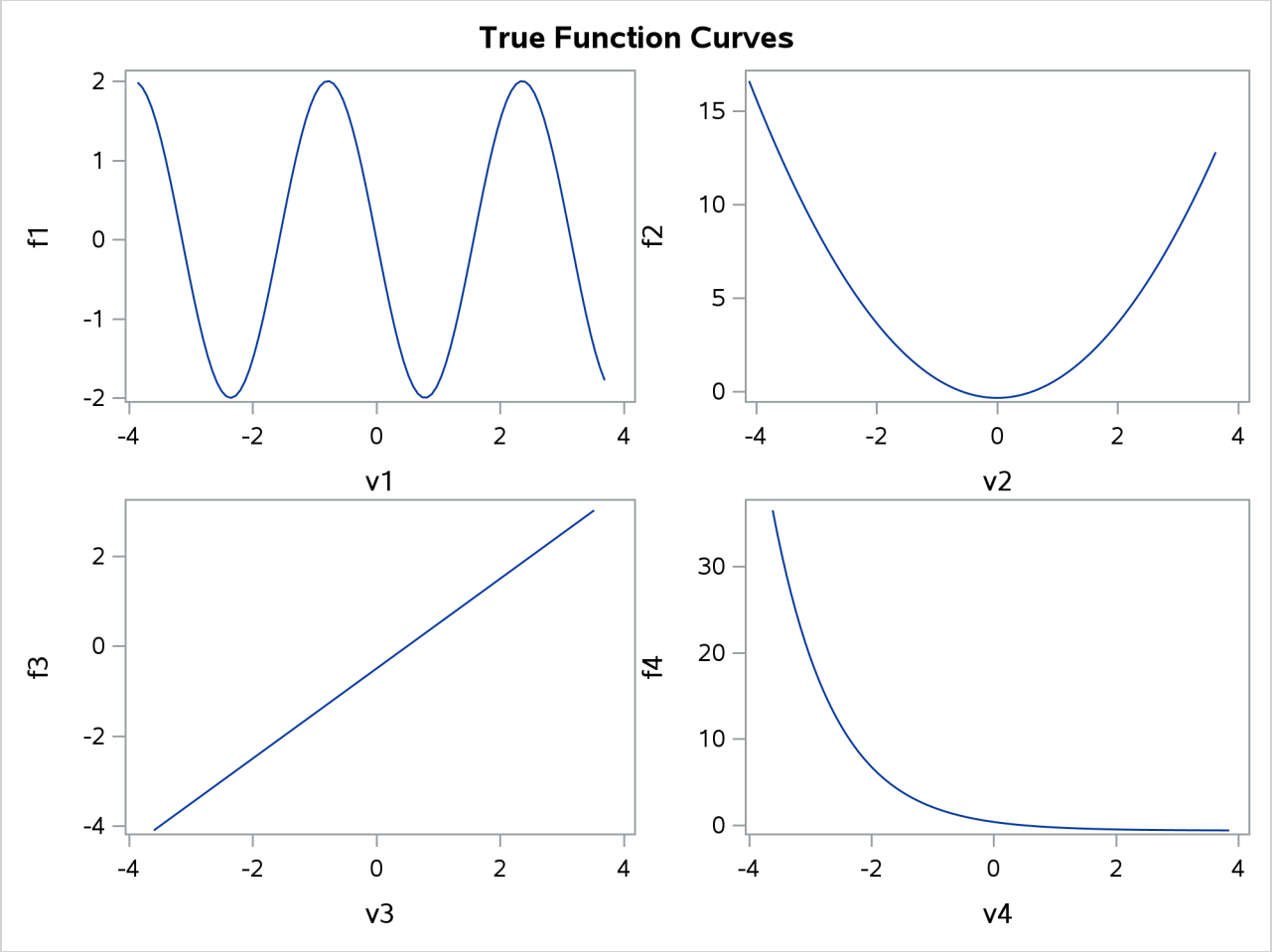
(View the complete code for this example (gamselex2.sas) in the example repository.)

This example and Example 13.3 demonstrate how you can build an additive model and how you can control the complexity of the model that PROC GAMSELECT fits. The following DATA step creates the data table one in the current CAS engine libref named mylib:

```
data mylib.one;
   call streaminit(1);
   array v{200} v1-v200;
   do i=1 to 10000;
      do j=1 to 200;
          v{j}=rand("normal");
      end;
      f1=-2*sin(2*v1);
      f2=v2*v2-1./3;
      f3=v3-0.5;
      f4=exp(-v4)+exp(-1)-1;
      linp=f1+f2+f3+f4;
      y=rand("normal",linp);
      output;
   end;
 run;
```

The data table one consists of 10,000 observations for a continuous response variable (y) and 200 continuous variables (v1–v200). The model for the mean of response variable y includes a sine function of the variable v1, a quadratic function of the variable v2, an exponential function of the variable v3, and a linear function of the variable v4. Output 13.2.1 shows the true function curves of the four variables that construct the response.

Output 13.2.1 True Function Curves



Without knowing the data generating process, you could form an initial analysis plan by using some basic exploratory tools. A density histogram of the response variable y (not shown) suggests that it follows a normal distribution, and a scatter matrix (not shown) indicates nonlinear relationships between some of the predictors and the response and many possible nuisance predictors. On the basis of these observations, you might decide to first use PROC REGSELECT to perform the selection. This procedure enables you to use the EFFECT statement to construct regression splines to model nonlinear terms. For more information about the procedure, see Chapter 29, "REGSELECT Procedure." For more information about using the EFFECT statement in PROC REGSELECT to construct regression splines, see the section "Spline Effects" on page 25 in Chapter 2, "Shared Concepts." The following program shows how you can start your analysis by performing the stepwise selection on 200 constructed regression splines:

```
proc regselect data=mylib.one;
   effect spl=spline(v1-v200/separate);
   model y=spl;
   selection method=stepwise;
run;
```

The output from PROC REGSELECT (Output 13.2.2) shows that five effects, including the intercept, are selected and shows an average square error (ASE) of about 1.4 for the selected model.

Output 13.2.2 Selected Effects and Fit Statistics

The REGSELECT Procedure

Selection Details

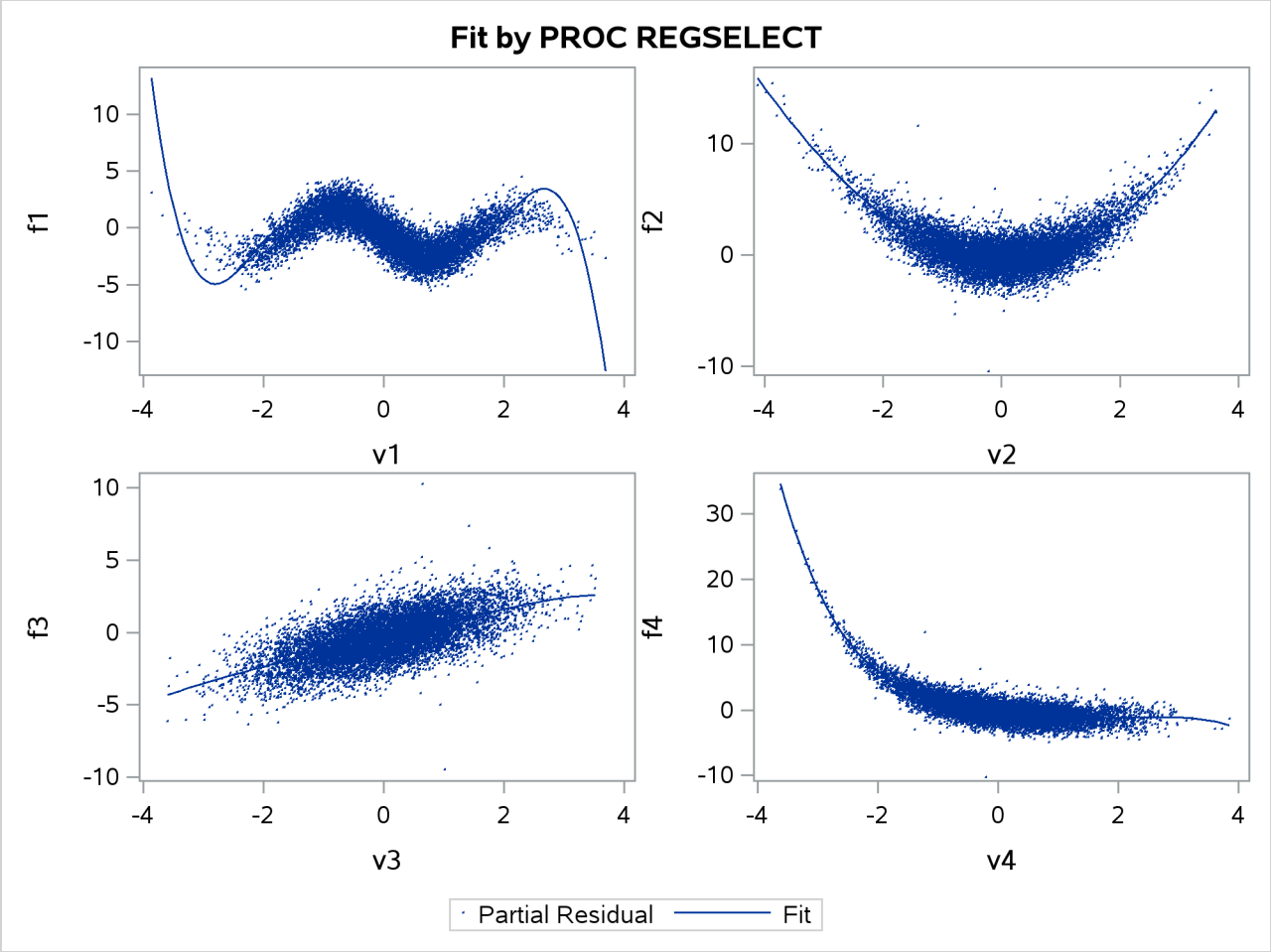
Selected Effects:	Intercept spl_v1 spl_v2 spl_v3 spl_v4

Root MSE	1.18341
R-Square	0.86648
Adj R-Sq	0.86616
AIC	13395
AICC	13395
SBC	3573.31162
ASE	1.39697

For regression modeling of data that have nonlinear structures, it is often necessary to check partial residuals and partial predictions to determine whether the included model effects sufficiently explain the nonlinear structure. Output 13.2.3 displays the partial predictions overlaid with fitted partial prediction curves for the selected model.

In Output 13.2.3, you observe that the fitted curves explain the pattern in the partial residuals reasonably well, except for the first variable, v1. The fitted curve seems to overfit the partial residuals, especially in the tail areas. This suggests that the default regression spline construction is not sufficient for modeling the nonlinear relationship between v1 and y.

Output 13.2.3 PROC REGSELECT Fit



This problem has a few possible solutions. One solution is to construct multiple spline bases at different scales of knot intervals in order to approximate the nonlinear structure at finer scales and plot the partial residuals for the new models. In general, you will have to keep repeating these steps in order to obtain a model that explains the nonlinear structure well. For data that have complicated dependency structures, this might take many iterations. However, you can use PROC GAMSELECT to perform an additive model selection that uses penalized splines. Because of the spline penalty, the fitted splines are less prone to exhibit extreme boundary behaviors. This example and Example 13.3 demonstrate how you can use the procedure to perform the selection and use some options to fine-tune the selection results.

The following PROC SQL statements create a macro variable, splineTerms, which contains the list of spline terms for the variables v1–v200:

```
proc sql noprint;
   select cat("spline(",strip(name),")") into :splineTerms separated by ' '
   from dictionary.columns
   where libname = "MYLIB" and memname = "ONE" and
   upcase(name) like 'V%';
quit;
```

The following PROC GAMSELECT statements use the macro variable splineTerms to specify a model for the variable y by using the boosting method without specifying any options to control the model fit:

```
proc gamselect data=mylib.one plots=all;
   model y = &splineTerms;
   selection method=boosting;
run;
```

Output 13.2.4 shows that the boosting algorithm executed 500 iterations and selected the model that corresponds to the final boosting iteration.

Output 13.2.4 Iteration Summary for Default Model

The GAMSELECT Procedure

Iteration Summary	
Iterations Executed	500
Selected Iteration	500
Number of Selected Effects	8

The "Selected Effects" table in Output 13.2.5 shows that the four spline terms that are constructed from the variables v1–v4 are included in the final model and that at most iterations the effect that was selected for an update was one of these four terms.

Output 13.2.5 Selected Effects for Default Model

Selected Effects		
Effect	Entry Iteration	Times Selected
Spline(v4)	1	187
Spline(v2)	5	67
Spline(v1)	9	204
Spline(v3)	18	35
Spline(v15)	443	3
Spline(v51)	464	2
Spline(v148)	472	1
Spline(v98)	493	1

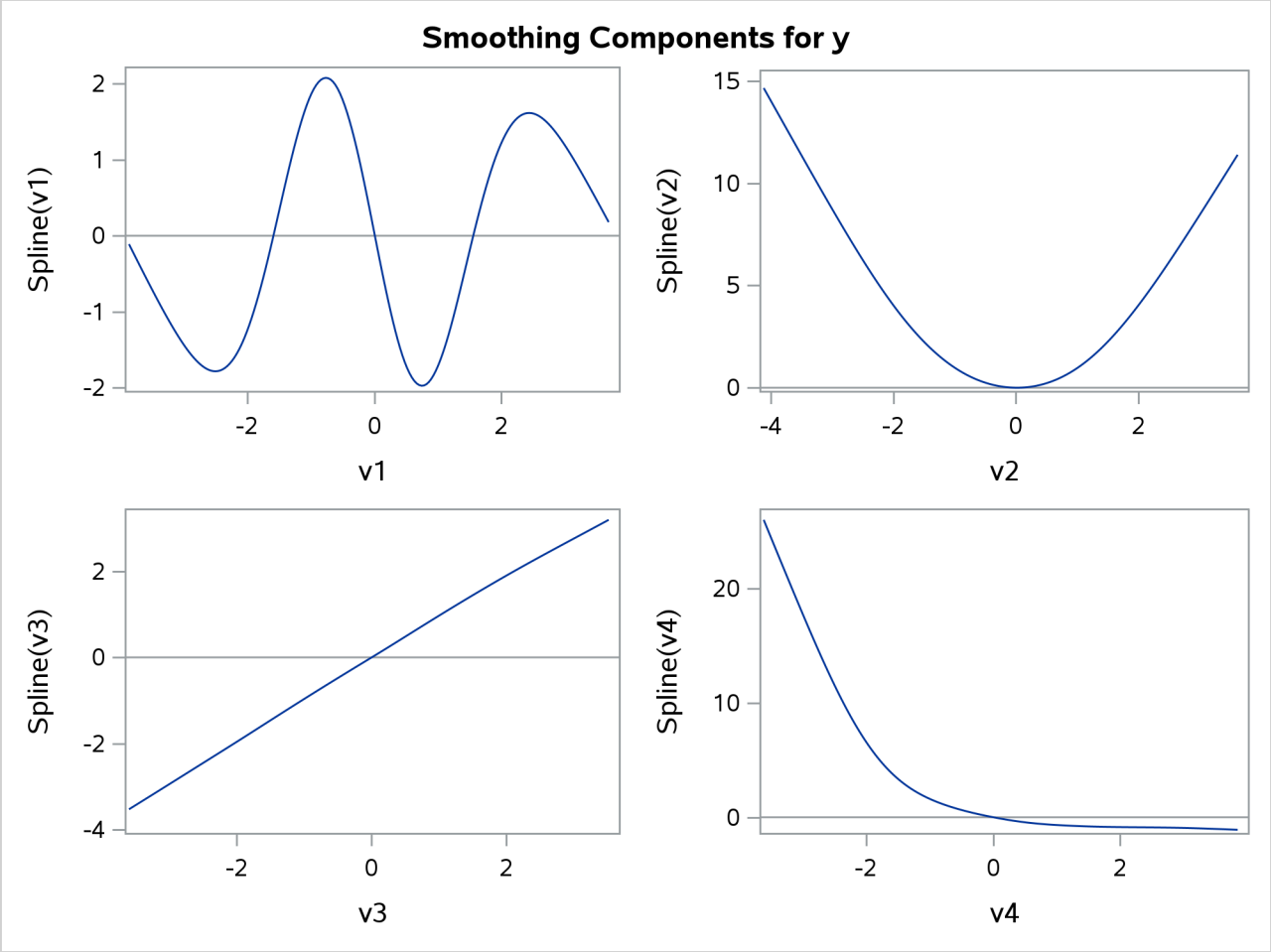
The model also includes four effects that do not correspond to any true signal. These effects enter the model at iterations later in the run of the boosting algorithm and are selected for a small number of iterations. Output 13.2.6 shows the fit statistics for the selected model; in this case, there is only one—the ASE.

Output 13.2.6 Fit Statistics for Default Model

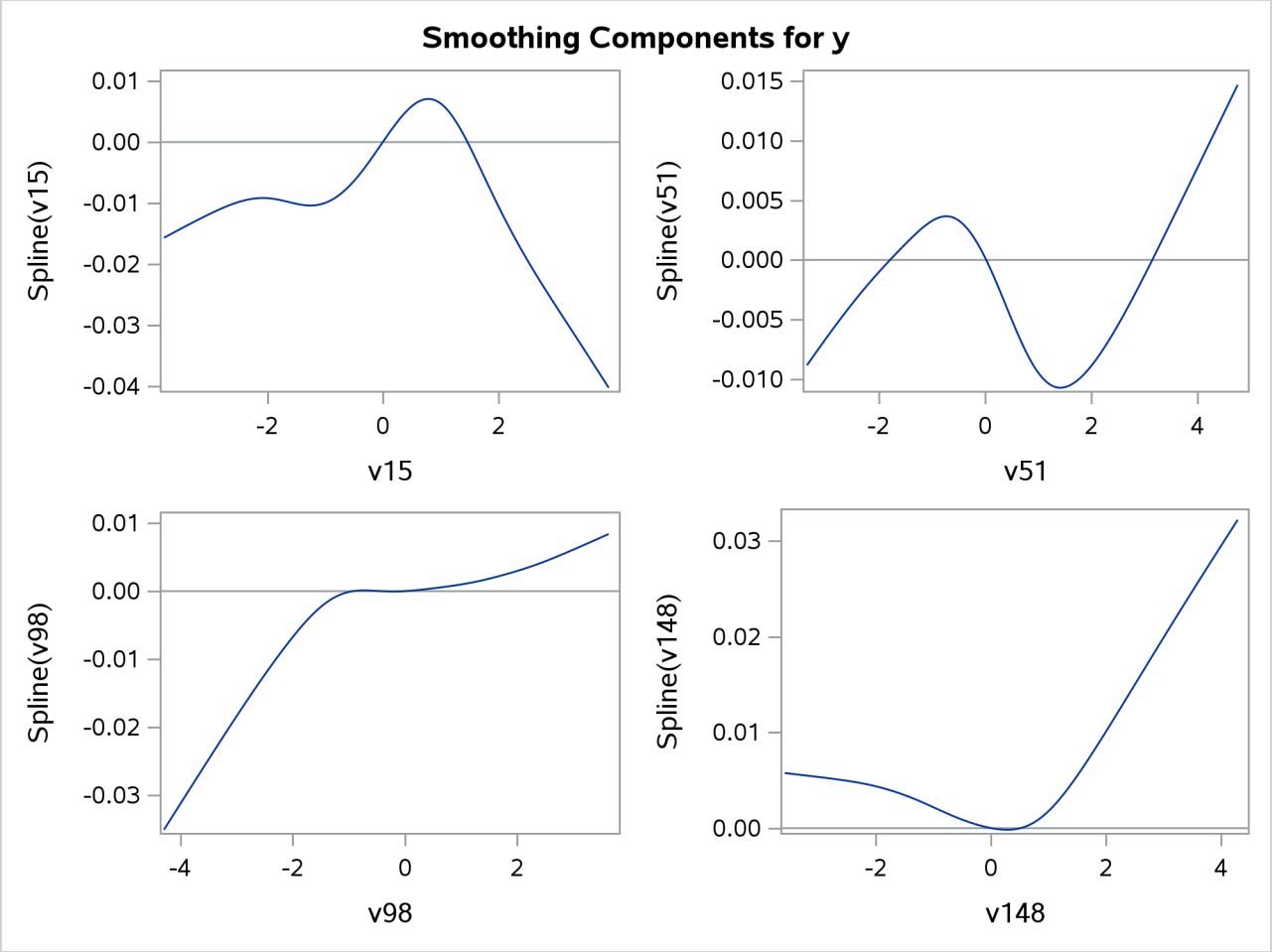
Fit Statistics	
Average Square Error	1.03910

The smoothing component plots for the selected effects in Output 13.2.7 show well-reconstructed curves for the splines that are constructed from the variables v1–v4 and show small contributions from the remaining components.

Output 13.2.7 Smoothing Component Panel for Default Model



Output 13.2.7 *continued*



By default, PROC GAMSELECT selects the model $\hat{f}^{m^*}$ that minimizes the ASE for the training data from the set of models that correspond to each boosting iteration. This criterion for selecting the final model might result in overfitting of the training data. To prevent overfitting of the training data, you can use the CHOOSE= option in the SELECTION statement to change the criterion that is used to select the final model. In addition to changing this criterion, you can implement early stopping of the selection process by using the STOPHORIZON= and STOPTOL= options in the SELECTION statement. For more information about how PROC GAMSELECT implements early stopping, see the section "Boosting" on page 674.

To demonstrate the use of early stopping, the model selection process is repeated using k-fold cross validation of the ASE as the selection criterion. PROC GAMSELECT can perform cross validation either by randomly assigning observations to folds or by using an input variable to partition the data into folds. Note that the random sampling of observations for cross validation depends on the initial random number seed and how the data are distributed across machines and threads. Therefore, even if you specify the same seed value by using the SEED= option in the PROC GAMSELECT statement, results might differ between runs of PROC GAMSELECT. The following DATA step adds a variable cvFold that can be used to partition the data into folds for cross validation:

```
data mylib.one / single=yes;
  set mylib.one;
  call streaminit(1848);
  cvFold = rand("table",0.2,0.2,0.2,0.2,0.2);
run;
```

The following PROC GAMSELECT statements perform model selection by using early stopping, k-fold cross validation to select the final model, and the same list of spline terms that is specified in the splineTerms macro variable:

```
proc gamselect data=mylib.one plots=all;
   model y = &splineTerms;
   selection method=boosting(choose=CV index=cvFold
             stopHorizon=10 stopTol = 0.0005);
run;
```

Output 13.2.8 shows that the selection process terminates early, at the 300th boosting iteration, and that the selected model includes four effects.

Output 13.2.8 Iteration Summary Using Cross Validation

The GAMSELECT Procedure

Iteration Summary	
Iterations Executed	300
Selected Iteration	300
Number of Selected Effects	4

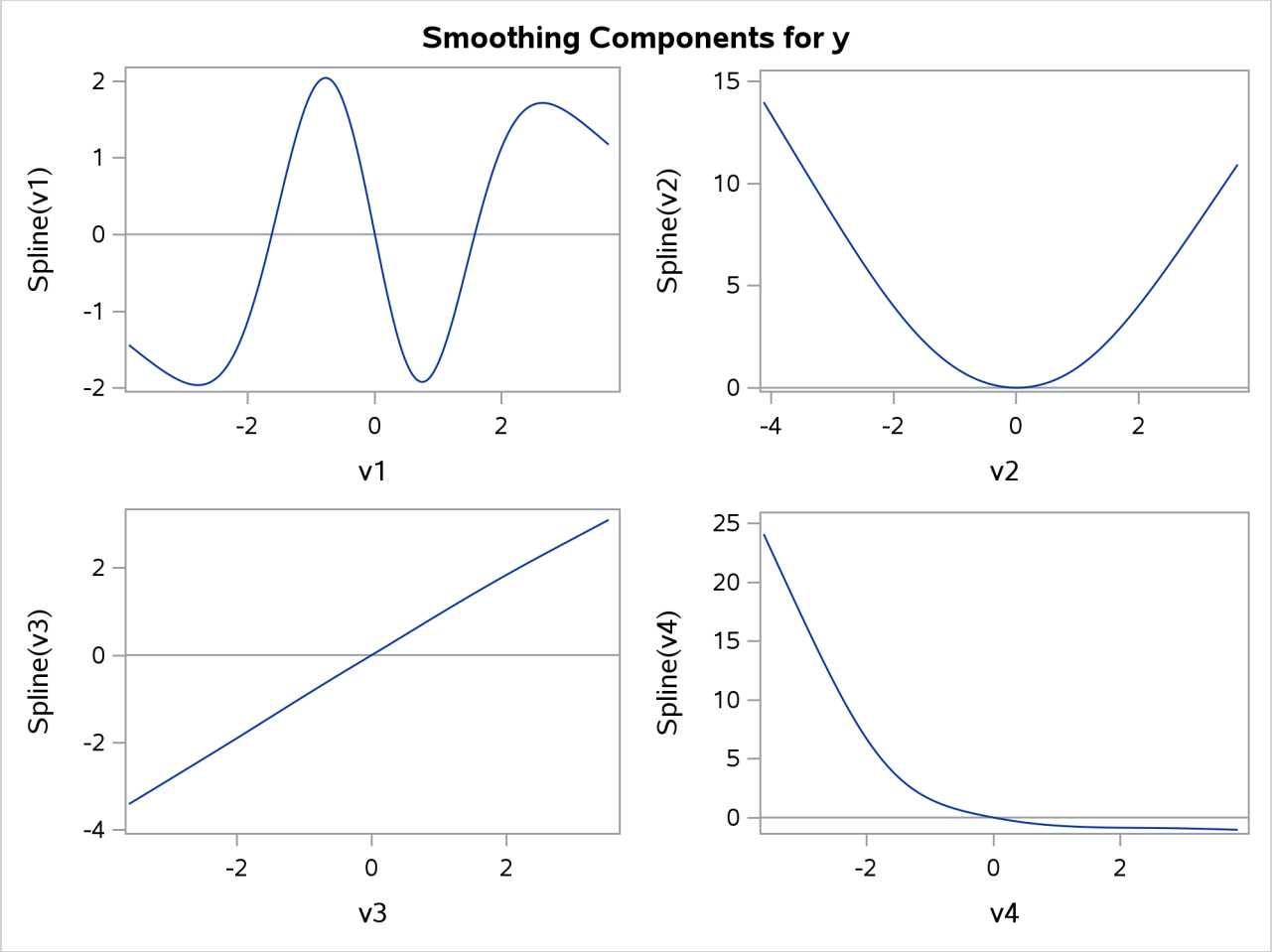
The "Fit Statistics" table in Output 13.2.9 shows the ASE for the training data and the k-fold cross validation of the ASE. The two ASE values are similar and are larger than the ASE value for the default model (Output 13.2.6).

Output 13.2.9 Fit Statistics Using Cross Validation

Fit Statistics	
Average Square Error	1.07588
Cross Validation ASE	1.09528

Output 13.2.10 displays the smoothing component panel for the model that is selected using cross validation and early stopping. Compared to the smoothing component panel for the default model (Output 13.2.7), the curve for Spline(v1) in Output 13.2.10 shows a worse reconstruction near the maximum and minimum values of the variable v1. This difference suggests that the fit might benefit from increasing the degrees of freedom for the penalized B-spline for Spline(v1) fit at each boosting iteration. For more information about the construction of penalized B-splines, see the section "Penalized B-Splines" on page 671.

Output 13.2.10 Smoothing Component Panel Using Cross Validation



The following PROC GAMSELECT statements fit a reduced model by using spline terms for the variables v1–v4 with 10 degrees of freedom for the Spline(v1) estimate fit at each boosting iteration:

```
proc gamselect data=mylib.one plots=all;
   model y = spline(v1 / df = 10) spline(v2) spline(v3) spline(v4);
   selection method=boosting(choose=CV index=cvFold
             stopHorizon=10 stopTol = 0.0005);
run;
```

Output 13.2.11 displays the "Fit Statistics" table and "Iteration Summary" table for the reduced model. The selection process terminates early, at the 219th iteration, and has an ASE closer to the ASE of the default model (Output 13.2.6).

Output 13.2.11 Iteration Summary and Fit Statistics for Reduced Model

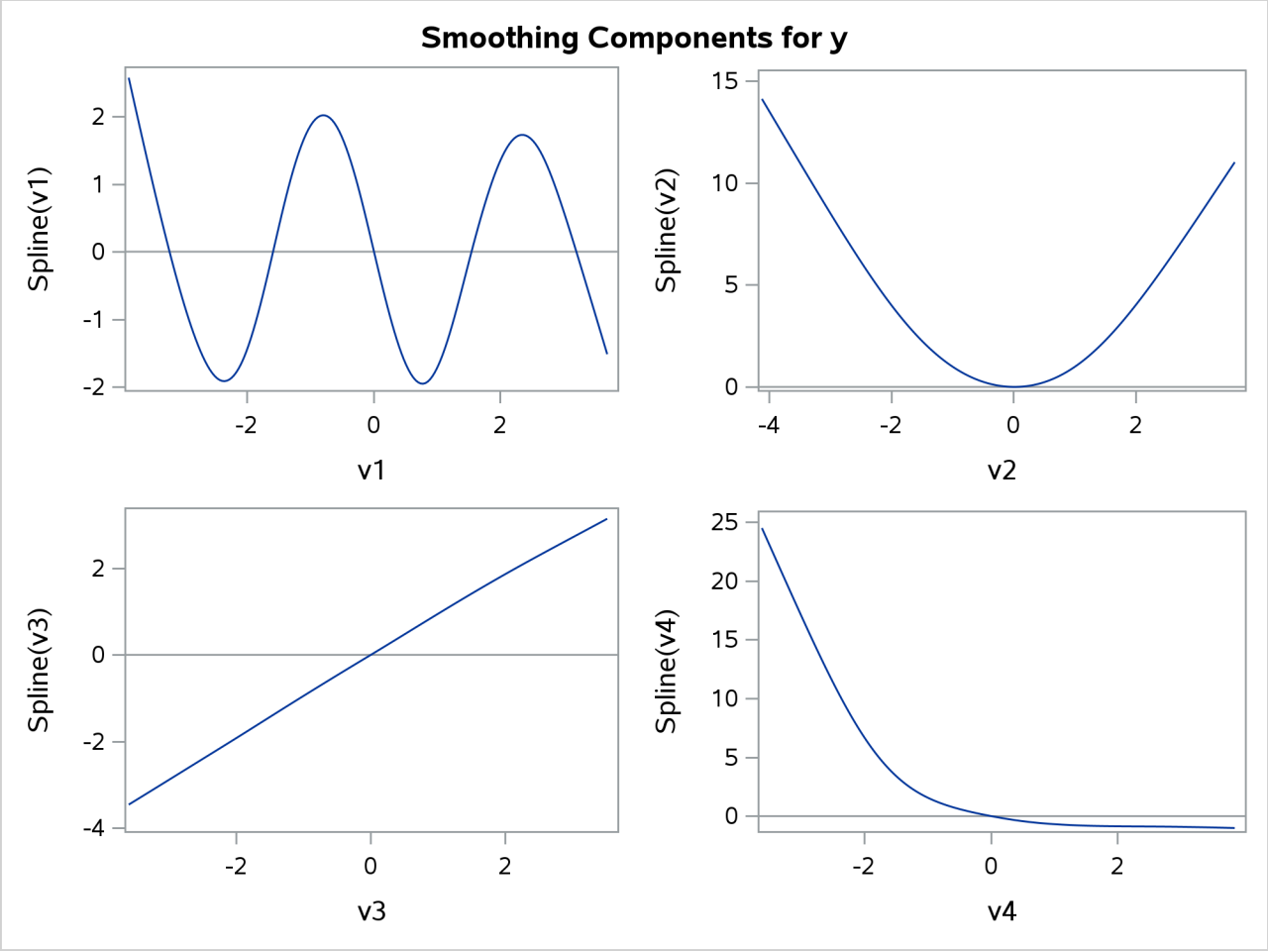
The GAMSELECT Procedure

Iteration Summary	
Iterations Executed	219
Selected Iteration	219
Number of Selected Effects	4

Fit Statistics	
Average Square Error	1.04808
Cross Validation ASE	1.06386

The increased degrees of freedom for Spline(v1) results in a better reconstruction than the model fit that uses the default 4 degrees of freedom with cross validation and early stopping (Output 13.2.10).

Output 13.2.12 Smoothing Component Panel for Reduced Model



Example 13.3: Options to Control Shrinkage Selection

(View the complete code for this example (gamselex3.sas) in the example repository.)

This example continues the analysis of building an additive model for the data table that is formed in Example 13.2.

The following program uses the shrinkage method to perform the selection:

```
proc gamselect data=mylib.one seed=123 plots=all;
   model y = &splineTerms;
   selection method=shrinkage;
run;
```

Output 13.3.1 shows that the selected model converged and the fittings for all the regularization parameter candidates succeeded.

Output 13.3.1 Convergence Status for Default Model

The GAMSELECT Procedure

Selected model converged. All regularization parameters succeeded.

Output 13.3.2 shows the number of tries and successes of regularization parameter fittings. It also shows how many iterations the selected model required and how many effects were selected.

Output 13.3.2 Iteration Summary for Default Model

Iteration Summary	
Number of Penalty Parameter Set Tries	200
Number of Penalty Parameter Set Successes	200
Number of Reweighted Iterations for Selected Model	12
Number of Selected Effects	6

Output 13.3.3 displays the upper bounds for the regularization parameters that PROC GAMSELECT computed from the input data table and the selected values after the grid search.

Output 13.3.3 Regularization Parameter Summary for Default Model

Shrinkage Regularization Parameter Summary		
Penalty	**Maximum**	**Chosen**
Sparsity	2.097253	0.030225
Smoothness	0.5503	0.001075
Generalized Ridge	0	0

Output 13.3.4 shows that six effects are selected: the intercept, the splines of four variables (v1–v4) that construct the true model, and one additional spline of variable v15.

Output 13.3.4 Selected Effects for Default Model

Selected Effects	Intercept Spline(v1) Spline(v2) Spline(v3) Spline(v4) Spline(v15)

Output 13.3.5 displays the fit statistics for the model; in this case, there is only one—the ASE.

Output 13.3.5 Fit Statistics for Default Model

Fit Statistics	
Average Square Error	1.01459

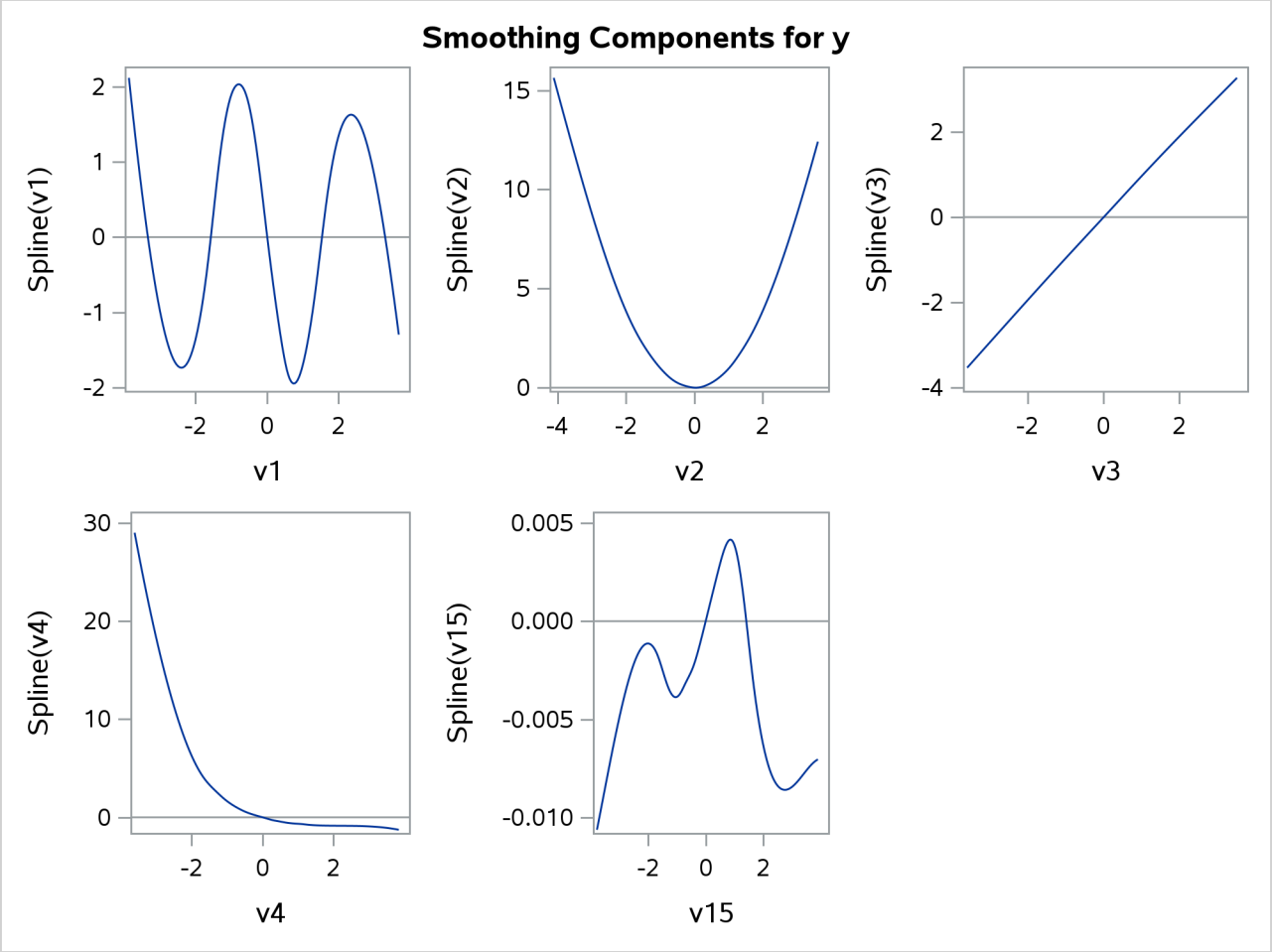
Output 13.3.6 displays the values of sparsity penalty, smoothness penalty, and generalized ridge penalty for the selected spline terms. Compared to the first four splines, the last spline of variable v15 has somewhat smaller penalty values.

Output 13.3.6 Spline Regularization for Default Model

Spline Regularization			
Component	**Sparsity Penalty**	**Smoothness Penalty**	**Generalized Ridge Penalty**
Spline(v1)	0.0422	0.0143	0
Spline(v2)	0.0436	0.00521	0
Spline(v3)	0.0289	0.000061	0
Spline(v4)	0.0620	0.0120	0
Spline(v15)	0.000088	0.000031	0

The "Smoothing Component Panel" displays the fitted curves of the five selected spline terms. The vertical scale of Spline(v15) indicates that the contribution from this effect is small compared to the contribution from the other spline effects.

Output 13.3.7 Smoothing Component Panel for Default Model



Meier, Van de Geer, and Bühlmann (2009) propose an adaptive operation that forms sparsity penalties and smoothness penalties in a data-adaptive way, which is similar to the idea of the adaptive LASSO (Zou 2006). In the context of shrinkage selection, this adaptive option corresponds to setting the WEIGHT1 and WEIGHT2 options in individual spline terms, on the basis of previously computed sparsity penalty and smoothness penalty values. Using the notation in the section "Shrinkage" on page 676, you compute the adaptive weights as

$$\phi_j = 1/\sqrt{\hat{\boldsymbol{\beta}}_j' \mathbf{K}_j \hat{\boldsymbol{\beta}}_j}$$
$$\psi_j = 1/\sqrt{\hat{\boldsymbol{\beta}}_j' \boldsymbol{\Omega}_j \hat{\boldsymbol{\beta}}_j}$$

where $\sqrt{\hat{\boldsymbol{\beta}}_j' \mathbf{K}_j \hat{\boldsymbol{\beta}}_j}$ and $\sqrt{\hat{\boldsymbol{\beta}}_j' \boldsymbol{\Omega}_j \hat{\boldsymbol{\beta}}_j}$ are computed by using the selected regularization parameters (Output 13.3.3) divided by the reported sparsity penalty and smoothness penalty values, respectively, shown in Output 13.3.6.

The following program uses the computed values of ϕ_j and ψ_j from the previous analysis and performs one additional step of additive model selection in an adaptive fashion:

```
proc gamselect data=mylib.one seed=123 plots=all;
   model y=spline(v1 / weight1=0.716  weight2=0.075)
           spline(v2 / weight1=0.693  weight2=0.206)
           spline(v3 / weight1=1.046  weight2=17.62)
           spline(v4 / weight1=0.488  weight2=0.090)
           spline(v15/ weight1=343.5  weight2=34.68);
   selection method=shrinkage;
run;
```

Output 13.3.8 shows that the intercept and splines of the four variables that construct the true response were selected in the adaptive model.

Output 13.3.8 Selected Effects for Adaptive Model

The GAMSELECT Procedure

Selected Effects	Intercept Spline(v1) Spline(v2) Spline(v3) Spline(v4)

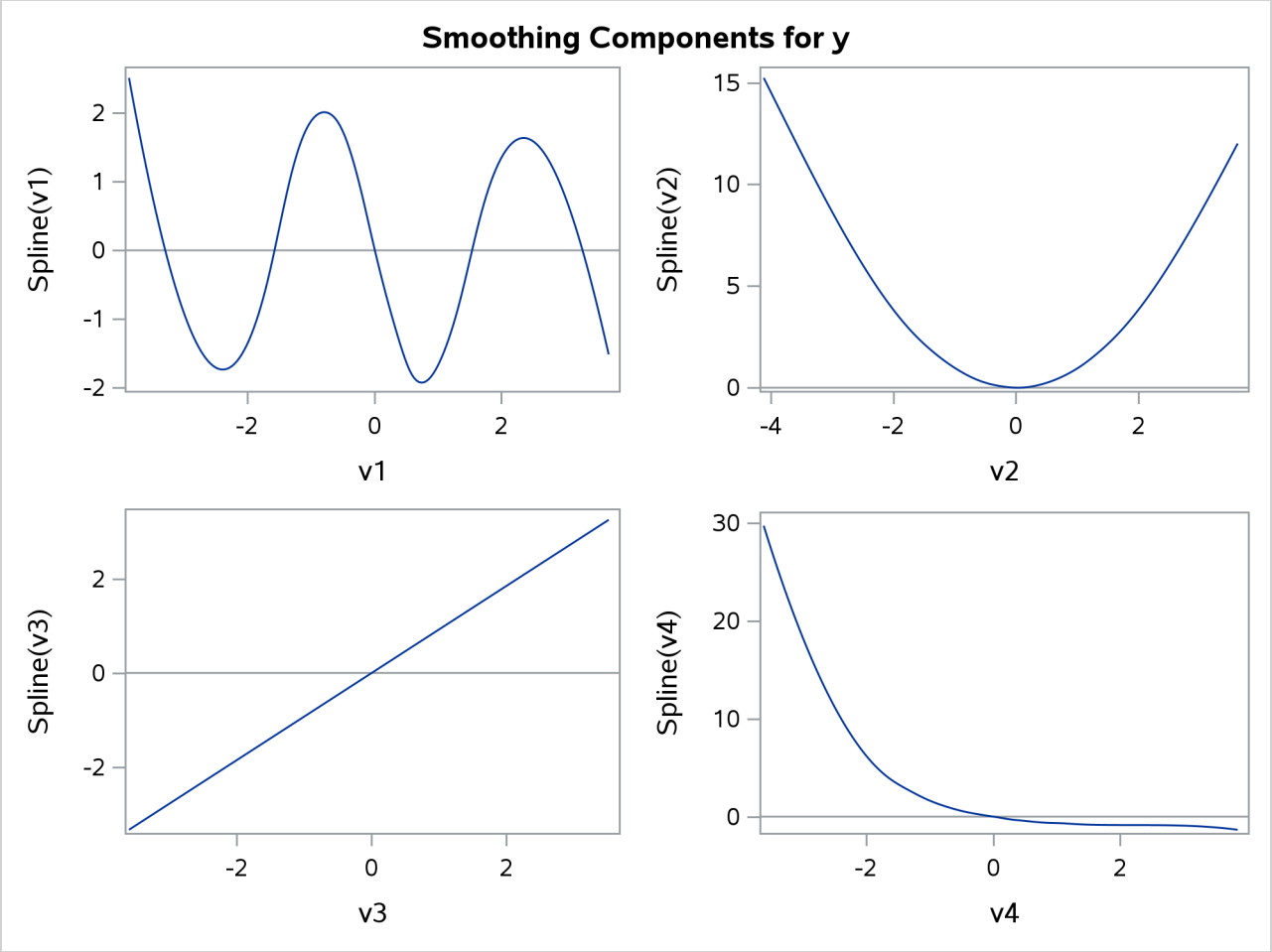
Output 13.3.9 displays the fit statistics for the adaptive model; in this case, there is only one—the ASE.

Output 13.3.9 Fit Statistics for Adaptive Model

Fit Statistics	
Average Square Error	1.01867

Output 13.3.10 shows the fitted curves from the adaptive model.

Output 13.3.10 Smoothing Component Panel for Adaptive Model



References

Amato, U., Antoniadis, A., and De Feis, I. (2016). "Additive Model Selection." *Statistical Methods and Applications* 25:519–564.

Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. (2011). "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers." *Foundations and Trends in Machine Learning* 2:1–122.

Bühlmann, P., and Hothorn, T. (2007). "Boosting Algorithms: Regularization, Prediction and Model Fitting." *Statistical Science* 22:477–505. https://doi.org/10.1214/07-STS242.

Cantoni, E., Flemming, J. M., and Ronchetti, E. (2006). "Variable Selection in Additive Models by Non-negative Garrote." *Statistical Modelling* 11:237–252.

Chouldechova, A., and Hastie, T. J. (2015). "Generalized Additive Model Selection." Technical paper. Stanford University.

De Boor, C. (1978). *A Practical Guide to Splines*. New York: Springer-Verlag.

Eilers, P. H. C., and Marx, B. D. (1996). "Flexible Smoothing with *B*-Splines and Penalties." *Statistical Science* 11:89–121. With discussion.

Friedman, J. H. (2001). "Greedy Function Approximation: A Gradient Boosting Machine." *Annals of Statistics* 29:1189–1232.

Friedman, J. H., Hastie, T., and Tibshirani, R. (2000). "Additive Logistic Regression: A Statistical View of Boosting." *Annals of Statistics* 28:337–407. With discussion.

Koltchinskii, V., and Yuan, M. (2010). "Sparsity in Multiple Kernel Learning." *Annals of Statistics* 38:3660–3695.

Lin, Y., and Zhang, H. H. (2006). "Component Selection and Smoothing in Multivariate Nonparametric Regression." *Annals of Statistics* 34:2272–2297.

Marra, G., and Wood, S. N. (2011). "Practical Variable Selection for Generalized Additive Models." *Computational Statistics and Data Analysis* 55:2372–2387.

Marx, B. D., and Eilers, P. H. C. (2005). "Multidimensional Penalized Signal Regression." *Technometrics* 47:13–22. https://doi.org/10.1198/004017004000000626.

Meier, L., Van de Geer, S., and Bühlmann, P. (2009). "High-Dimensional Additive Modeling." *Annals of Statistics* 37:3779–3821.

Nelder, J. A., and Wedderburn, R. W. M. (1972). "Generalized Linear Models." *Journal of the Royal Statistical Society, Series A* 135:370–384.

O'Sullivan, F. (1986). "A Statistical Perspective on Ill-Posed Inverse Problems." *Statistical Science* 1:502–518.

Raskutti, G., Wainwright, M. J., and Yu, B. (2012). "Minimax-Optimal Rates for Sparse Additive Models over Kernel Classes via Convex Programming." *Journal of Machine Learning Research* 13:389–427.

Ravikumar, P., Lafferty, J., Liu, H., and Wasserman, L. (2009). "Sparse Additive Models." *Journal of the Royal Statistical Society, Series B* 71:1009–1030.

Suzuki, T., and Sugiyama, M. (2013). "Fast Learning Rate of Multiple Kernel Learning: Trade-Off between Sparsity and Smoothness." *Annals of Statistics* 41:1381–1405.

Tomioka, R., and Suzuki, T. (2009). "Sparsity-Accuracy Trade-Off in MKL." Presented at NIPS 2009 Workshop: Understanding Multiple Kernel Learning Methods, Whistler, British Columbia.

Wand, M. P., and Ormerod, J. T. (2008). "On Semiparametric Regression with O'Sullivan Penalized Splines." *Australian and New Zealand Journal of Statistics* 50:179–198.

Yuan, M., and Lin, L. (2006). "Model Selection and Estimation in Regression with Grouped Variables." *Journal of the Royal Statistical Society, Series B* 68:49–67.

Zou, H. (2006). "The Adaptive Lasso and Its Oracle Properties." *Journal of the American Statistical Association* 101:1418–1429.

Chapter 14
GENSELECT Procedure

Contents

Overview: GENSELECT Procedure **704**
PROC GENSELECT Features 704
PROC GENSELECT Compared with Other SAS Procedures 705
Using CAS Sessions and CAS Engine Librefs 707
Getting Started: GENSELECT Procedure **708**
Poisson Regression for Count Data 708
Syntax: GENSELECT Procedure **713**
PROC GENSELECT Statement 713
BY Statement 719
CLASS Statement 719
CODE Statement 720
DISPLAY Statement 720
DISPLAYOUT Statement 721
EFFECT Statement 722
FREQ Statement 724
LSMEANS Statement 724
MODEL Statement 725
OUTPUT Statement 732
PARTITION Statement 737
REPEATED Statement 738
RESTRICT Statement 741
SELECTION Statement 744
STORE Statement 746
WEIGHT Statement 746
Details: GENSELECT Procedure **747**
Missing Values 747
Exponential Family Distributions 747
Response Distributions 748
Response Probability Distribution Functions 749
Log-Likelihood Functions 754
Existence of Maximum Likelihood Estimates 759
The LASSO Method of Model Selection 760
Generalized Estimating Equations 761
Partition Fit Statistics 767
Model Fit and Assessment Statistics 767
Predicted Values and Regression Diagnostics 768

Joint Tests and Type 3 Tests . . . 772
Multithreading . . . 773
Optimization Algorithms . . . 773
Displayed Output . . . 774
ODS Table Names . . . 780
ODS Graphics . . . 782
Examples: GENSELECT Procedure . . . **783**
Example 14.1: Model Selection . . . 783
Example 14.2: Gamma Model . . . 787
Example 14.3: Binary Model with Repeated Measures . . . 789
References . . . **792**

Overview: GENSELECT Procedure

The GENSELECT procedure fits and performs model selection for generalized linear models in SAS Viya.

The models that PROC GENSELECT supports can contain main effects that consist of both continuous and classification variables and interaction effects of these variables. The models can also include constructed effects such as splines. The procedure offers a number of effect-selection methods, including stepwise methods, modern LASSO methods, and elastic net selection. It also offers extensive capabilities for customizing the model selection by using a wide variety of selection and stopping criteria, from computationally efficient significance-level-based criteria to modern, computationally intensive validation-based criteria. PROC GENSELECT also provides a variety of diagnostics that are conditional on the selected model.

PROC GENSELECT fits models for standard distributions in the exponential family, such as the normal, Poisson, binomial, gamma, lognormal, and Tweedie distributions. In addition, the procedure fits multinomial models for ordinal and nominal response data, and it fits models for the beta, generalized Poisson, and negative binomial distributions.

The GENSELECT procedure can also fit models to correlated responses by using the generalized estimating equations (GEE) method. PROC GENSELECT supports many of the correlation structures from Liang and Zeger (1986).

PROC GENSELECT Features

The GENSELECT procedure does the following:

- estimates the parameters of a generalized linear regression model by using maximum likelihood techniques
- provides model-building syntax in the CLASS statement and the effect-based MODEL statement, which are familiar from SAS/STAT procedures (in particular, the GLM, GENMOD, LOGISTIC, GLIMMIX, and MIXED procedures)

- enables you to split classification effects into individual components by using the SPLIT option in the CLASS statement
- permits any degree of interaction effects that involve classification and continuous variables
- provides multiple link functions
- provides cumulative link models for ordinal response data and generalized logit modeling for unordered multinomial data
- provides response-variable options for binary and multinomial data as in the LOGISTIC procedure
- supports repeated measures analysis by using generalized estimating equations (GEEs) for models that use the normal, gamma, inverse Gaussian, lognormal, negative binomial, binomial, binary, and Poisson distributions
- enables model building (variable selection) through the SELECTION statement
- provides a WEIGHT statement for weighted analysis
- provides a FREQ statement for grouped analysis
- provides a CODE statement to produce SAS code that can score a new data set
- provides an OUTPUT statement to produce a data table that contains predicted probabilities and other observationwise statistics
- uses ODS Graphics to create model selection plots as part of its output. For more information about ODS Graphics, see the section "ODS Graphics" on page 782.

Because the GENSELECT procedure runs on CAS, it also does the following:

- enables you to run on a cluster of machines that distribute the data and the computations
- enables you to run in single-machine mode on CAS
- exploits all the available cores and concurrent threads. For information about how PROC GENSELECT uses threads, see the section "Multithreading" on page 102 in Chapter 2, "Shared Concepts."

PROC GENSELECT Compared with Other SAS Procedures

The GENSELECT procedure provides generalized linear modeling functionality that is comparable to that of the HPGENSELECT and GENMOD procedures in SAS/STAT software.

PROC GENSELECT Compared with the HPGENSELECT Procedure

The functionality of the GENSELECT procedure closely resembles that of the HPGENSELECT procedure, which is a high-performance procedure. The GENSELECT procedure is the next generation of the HPGENSELECT procedure, and it was developed specifically for SAS Viya. Both procedures are designed to run on a cluster of machines that distribute the data and the computations.

Both the GENSELECT and HPGENSELECT procedures fit and perform model selection for generalized linear models. The models can contain main effects that consist of both continuous and classification variables and interaction effects of these variables. The GENSELECT procedure is additionally capable of constructing complex effects, including univariate spline and polynomial expansions. The default parameterization of CLASS variables in both procedures is the GLM parameterization.

If you do not specify a link function with the LINK= option, the GENSELECT procedure uses the logarithm link function as the default for both the gamma and the inverse Gaussian distributions. The HPGENSELECT procedure uses the reciprocal link function as the default for the gamma distribution, and the reciprocal squared as the default for the inverse Gaussian distribution.

With the GENSELECT and HPGENSELECT procedures, you request model selection by using the SELECTION statement. Both procedures offer the same methods of effect selection, but the GENSELECT procedure also produces selection plots by using ODS Graphics.

Both procedures use a modification of the Newton-Raphson algorithm with a ridged Hessian by default. You can choose different optimization techniques, including first-order methods that do not require a crossproducts matrix or Hessian, by using the TECHNIQUE= option. The default method for the Tweedie distribution is a quasi-Newton method.

PROC GENSELECT Compared with the GENMOD Procedure

The GENMOD procedure fits generalized linear models. The GENSELECT procedure fits and performs model selection for generalized linear models. The models can contain main effects that consist of both continuous and classification variables and interaction effects of these variables. The default parameterization of CLASS variables in both procedures is the GLM parameterization.

If you do not specify a link function with the LINK= option, the GENSELECT procedure uses the logarithm link function as the default for both the gamma and the inverse Gaussian distributions. The GENMOD procedure uses the reciprocal link function as the default for the gamma distribution, and the reciprocal squared as the default for the inverse Gaussian distribution.

Both procedures use a modification of the Newton-Raphson algorithm with a ridged Hessian by default. You can instead choose Fisher scoring in PROC GENMOD.

The GENMOD procedure offers a wide variety of postfitting analyses, such as contrasts, estimates, tests of model effects, and least squares means. The GENSELECT procedure is limited in postfitting functionality because it is primarily designed for large-data tasks, such as predictive model building, model fitting, and scoring.

The GENSELECT procedure is specifically designed to operate in SAS Viya, and it performs computations in multiple threads. The GENMOD procedure executes in a single thread on a single machine.

Using CAS Sessions and CAS Engine Librefs

SAS Cloud Analytic Services (CAS) is the analytic server and associated cloud services in SAS Viya. This section describes how to create a CAS session and set up a CAS engine libref that you can use to connect to the CAS session. It assumes that you have a CAS server already available; contact your system administrator if you need help starting and terminating a server. This CAS server is identified by specifying the host on which it runs and the port on which it listens for communications. To simplify your interactions with this CAS server, the host information and port information for the server are stored as SAS option values that are retrieved automatically whenever this CAS server needs to be accessed. You can examine the host and port values for the server at your site by using the following statements:

```
proc options option=(CASHOST CASPORT);
run;
```

In addition to starting a CAS server, your system administrator might also have created a CAS session and a CAS engine libref for your use. You can define your own sessions and CAS engine librefs that connect to the CAS server as shown in the following statements:

```
cas mysess;
libname mylib cas sessref=mysess;
```

The CAS statement creates the CAS session named mysess, and the LIBNAME statement creates the mylib CAS engine libref that you use to connect to this session. It is not necessary to explicitly name the CASHOST and CASPORT of the CAS server in the CAS statement, because these values are retrieved from the corresponding SAS option values.

If you have created the mysess session, you can terminate it by using the TERMINATE option in the CAS statement as follows:

```
cas mysess terminate;
```

For more information about the CAS and LIBNAME statements, see the section "Introduction to Shared Concepts" on page 6 in Chapter 2, "Shared Concepts."

Getting Started: GENSELECT Procedure

Poisson Regression for Count Data

(View the complete code for this example at https://github.com/sascommunities/doc-supplement-casstat/tree/main/Examples/gengs1.sas.)

NOTE: Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections "Using CAS Sessions and CAS Engine Librefs" on page 7 and "Loading a SAS Data Set onto a CAS Server" on page 7 in Chapter 2, "Shared Concepts."

This example illustrates how you can use PROC GENSELECT to perform Poisson regression for count data. The following DATA step creates the data table getStarted, which consists of 100 observations on a count response variable (Y); a continuous variable (Total) to be used in a later analysis; and five categorical variables (C1–C5), each of which has four numerical levels:

```
data mylib.getStarted;
  input C1-C5 Y Total;
  datalines;
0 3 1 1 3 2 28.361
2 3 0 3 1 2 39.831
1 3 2 2 2 1 17.133
1 2 0 0 3 2 12.769
0 2 1 0 1 1 29.464
0 2 1 0 2 1 4.152
1 2 1 0 1 0 0.000
0 2 1 1 2 1 20.199
1 2 0 0 1 0 0.000
0 1 1 3 3 2 53.376
2 2 2 2 1 1 31.923
0 3 2 0 3 2 37.987
2 2 2 0 0 1 1.082
0 2 0 2 0 1 6.323
1 3 0 0 0 0 0.000
1 2 1 2 3 2 4.217
0 1 2 3 1 1 26.084
1 1 0 0 1 0 0.000
1 3 2 2 2 0 0.000
2 1 3 1 1 2 52.640
1 3 0 1 2 1 3.257
2 0 2 3 0 5 88.066
2 2 2 1 0 1 15.196
3 1 3 1 0 1 11.955
3 1 3 1 2 3 91.790
3 1 1 2 3 7 232.417
3 1 1 1 0 1 2.124
3 1 0 0 0 2 32.762
3 1 2 3 0 1 25.415
2 2 0 1 2 1 42.753
```

```
3 3 2 2 3 1 23.854
2 0 0 2 3 2 49.438
1 0 0 2 3 4 105.449
0 0 2 3 0 6 101.536
0 3 1 0 0 0 0.000
3 0 1 0 1 1 5.937
2 0 0 0 3 2 53.952
1 0 1 0 3 2 23.686
1 1 3 1 1 1 0.287
2 1 3 0 3 7 281.551
1 3 2 1 1 0 0.000
2 1 0 0 1 0 0.000
0 0 1 1 2 3 93.009
0 1 0 1 0 2 25.055
1 2 2 2 3 1 1.691
0 3 2 3 1 1 10.719
3 3 0 3 3 1 19.279
2 0 0 2 1 2 40.802
2 2 3 0 3 3 72.924
0 2 0 3 0 1 10.216
3 0 1 2 2 2 87.773
2 1 2 3 1 0 0.000
3 2 0 3 1 0 0.000
3 0 3 0 0 2 62.016
1 3 2 2 1 3 36.355
2 3 2 0 3 1 23.190
1 0 1 2 1 1 11.784
2 1 2 2 2 5 204.527
3 0 1 1 2 5 115.937
0 1 1 3 2 1 44.028
2 2 1 3 1 4 52.247
1 1 0 0 1 1 17.621
3 3 1 2 1 2 10.706
2 2 0 2 3 3 81.506
0 1 0 0 2 2 81.835
0 1 2 0 1 2 20.647
3 2 2 2 0 1 3.110
2 2 3 0 0 1 13.679
1 2 2 3 2 1 6.486
3 3 2 2 1 2 30.025
0 0 3 1 3 6 202.172
3 2 3 1 2 3 44.221
0 3 0 0 0 1 27.645
3 3 3 0 3 2 22.470
2 3 2 0 2 0 0.000
1 3 0 2 0 1 1.628
1 3 1 0 2 0 0.000
3 2 3 3 0 1 20.684
3 1 0 2 0 4 108.000
0 1 2 2 1 1 4.615
0 2 3 2 2 1 12.461
0 3 2 0 1 3 53.798
2 1 1 2 0 1 36.320
1 0 3 0 0 0 0.000
```

```
0 0 3 2 0 1 19.902
0 2 3 1 0 0 0.000
2 2 2 1 3 2 31.815
3 3 3 0 0 0 0.000
2 2 1 3 3 2 17.915
0 2 3 2 3 2 69.315
1 3 1 2 1 0 0.000
3 0 1 1 1 4 94.050
2 1 1 1 3 6 242.266
0 2 0 3 2 1 40.885
2 0 1 1 2 2 74.708
2 2 2 2 3 2 50.734
1 0 2 2 1 3 35.950
1 3 3 1 1 1 2.777
3 1 2 1 3 5 118.065
0 3 2 1 2 0 0.000
;
```

These statements assume that your libref is named mylib, but you can substitute any appropriately defined libref.

The following statements fit a log-linked Poisson model to these data by using classification effects for the variables C1–C5:

```
proc genselect data=mylib.getStarted;
   class C1-C5;
   model Y = C1-C5 /  Distribution=Poisson Link=Log;
run;
```

The output from this analysis is presented in Figure 14.1 through Figure 14.7.

Figure 14.1 displays the "Model Information" table. The variable Y is an integer-valued variable that is modeled by using a Poisson probability distribution, and the mean of Y is modeled by using a log link function. The GENSELECT procedure uses a Newton-Raphson algorithm to fit the model. The CLASS variables C1–C5 are parameterized by using GLM parameterization, which is the default.

Figure 14.1 Model Information

The GENSELECT Procedure

Model Information	
Data Source	GETSTARTED
Response Variable	Y
Distribution	Poisson
Link Function	Log
Optimization Technique	Newton-Raphson with Ridging

Each of the CLASS variables C1–C5 has four unique formatted levels, which are displayed in the "Class Level Information" table in Figure 14.2.

Figure 14.2 Class Level Information

Class Level Information		
Class	**Levels**	**Values**
C1	4	0 1 2 3
C2	4	0 1 2 3
C3	4	0 1 2 3
C4	4	0 1 2 3
C5	4	0 1 2 3

Figure 14.3 displays the "Number of Observations" table. All 100 observations in the data set are used in the analysis.

Figure 14.3 Number of Observations

Number of Observations Read	100
Number of Observations Used	100

Figure 14.4 displays the "Dimensions" table for this model. This table summarizes some important sizes of various model components. For example, it shows that there are 21 columns in the design matrix $\mathbf{X}$: 1 column for the intercept and 20 columns for the effects that are associated with the classification variables C1–C5. However, the rank of the crossproducts matrix is only 16. Because the classification variables C1–C5 use GLM parameterization and because the model contains an intercept, there is one singularity in the crossproducts matrix of the model for each classification variable. Consequently, only 16 parameters enter the optimization.

Figure 14.4 Dimensions in Poisson Regression

Dimensions	
Columns in Design	21
Number of Effects	6
Max Effect Columns	4
Rank of Design	16
Parameters in Optimization	16

Figure 14.5 displays the final convergence status of the Newton-Raphson algorithm. The FCONV= convergence criterion is satisfied.

Figure 14.5 Convergence Status

Convergence criterion (FCONV=1E-7) satisfied.

The "Fit Statistics" table is shown in Figure 14.6. The –2 log likelihood at the converged estimates is 290.16169. You can use this value to compare the model to nested model alternatives by means of a likelihood ratio test. To compare models that are not nested, information criteria such as AIC (Akaike's information criterion), AICC (Akaike's bias-corrected information criterion), and BIC (Schwarz Bayesian information criterion) are used. These criteria penalize the –2 log likelihood for the number of parameters.

Figure 14.6 Fit Statistics

Fit Statistics	
-2 Log Likelihood	290.16169
AIC (smaller is better)	322.16169
AICC (smaller is better)	328.71590
SBC (smaller is better)	363.84441

The "Parameter Estimates" table in Figure 14.7 shows that many parameters have fairly large *p*-values, indicating that one or more of the model effects might not be necessary.

Figure 14.7 Parameter Estimates

Parameter Estimates

Parameter	DF	Estimate	Standard Error	Chi-Square	Pr > ChiSq
Intercept	1	0.881904	0.382730	5.3095	0.0212
C1 0	1	-0.196002	0.211482	0.8590	0.3540
C1 1	1	-0.605161	0.263508	5.2742	0.0216
C1 2	1	-0.068458	0.210776	0.1055	0.7453
C1 3	0	0	.	.	.
C2 0	1	0.961117	0.255485	14.1521	0.0002
C2 1	1	0.708188	0.246768	8.2360	0.0041
C2 2	1	0.161741	0.266365	0.3687	0.5437
C2 3	0	0	.	.	.
C3 0	1	-0.227016	0.252561	0.8079	0.3687
C3 1	1	-0.094775	0.229519	0.1705	0.6797
C3 2	1	0.044801	0.238127	0.0354	0.8508
C3 3	0	0	.	.	.
C4 0	1	-0.280476	0.263589	1.1322	0.2873
C4 1	1	0.028157	0.249652	0.0127	0.9102
C4 2	1	0.047803	0.240378	0.0395	0.8424
C4 3	0	0	.	.	.
C5 0	1	-0.817936	0.219901	13.8351	0.0002
C5 1	1	-0.710595	0.206265	11.8684	0.0006
C5 2	1	-0.602080	0.217724	7.6471	0.0057
C5 3	0	0	.	.	.

Syntax: GENSELECT Procedure

The following statements are available in the GENSELECT procedure:

PROC GENSELECT < *options* > ;
BY *variables* ;
CLASS *variable* < **(***options***)** >. . . < *variable* < **(***options***)** > > < / *global-options* > ;
CODE < *options* > ;
DISPLAY < *table-list* > < / *options* > ;
DISPLAYOUT *table-spec-list* < / *options* > ;
EFFECT *name***=***effect-type***(***variables* < / *options* >**)** ;
FREQ *variable* ;
MODEL *response*< **(***response-options***)** > **=** < *effects* > < / *model-options* > ;
MODEL *events/trials*< **(***response-options***)** > **=** < *effects* > < / *model-options* > ;
LSMEANS < *model-effects* > < / *options* > ;
OUTPUT OUT=*libref.data-table* < *keyword* < **=***name* > >. . . < *keyword* < **=***name* > > < *options* > ;
PARTITION < *partition-options* > ;
REPEATED SUBJECT=*subject-effect* < / *repeated-options* > ;
RESTRICT *constraint-specification* < / *restriction-options* > ;
SELECTION < **METHOD=***method* < *(method-options)* > >< *options* > ;
STORE < **OUT=** >*libref.data-table* ;
WEIGHT *variable* < / **NORMALIZE** > ;

The PROC GENSELECT statement and a MODEL statement are required. The CLASS statement can appear multiple times. If a CLASS statement is specified, it must precede the MODEL statement.

PROC GENSELECT Statement

PROC GENSELECT < *options* > ;

The PROC GENSELECT statement invokes the procedure. Table 14.1 summarizes the available *options* in the PROC GENSELECT statement by function. They are then described fully in alphabetical order.

Table 14.1 PROC GENSELECT Statement Options

Option	Description
Basic Options	
ALPHA=	Specifies a global significance level
APPLYROWORDER	Uses group and order information from the input data table
DATA=	Specifies the input data table
FITDATA	Declares the DATA= table to be the same input data table used to build the model
INPARMEST=	Specifies starting values for the optimization
MAXRESPONSELEVELS=	Specifies the maximum number of response levels allowed when you have a polytomous response variable
PAGEOBS=	Specifies the maximum number of observations to be computed in each batch

Table 14.1 *continued*

Option	Description
RESTORE=	Specifies the input analytic store
SEED=	Specifies the random number seed for LS-means and partition fractions
Options Related to Output	
CLB	Constructs confidence limits for the parameter estimates
CORRB	Displays the "Parameter Estimates Correlation Matrix" table
COVB	Displays the "Parameter Estimates Covariance Matrix" table
ITHIST	Displays the "Iteration History" table
NOCHECK	Disables checking for infinite parameters
NOCLPRINT	Limits or suppresses the display of classification variable levels
NOSTDERR	Suppresses computation of the covariance matrix and standard errors
NOXPX	Suppresses computation of the Hessian matrices
PARTFIT	Displays the fit statistics that are produced when your data are partitioned
STB	Displays standardized estimates
TYPE3	Produces the Type 3 or joint tests of effects
USELASTITER	Continues computations when the optimization fails
Options Related to Optimization	
ABSCONV=	Tunes the absolute function convergence criterion
ABSFCONV=	Tunes the absolute function difference convergence criterion
ABSGCONV=	Tunes the absolute gradient convergence criterion
ABSXCONV=	Tunes the absolute parameter convergence criterion
FCONV=	Tunes the relative function difference convergence criterion
FCONV2=	Tunes the second relative function difference convergence criterion
GCONV=	Tunes the relative gradient convergence criterion
GCONV2=	Tunes the second relative gradient convergence criterion
XCONV=	Tunes the relative gradient convergence criterion
MAXFUNC=	Specifies the maximum number of function evaluations in any optimization
MAXITER=	Chooses the maximum number of iterations in any optimization
MAXTIME=	Specifies the upper limit of CPU time (in seconds) for any optimization
MINITER=	Specifies the minimum number of iterations in any optimization
NORMALIZE=	Specifies whether the objective function is normalized during optimization
TECHNIQUE=	Selects the optimization technique
LASSO Options	
LASSORHO=	Specifies the base regularization parameter for the LASSO method
LASSOSTEPS=	Specifies the maximum number of steps for the LASSO method
LASSOTOL=	Specifies the convergence criterion for the LASSO method

The optimization options are fully described in the section "Optimization Options" on page 57 in Chapter 2, "Shared Concepts." The following list describes the other options available in the PROC GENSELECT statement:

ALPHA=*number*
specifies a global significance level for the construction of confidence intervals. The confidence level is 1 – *number*. The value of *number* must be between 0 and 1. You can override this global significance level by specifying this option in the OUTPUT statement. By default, ALPHA=0.05.

APPLYROWORDER
uses group and order information from the input data table. You can add this information to the input table by using the **`partition`** action in the **`table`** action set. For more information, see the section "The APPLYROWORDER Option" on page 107 in Chapter 2, "Shared Concepts." By default, the APPLYROWORDER option is not enabled. If you specify this option but the table does not contain group and order information, the procedure terminates with an error.

NOTE: You can use group and order information for a repeated measures analysis if the group information meets certain criteria. For more information, see the section "Using a Preexisting Table Order" on page 767.

CLB
constructs confidence limits for each of the parameter estimates. The confidence level is 0.95 by default; you can change it by specifying the ALPHA= option. This option is not available when you use either LASSO selection or elastic net selection.

CORRB
creates the "Parameter Estimates Correlation Matrix" table. The correlation matrix is computed by normalizing the covariance matrix $\mathbf{\Sigma}$. That is, if σ_{ij} is an element of $\mathbf{\Sigma}$, then the corresponding element of the correlation matrix is $\sigma_{ij}/\sigma_i\sigma_j$, where $\sigma_i = \sqrt{\sigma_{ii}}$. This option is not available when you use either LASSO selection or elastic net selection.

COVB
creates the "Parameter Estimates Covariance Matrix" table. The covariance matrix is computed as the inverse of the negative of the matrix of second derivatives of the log-likelihood function with respect to the model parameters (the Hessian matrix). This option is not available when you use either LASSO selection or elastic net selection.

DATA=*libref.data-table*
names the input data table for PROC GENSELECT to use. The default is the most recently created data table. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to the data, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 707.

data-table specifies the name of the input data table.

FITDATA
declares the DATA= table to be the same input data table that is used for building the model, when you also specify the RESTORE= option. The FITDATA option enables you to specify the PARTFIT option to produce more fit statistics, to compute all the statistics from the OUTPUT statement that are available for your model, and to group computations according to the partition roles.

INPARMEST=*libref.data-table*
names an input data table that contains starting values. *libref.data-table* is a two-level name, where *libref* refers to the library, and *data-table* specifies the name of the output data table. For more information about this two-level name, see the DATA= option and the section "Using CAS Sessions and CAS Engine Librefs" on page 707.

The input data table must contain an Estimate column and a ParmName column, and it can optionally contain your BY variables. You can create a version of this table by using a DISPLAYOUT statement to produce the "Parameter Estimates" table from a PROC GENSELECT program. For example:

```
displayout parameterestimates=pe;
```

You can then modify the parameter estimates in this table and read them back in by using the following statement:

```
proc genselect inparmest=mylib.pe;
```

ITHIST
generates the "Iteration History" table.

LASSORHO=*r*
specifies the base regularization parameter for the LASSO model selection method. The regularization parameter for step i is r^i. By default, LASSORHO=0.8.

LASSOSTEPS=*n*
specifies the maximum number of steps for LASSO model selection. By default, LASSOSTEPS=20.

LASSOTOL=*r*
specifies the convergence tolerance for the optimization algorithm that solves for the LASSO parameter estimates at each step of LASSO model selection. By default, LASSOTOL=1E–6.

MAXRESPONSELEVELS=*value*
specifies the maximum number of response levels that are allowed when you have a polytomous response variable. By default, MAXRESPONSELEVELS=100.

NOCHECK
disables the checking process that determines whether maximum likelihood estimates of the regression parameters exist. For more information, see the section "Existence of Maximum Likelihood Estimates" on page 759.

NOCLPRINT<=*number*>
suppresses the display of the "Class Level Information" table if you do not specify *number*. If you specify *number*, the values of the classification variables are displayed for only those variables whose number of levels is less than *number*. Specifying *number* helps to reduce the size of the "Class Level Information" table if some classification variables have a large number of levels.

NOSTDERR
suppresses computation of the covariance matrix and the standard errors of the regression coefficients. When the model contains many variables (thousands), the inversion of the Hessian matrix to derive the covariance matrix and the standard errors of the regression coefficients can be time-consuming. The CORRB, COVB, and TYPE3 options are not available when the NOSTDERR option is specified. This option also disables the quasi-complete separation check; for more information, see the section "Existence of Maximum Likelihood Estimates" on page 759.

NOXPX
suppresses Hessian and $\mathbf{X}'\mathbf{X}$ computations and invokes the NOSTDERR option. When the model contains many variables (thousands), computing the Hessian matrix during model fitting can be time-consuming. The CLB, CORRB, COVB, and TYPE3 options, the REPEATED statement, the OUTPUT statement options that rely on the covariance, and confidence limits for the parameters and for the OUTPUT and LSMEANS statements are not available when you specify the NOXPX option. The forward, backward, and stepwise selection methods are not available; however, you can specify the LASSO and elastic net methods. This option invokes the TECHNIQUE=LBFGS option by default, and the NEWRAP, NRRIDG, and TRUREG optimization techniques are not available. Because the method of identifying linearly dependent variables relies on the $\mathbf{X}'\mathbf{X}$ matrix, some parameters that should be assigned 0 degrees of freedom can be missed.

PAGEOBS=*number* | **AUTO**

MAXOPTBATCH=*number* | **AUTO**
specifies the maximum number of observations to be included in a batch. During the optimization, the GENSELECT procedure reads at most *number* observations from the data table into memory; performs the appropriate log-likelihood, gradient, and Hessian computations on that batch of observations; then discards those observations and reads in the next batch of data for processing. Generally, a smaller *number* decreases memory usage but might lead to longer computation times, whereas a larger *number* might lead to shorter computation times but increases memory usage. The default PAGEOBS=AUTO option determines whether the entire data table can be held in a subset of your available memory; if it cannot, then *number* is set to 256.

PARTFIT
displays fit statistics in the "Fit Statistics" table that are usually produced when your data are partitioned. This option is not required when you specify a PARTITION statement. The statistic that is added to the table is the average square error (or Brier score).

RESTORE=*libref.data-table*
specifies the name of the analytic store that contains the context and results of a model that is selected and stored from a previous statistical analysis. The analytic store is created by a STORE statement from a previous PROC GENSELECT call or by a **`store`** parameter that is specified in a previous **`genmod`** action call.

The displayed output can include the following content:

- information about the analytic store
- notes that are stored by the TEXT= option of the STORE statement
- a Replay section, consisting of tables that are produced when the stored model is fit
- a Restore section, consisting of tables that are produced by applying the stored model to a specified input data table

You can specify the STB and CLB options in the PROC GENSELECT statement to modify the "Parameter Estimates" table; you can specify the CORRB, COVB, and TYPE3 options to display those tables; you can specify the CODE statement to produce SAS code for scoring; and you can specify LSMEANS statements to compute least squares means. If you specify a DATA= data table, then you can also specify the PARTFIT option, and you can score that data table by specifying the OUTPUT statement.

If the analytic store is created using BY-group processing, then the results are displayed according to each BY group. If you specify a DATA= input data table, then you must also specify the same BY statement that was used to create the store.

If a PARTITION statement is used to create the stored model, then the Replay section displays only information that is computed from the original training data. For the Restore section, if you create the partition for the stored model by using a ROLE variable and also specify the FITDATA option, then the ROLE variable must exist in the DATA= table and the Restore section displays results according to the roles. However, if the partitions for the stored model are created using a FRACTION option, then the roles are ignored because the roles cannot be reliably re-created; this means that output statistics that rely on the input data are not computed.

Because the selected model from a previous analysis is stored, statements and options specific to model specification, selection, and optimization are not available. In particular, the CLASS, EFFECT, FREQ, MODEL, PARTITION, REPEATED, SELECTION, STORE, and WEIGHT statements are not available to be used with the RESTORE= option.

libref.data-table is a two-level name, where *libref* refers to the library, and *data-table* specifies the name of the input data table. For more information about this two-level name, see the DATA= option and the section "Using CAS Sessions and CAS Engine Librefs" on page 707.

SEED=*number*

specifies an integer to be used to start the pseudorandom number generator for the LSMEANS statement's ADJUST=SIMULATE option and for the PARTITION statement's FRACTION option. If you do not specify a seed, or if you specify a *number* less than or equal to 0, the seed is generated by reading the time of day from the computer's clock.

STB

displays the standardized estimates of the parameters in the "Parameter Estimates" table. The standardized estimate of β_i is given by $\widehat{\beta}_i/(s/s_i)$, where s_i is the total sample standard deviation for the ith explanatory variable and

$$s = \begin{cases} \pi/\sqrt{3} & \text{LOGIT and GLOGIT links} \\ \pi/\sqrt{6} & \text{CLOGLOG and LOGLOG links} \\ 1 & \text{all other links} \end{cases}$$

The sample standard deviations for parameters that are associated with CLASS variables are computed using their codings. The standardized estimates are not computed for the intercept parameters.

TYPE3
computes Wald statistics for Type 3 contrasts for each effect that you specify in the MODEL statement. This option is not available for models that you fit by using the GEE method. This option is also not available when you use either LASSO selection or elastic net selection. For more information, see the section "Joint Tests and Type 3 Tests" on page 772.

USELASTITER
continues to perform computations by using the last iteration of the optimization when the optimization fails. By default, computations cease when a failure occurs.

BY Statement

BY *variables* **;**

You can specify a BY statement in PROC GENSELECT to obtain separate analyses of observations in groups that are defined by the values of the BY variables. If you specify more than one BY statement, only the last one specified is used. For more information, see the discussion of BY-group processing in the "Grouping Data" section of *SAS Programmers Guide: Essentials*.

CLASS Statement

CLASS *variable* <**(***options***)**>. . . <*variable* <**(***options***)**>> </ *global-options*> **;**

The CLASS statement names the classification variables to be used as explanatory variables in the analysis. You can list the response variable for binary models in the CLASS statement, but this is not required. Table 14.2 summarizes the values that you can use for either an *option* or a *global-option*. The options are fully documented in the section "CLASS Statement" on page 9 in Chapter 2, "Shared Concepts."

Table 14.2 CLASS Statement Options

Option	Description
DESCENDING	Reverses the sort order
MISSING	Treats missing values as valid levels
ORDER=	Specifies the sort order for the levels
PARAM=	Specifies the parameterization of the variable
REF=	Specifies the reference level of the variable
SPLIT	Splits levels of CLASS variables into independent effects

CODE Statement

CODE < *options* > **;**

The CODE statement writes SAS DATA step code for computing predicted values of the fitted model to a file or to a table. To score new data, you can then include the file in a DATA step, or you can specify the table in the **`runCodeTable`** action in the **`dataStep`** action set (for more information, see *SAS Viya: System Programming Guide*).

Table 14.3 summarizes the *options* available in the CODE statement.

Table 14.3 CODE Statement Options

Option	Description
COMMENT	Adds comments to the generated code
FILE=	Names the file in which to save the generated code
FORMATWIDTH=	Specifies the numeric format width for the regression coefficients
INDENTSIZE=	Specifies the number of spaces to indent the generated code
LABELID=	Specifies a number used to construct names and labels
LINESIZE=	Specifies the line size for the generated code
NOTRIM	Compares formatted values, including blank padding
OUT=	Names an output table in which to save the generated code

For more information about the syntax of the CODE statement, see the section "CODE Statement" on page 13 in Chapter 2, "Shared Concepts."

DISPLAY Statement

DISPLAY < *table-list* > < / *options* > **;**

The DISPLAY statement enables you to specify a list of display tables to display or exclude. This statement is similar to the ODS SELECT, ODS EXCLUDE, and ODS TRACE statements. However, the DISPLAY statement can improve performance when a large number of tables could be generated (such as in BY-group processing). The procedure processes the DISPLAY statement on a CAS server and thus sends only a subset of ODS tables to the SAS client. Because ODS statements are processed on a SAS client, first all the generated display tables are sent to the client, and then the client creates a subset.

If you use both DISPLAY and ODS statements together, the DISPLAY statement takes precedence over the ODS statements. Note that the ODS EXCLUDE statement processes tables that are sent to the client after they have been filtered by the DISPLAY statement. In some cases, it might appear that the ODS EXCLUDE statement is taking precedence because it can further filter the tables. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

You can specify the *table-list* as a list of table names, paths, partial pathnames, and regular expressions.

The table names that you can specify are listed in the section "ODS Table Names" on page 780. A path is a table name that is prefixed with dot-separated grouping information. For example, a

SelectionSummary table that a procedure produces during a selection routine might have the path *Bygroup1.Summary.SelectionSummary*. A partial pathname does not include all groups; for example, *SelectionSummary* and *Summary.SelectionSummary* are partial pathnames for *Bygroup1.Summary.SelectionSummary*.

When you specify a table name or partial pathname, all display tables whose paths end in the specified name are selected for display or exclusion. For example, both *SelectionSummary* and *Summary.SelectionSummary* select *Bygroup1.Summary.SelectionSummary*.

A regular expression is enclosed in forward slashes (/). For example, specifying "/tions/" selects all pathnames that contain the substring "tions"; in particular, the *Bygroup1.Summary.SelectionSummary* table is selected. Specifying "!/tions/" selects all pathnames that do not contain the substring "tions"; in particular, the *Bygroup1.Summary.SelectionSummary* table is not selected.

You can specify the following *options* after a slash (/):

CASESENSITIVE
performs a case-sensitive comparison of table names in the *table-list* to display table names when tables are subsetted for display. To preserve case, you must enclose table names in the *table-list* in quotation marks.

EXCLUDE
displays all display tables except those that you specify in the *table-list*.

EXCLUDEALL
suppresses display of all tables. This option takes precedence over the other options.

TRACE
displays the display table names, labels, and paths.

DISPLAYOUT Statement

DISPLAYOUT *table-spec-list* < / *options* > **;**

The DISPLAYOUT statement enables you to create output tables from your displayed output. This statement is similar to the ODS OUTPUT statement. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

The *table-spec-list* specifies a list of output tables to create. Each entry in the list has either a *key=value* format or a *key* format:

key=value specifies *key* as the ODS table name, path, or partial pathname, and specifies *value* as the output table name.

key specifies *key* as the ODS table name and also as the output table name.

The ODS table names that you can specify are listed in the section "ODS Table Names" on page 780. You cannot specify the ODS table named OutputCasTables in the *table-spec-list*.

Table names and partial pathnames are discussed under the DISPLAY statement. The DISPLAYOUT statement does not support regular expressions.

You can specify the following *options* after a slash (/):

INCLUDEALL
creates output tables for all display tables. The name of the created output table is the same as the name of the corresponding display table. If you specify this option, the *table-spec-list* specification is ignored.

NOREPLACE
does not replace any existing output table of the same name.

REPEATED
replicates all output tables on all nodes.

EFFECT Statement

EFFECT *name*=*effect-type* **(***variables* < / *options* >**) ;**

The EFFECT statement enables you to construct special collections of columns for design matrices. These collections are referred to as *constructed effects* to distinguish them from the usual model effects that are formed from continuous or classification variables, as discussed in the section "GLM Parameterization of Classification Variables and Effects" on page 67 in Chapter 2, "Shared Concepts."

You can specify the following *effect-types*:

COLLECTION specifies a collection effect that defines one or more variables as a single effect that has multiple degrees of freedom. The variables in a collection are considered as a unit for purposes of estimation and inference.

MULTIMEMBER | MM specifies a multimember classification effect whose levels are determined by one or more variables that appear in a CLASS statement.

POLYNOMIAL | POLY specifies a multivariate polynomial effect in the specified numeric variables.

SPLINE specifies a regression spline effect whose columns are univariate spline expansions of one or more variables. A spline expansion replaces the original variable with an expanded or larger set of new variables.

Table 14.4 summarizes the *options* available in the EFFECT statement.

Table 14.4 EFFECT Statement Options

Option	Description
Collection Effects Options	
DETAILS	Displays the constituents of the collection effect

Table 14.4 *continued*

Option	Description
Multimember Effects Options	
DETAILS	Displays the levels of the multimember effect
NOEFFECT	Specifies that observations whose levels are all missing for the multimember variables should have 0 values in the corresponding design matrix columns
STDIZE	Standardizes the design matrix entries so that each observation has a sum of 1
WEIGHT=	Specifies the weight variable for the contributions of each classification effect
Polynomial Effects Options	
DEGREE=	Specifies the degree of the polynomial
DETAILS	Displays details of the specified polynomial
MDEGREE=	Specifies the maximum degree of any variable in a term of the polynomial
NOSEPARATE	Treats the polynomial as a single effect with multiple degrees of freedom
STANDARDIZE=	Specifies centering and scaling suboptions for the variables that define the polynomial
Spline Effects Options	
BASIS=	Specifies the type of basis (B-spline basis or truncated power function basis) for the spline effect
DATABOUNDARY	Uses the extremes of the data as boundary knots for a B-spline basis
DEGREE=	Specifies the degree of the spline effect
DETAILS	Displays the knots and locations for each spline basis function
KNOTMAX=	Requests equally spaced right-side boundary knots starting at the variables' maximum and ending at the KNOTMAX= value
KNOTMETHOD=	Specifies how to construct the knots for the spline effect
KNOTMIN=	Requests equally spaced left-side boundary knots starting at the KNOTMIN= value and ending at the variables' minimum value
NATURALCUBIC	Specifies a natural cubic spline basis for the spline effect
SEPARATE	Treats the spline basis for each variable as a separate effect when multiple variables are specified
SPLIT	Treats each design matrix column as a separate effect for selection methods

For more information about the syntax of these *effect-types* and how columns of constructed effects are computed, see the section "EFFECT Statement" on page 18 in Chapter 2, "Shared Concepts."

FREQ Statement

FREQ *variable* **;**

The *variable* in the FREQ statement identifies a numeric variable in the input data table that contains the frequency of occurrence of each observation. PROC GENSELECT treats each observation as if it appears f times, where f is the value of the FREQ *variable* for the observation. If f is not an integer, it is truncated to an integer. If f is less than 1 or missing, the observation is not used in the analysis. When you omit the FREQ statement, each observation is assigned a frequency of 1.

In the case of models that are fit using generalized estimating equations (GEEs), the frequencies apply to the subject, so the frequency must be the same for all observations within each subject.

LSMEANS Statement

LSMEANS *model-effects* < / *options* > **;**

LSMEANS EFFECT=*model-effects* < / *options* > **;**

The LSMEANS statement computes and compares least squares means (LS-means) of fixed effects. LS-means are *predicted population margins*—that is, they estimate the marginal means over a balanced population. In a sense, LS-means are to unbalanced designs as class and subclass arithmetic means are to balanced designs.

LS-means can be computed for any effect in the procedure's MODEL statement that involves only classification variables. You can specify multiple *model-effects* in one or more LSMEANS statements that appear after the MODEL statement. When you restore a model from an item store, you must specify the desired *model-effects* in the EFFECT= option before the slash (/).

Table 14.5 summarizes the *options* available in the LSMEANS statement.

Table 14.5 LSMEANS Statement Options

Option	Description
Construction and Computation of LS-Means	
AT	Modifies covariate values used to compute LS-means
ATVAR	Modifies covariate values for restored models
DIFF	Computes differences of LS-means
SINGULAR=	Tunes estimability checking
***p*-Values**	
ADJUST=	Specifies the multiple comparison adjustment method for LS-means differences
ALPHA=α	Specifies the confidence level ($1-\alpha$)
Statistical Output	
CL	Constructs confidence limits for means and mean differences
CORR	Displays the correlation matrix of LS-means
COV	Displays the covariance matrix of LS-means
E	Displays the $\mathbf{L}$ matrix

For more information about the syntax of the LSMEANS statement, see the section "LSMEANS Statement" on page 34 in Chapter 2, "Shared Concepts."

LS-means analysis is not available if you use LASSO or elastic net selection.

MODEL Statement

MODEL *response* < **(***response-options***)** > **=** < *effects* > < / *model-options* > **;**

MODEL *events* / *trials* < **(***response-options***)** > **=** < *effects* > < / *model-options* > **;**

The MODEL statement defines the statistical model in terms of a *response* variable (the target) or an *events/trials* specification, model *effects* that are constructed from variables in the input data table, and *model-options*. An intercept is included in the model by default. You can remove the intercept by specifying the NOINT option.

You can specify a single *response* variable that contains your response values. When you have binomial data, you can specify the *events/trials* form of the response, where one variable contains the number of positive responses (or events) and another variable contains the number of trials. Note that the values of both *events* and (*trials* – *events*) must be nonnegative and the value of *trials* must be positive.

For information about constructing the model effects, see the section "Specification and Parameterization of Model Effects" on page 64 in Chapter 2, "Shared Concepts."

There are two sets of options in the MODEL statement. The *response-options* determine how the GENSELECT procedure models probabilities for binary and multinomial data. The *model-options* control other aspects of model formation and inference. Table 14.6 summarizes these options.

Table 14.6 MODEL Statement Options

Option	Description
Response Variable Options	
DESCENDING	Reverses the response categories
EVENT=	Specifies the event category
ORDER=	Specifies the sort order
REF=	Specifies the reference category
Model Options	
CENTER	Centers and scales continuous main effects
CENTERLASSO	Centers and scales all effects for model selection by the LASSO method and the elastic net method
CLB	Requests confidence limits
DISTRIBUTION=	Specifies the response distribution
INCLUDE=	Includes effects in all models for model selection
INFORMATIVE	Models missing values by using extra indicator variables
LINK=	Specifies the link function
NOINT	Suppresses the intercept
OFFSET=	Specifies the offset variable
PHI=	Specifies a fixed dispersion parameter

Table 14.6 *continued*

Option	Description
START=	Includes effects in the initial model for model selection
TYPE3	Displays the Type 3 or joint tests of effects

Response Variable Options

Response variable options determine how the GENSELECT procedure models probabilities for binary and multinomial data.

You can specify the following *response-options* by enclosing them in parentheses after the *response* or *trials* variable.

DESCENDING

DESC

reverses the order of the response categories. If you specify both the DESCENDING and ORDER= options, PROC GENSELECT orders the response categories according to the ORDER= option and then reverses that order.

EVENT='*category***' | FIRST | LAST**

specifies the event category for the binary and multinomial response model. PROC GENSELECT models the probability of the event category. The EVENT= option has no effect when there are more than two response categories.

You can specify one of the following:

'*category*'

specifies the value (formatted, if a format is applied) of the event category in quotation marks.

FIRST

designates the first ordered category as the event.

LAST

designates the last ordered category as the event.

By default, EVENT=FIRST.

For example, the following statements specify that observations whose formatted value is 1 represent events in the data. The probability that PROC GENSELECT models is thus the probability that the variable def takes the (formatted) value 1.

```
proc genselect data=mylib.MyData;
   class A B C;
   model def(event ='1') = A B C x1 x2 x3;
run;
```

ORDER=FORMATTED | FREQ | INTERNAL
specifies the sort order for the levels of the *response* variable. You can specify the following values:

FORMATTED sorts the levels by external formatted value, except for numeric variables that have no explicit format, which are sorted by their unformatted (internal) value. For numeric variables for which you have supplied no explicit format (that is, for which there is no corresponding FORMAT statement in the current PROC GENSELECT run or in the DATA step that created the data table), the levels are ordered by their internal (numeric) value. The sort order is machine-dependent.

FREQ sorts the levels by descending frequency count (levels that have the most observations come first in the order).

INTERNAL sorts the levels by unformatted value. The sort order is machine-dependent.

By default, ORDER=FORMATTED.

For more information about sort order, see the chapter on the SORT procedure in the *Base SAS Procedures Guide* and the discussion of BY-group processing in *SAS Programmers Guide: Essentials*.

REF='*category***' | FIRST | LAST**
specifies the reference category for the generalized logit model and the binary response model. For the generalized logit model, each logit contrasts a nonreference category with the reference category. For the binary response model, specifying one response category as the reference is the same as specifying the other response category as the event. You can specify one of the following:

'*category*'
specifies the value (formatted, if a format is applied) of the reference category in quotation marks.

FIRST
designates the first ordered category as the reference.

LAST
designates the last ordered category as the reference.

By default, REF=LAST.

Model Options

You can specify the following *model-options* after a slash (/):

CENTER
centers and scales continuous main effects for internal computation. (Continuous main effects are centered and scaled to aid in computing maximum likelihood estimates.) Parameter estimates and related statistics are always reported on the original scale. This option has no effect if you specify the LASSO model selection method or the elastic net selection method; instead, you should specify the CENTERLASSO option.

CENTERLASSO< =TRUE | FALSE >
specifies whether all effects, including categorical effects, are centered and scaled internally for the LASSO model selection method or the elastic net selection method. (Effects are centered and scaled to aid in model selection.) Parameter estimates and related statistics are always reported on the original scale. By default, CENTERLASSO=TRUE.

CLB
constructs confidence limits for each parameter estimate. The confidence level is 0.95 by default; you can change it by specifying the ALPHA= option. The CLB option is not available when you use either LASSO selection or elastic net selection.

DISTRIBUTION=*keyword*
specifies the response distribution for the model. The *keywords* and the associated distributions are shown in Table 14.7. For information about default and commonly used link functions for each distribution function, see Table 14.9.

Table 14.7 Built-In Distribution Functions

Keyword	Distribution Function
BETA	Beta
BINARY	Binary
BINOMIAL	Binary or binomial
EXPONENTIAL	Exponential
GAMMA	Gamma
GENPOISSON | GPOISSON	Generalized Poisson
GEOMETRIC	Geometric
IGAUSSIAN | IG	Inverse Gaussian
LOGNORMAL | LOGN	Lognormal
MULTINOMIAL	Multinomial. For ordinal responses, it fits a model that has a cumulative link function. For nominal responses, it fits a model that has a generalized logit link function.
NEGATIVEBINOMIAL | NB	Negative binomial
NORMAL | GAUSSIAN	Normal
POISSON	Poisson
T<ν>	t with ν degrees of freedom. If you do not specify ν, a value of $\nu = 3$ is used.
TWEEDIE< (*Tweedie-options*) >	Tweedie
WEIBULL	Two-parameter Weibull

When DISTRIBUTION=TWEEDIE, you can specify the following *Tweedie-options*:

EQL
uses extended quasi-likelihood instead of Tweedie log likelihood in parameter estimation.

INITIALP=*value*
specifies a starting *value* for iterative estimation of the Tweedie power parameter.

OPTMETHOD=*Tweedie-optimization-option*
specifies the optimization method for iterative estimation of the Tweedie model parameters. You can specify the following *Tweedie-optimization-options*:

EQL
uses extended quasi-likelihood for a sample of the data, followed by extended quasi-likelihood for the full data. This is equivalent to the EQL *Tweedie-option*.

EQLLHOOD
uses extended quasi-likelihood for a sample of the data, followed by Tweedie log likelihood for the full data. This is the default method.

FINALLHOOD
uses a four-stage approach to estimating the Tweedie model parameters. The four stages are as follows:

1. extended quasi-likelihood for a sample of the data
2. Tweedie log likelihood for a sample of the data
3. extended quasi-likelihood for the full data
4. Tweedie log likelihood for the full data

LHOOD
uses Tweedie log likelihood be for a sample of the data, followed by Tweedie log likelihood for the full data.

P=*value*
specifies a *value* to use as a fixed Tweedie power parameter.

SAMPLEFRAC=*value*
specifies a *value* to use as the fraction of the data that are used to compute starting values for the Tweedie distribution. The *value* must be between 0 and 1.

INCLUDE=*n*

INCLUDE=*single-effect*

INCLUDE=(*effect-list***)**
forces effects to be included in all models. If you specify INCLUDE=*n*, then the first *n* effects that are listed in the MODEL statement are included in all models. If you specify INCLUDE=*single-effect* or if you specify a list of effects within parentheses, then the specified effects are forced into all models. The effects that you specify in the INCLUDE= option must be explanatory effects that are specified in the MODEL statement before the slash (/). This option is not available if you use elastic net selection.

INFORMATIVE

models missing values by using extra model effects. These effects consist of dummy variables that take the value 1 when the value of a continuous model variable involved in the effect is missing, and take the value 0 otherwise. The missing value in the original model effect is replaced by the average value of the effect for the nonmissing values. For continuous-by-class effects, such as A*x, where A is a classification variable and x is a continuous variable, informative missingness creates multiple dummy columns and substitutes the effect mean of x that corresponds to the respective level of A. Missing values for classification variables are treated as valid levels. For more information about informative missingness, see the section "Informative Missingness" on page 99 in Chapter 2, "Shared Concepts."

LINK=*keyword*

specifies the link function for the model. The *keywords* and their associated link functions are shown in Table 14.8. Default and commonly used link functions for the available distributions are shown in Table 14.9.

Table 14.8 Built-In Link Functions

Keyword	Link Function	$g(\mu) = \eta =$
CLOGLOG | CLL	Complementary log-log or cumulative complementary log-log	$\log(-\log(1-\mu))$
GLOGIT	Generalized logit	$\log(\mu_j/\mu_J)$
IDENTITY | ID	Identity	μ
INVERSE | RECIPROCAL | RECIP	Reciprocal	$\frac{1}{\mu}$
POWERMINUS2 | INV2 | POWMINUS2	Reciprocal square	$\frac{1}{\mu^2}$
POWER(p)	Power	μ^p
LOG	Logarithm	$\log(\mu)$
LOGIT	Logit or cumulative logit	$\log(\mu/(1-\mu))$
LOGLOG	Log-log or cumulative log-log	$-\log(-\log(\mu))$
PROBIT | NORMIT	Probit or cumulative probit	$\Phi^{-1}(\mu)$

For the probit and cumulative probit links, $\Phi^{-1}(\cdot)$ denotes the quantile function of the standard normal distribution.

If you do not specify the LINK= option, a default link function is used, as shown in Table 14.9. For binary or multinomial distributions, only the link functions shown in Table 14.9 are available. For the other distributions, you can use any link function shown in Table 14.8 by specifying the LINK= option. Other commonly used link functions for each distribution are shown in Table 14.9.

Table 14.9 Default and Commonly Used Link Functions

Value of the DISTRIBUTION= Option	Default Link Function	Other Commonly Used Link Functions
BETA	Logit	Probit, complementary log-log, log-log
BINARY	Logit	Probit, complementary log-log, log-log
BINOMIAL	Logit	Probit, complementary log-log, log-log
EXPONENTIAL	Log	
GAMMA	Log	
GENPOISSON | GPOISSON	Log	
GEOMETRIC	Log	
INVERSEGAUSSIAN | IG	Log	
LOGNORMAL | LOGN	Identity	
MULTINOMIAL	Cumulative logit	Cumulative probit, cumulative complementary log-log, cumulative log-log, generalized logit
NEGATIVEBINOMIAL | NB	Log	
NORMAL | GAUSSIAN	Identity	Log
POISSON	Log	
T	Identity	
TWEEDIE	Log	
WEIBULL	Log	

NOINT
includes no intercept in the model. An intercept is included by default. The NOINT option is not available for multinomial models.

OFFSET=*variable*
specifies a *variable* to be used as an offset to the linear predictor. An offset plays the role of an effect whose coefficient is known to be 1. The offset variable cannot appear in the CLASS statement or elsewhere in the MODEL statement. Observations that have missing values for the offset variable are excluded from the analysis.

PHI=*number*
specifies a fixed dispersion parameter for those distributions that have a dispersion parameter. The dispersion parameter that is used in all computations is fixed at *number* and not estimated.

START=*n*

START=*single-effect*

START=(*effects***)**
begins the selection process from the designated initial model for the forward selection method. If you specify START=*n*, then the starting model includes the first *n* effects that are listed in the MODEL statement. If you specify START=*single-effect* or START=(*effects*), then the starting model includes those specified effects. The effects that you specify in the START= option must be explanatory effects that are specified in the MODEL statement before the slash (/). This option is not available when you specify METHOD=BACKWARD or ELASTICNET in the SELECTION statement.

TYPE3
computes Wald statistics for Type 3 contrasts for each effect that you specify in the MODEL statement. This option is not available for models that you fit by using the GEE method. For more information, see the section "Joint Tests and Type 3 Tests" on page 772.

OUTPUT Statement

OUTPUT OUT=*libref.data-table*
< **ALL** > < **ALPHA=***number* > < **COPYVARS=***(variables)* >
< *keyword* < **=***name* > >. . . < *keyword* < **=***name* > > **;**

The OUTPUT statement creates a data table that contains observationwise statistics that PROC GENSELECT computes after fitting the model. In order to avoid data duplication for large data tables, the variables in the input data table are *not* included in the output data table unless you specify them in the COPYVAR= option.

If the response variable has more than two categories, you can request the "Basic Options" listed in Table 14.10; the other diagnostic statistics are not available. These statistics are computed for every response category, and the automatic variable _LEVEL_ identifies the response category on which the computed values are based. That is, every observation generates several rows in the output data set. If you also specify the OBSCAT option, then the observationwise statistics are computed only for the observed response category, which is indicated by the value of the _LEVEL_ variable.

The output statistics are computed based on the final parameter estimates. If the optimization does not converge, then the output data table is not created. If you perform elastic net or LASSO selection, or if you specify the NOXPX or NOSTDERR option, then your analysis does not generate a covariance matrix, and you cannot produce the basic and diagnostic statistics in Table 14.10 that require a covariance matrix.

For observations in which only the response variable is missing, values of the linear predictor and the predicted values are computed even though these observations do not affect the model fit. This enables, for example, predicted values to be computed for new observations.

You must specify the following option:

OUT=*libref.data-table*
names the output data table for PROC GENSELECT to use. You must specify this option before any other options. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to where the data table is to be stored, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 707.

data-table specifies the name of the output data table.

You can also specify the following syntax elements:

ALL

ALLSTAT
adds all available statistics to the output data table.

ALPHA=*number*
specifies the significance level for the construction of confidence intervals in the output data table. The confidence level is $1 - number$. The value of *number* must be between 0 and 1. By default, *number* is equal to the value of the ALPHA= option in the PROC GENSELECT statement, or 0.05 if that option is not specified.

COPYVAR=*variable*

COPYVARS=(*variables***)**
transfers one or more *variables* from the input data table to the output data table.

OBSCAT
requests (for multinomial models) that observationwise statistics be produced only for the observed response level. If you do not specify this option and the response variable has J levels, then the following outputs are created: for cumulative link models, $J - 1$ records are output for every observation in the input data that corresponds to the $J - 1$ lower-ordered response categories; for generalized logit models, J records are output that correspond to all J response categories.

keyword **<** **=***name* **>**
specifies a statistic to include in the output data table and optionally names the variable *name*. If you do not provide a *name*, the GENSELECT procedure assigns a default name based on the type of statistic requested. If a statistic is not available for your model then that request is ignored.

Table 14.10 summarizes the *keywords* available in the OUTPUT statement.

Table 14.10 OUTPUT Statement Keywords

Keyword	Description	Default Name
Basic Options		
INDIVIDUAL	Specifies the individual predicted probabilities	_IPRED_
PREDICTED	Specifies the predicted probabilities	_PRED_
XBETA	Specifies the linear predictor	_XBETA_
Basic Options Requiring the Covariance		
LCL	Specifies the lower confidence limit for the linear predictor	_LCL_
LCLM	Specifies the lower confidence limit for the event probability	_LCLM_
STDXBETA	Specifies the standard error estimate of the linear predictor	_STDXBETA_
UCL	Specifies the upper confidence limit for the linear predictor	_UCL_
UCLM	Specifies the upper confidence limit for the event probability	_UCLM_
Diagnostic Options		
RESCHI	Specifies the Pearson chi-square residual	_RESCHI_
RESDEV	Specifies the deviance residual	_RESDEV_
RESRAW	Specifies the raw residual	_RESRAW_
RESWORK	Specifies the working residual	_RESWORK_
Diagnostic Options Requiring the Covariance		
CBAR	Specifies the confidence interval displacement	_CBAR_

Table 14.10 *continued*

Keyword	Description	Default Names
DIFCHISQ	Specifies the deletion chi-square goodness-of-fit change	_DIFCHISQUARE_
DIFDEV	Specifies the deletion deviance change	_DIFDEVIANCE_
H	Specifies the leverage	_HATDIAG_
RESLIK	Specifies the likelihood residual	_RESLIK_
STDRESCHI	Specifies the standardized Pearson chi-square residual	_STDRESCHI_
STDRESDEV	Specifies the standardized deviance residual	_STDRESDEV_
Miscellaneous Options for Binary and Multinomial Response Data		
INTO	Names the level into which the observation is classified	_INTO_
LEVEL	Names the response level for a row of the output	_LEVEL_
Option for Generalized Estimating Equations		
CLEVERAGE	Names the cluster leverage	_CLEVERAGE_

The following lists describe these *keywords*. For more information, see the section "Predicted Values and Regression Diagnostics" on page 768.

CBAR

specifies the confidence interval displacement diagnostic that measures the overall change in the global regression estimates that results from deleting an individual observation. The default name is _CBAR_.

DIFCHISQ

specifies the change in the chi-square goodness-of-fit statistic that results from deleting the individual observation. The default name is _DIFCHISQUARE_.

DIFDEV

specifies the change in the deviance that results from deleting the individual observation. The default name is _DIFDEVIANCE_. This statistic is not supported for GEEs.

H

specifies the diagonal element of the hat matrix (leverage) for detecting extreme points in the design space. The default name is _HATDIAG_. For GEEs, the leverage value is the corresponding diagonal element of the cluster leverage matrix.

INDIVIDUAL

IPRED

IPROB

IP

specifies the individual predicted values for multinomial response variables. For a response variable Y with three levels, 1, 2, and 3, the individual probabilities are $\Pr(Y = 1)$, $\Pr(Y = 2)$, and $\Pr(Y = 3)$. The default name is _IPRED_.

INTO< (*cutpoint*) >
names the variable that contains the level of the response into which an observation is classified. The default name is _INTO_. Multinomial models classify observations into the level that has the largest model-predicted probability. For binary or binomial response variables, if the predicted probability of an observation equals or exceeds the cutpoint, the observation is classified as an event; otherwise it is classified as a nonevent. You can specify the *cutpoint* value as a number between 0 and 1. The default value is 0.5.

LCL

LOWERXBETA
names the variable that contains the lower confidence limits for the linear predictor. The default name is _LCL_. You can set the confidence level by specifying the ALPHA= option.

LCLM

LOWERMEAN

LOWER
specifies the lower confidence limits for the mean. The default name is _LCLM_. You can set the confidence level by specifying the ALPHA= option.

LEVEL
names the variable that contains the level of the response for a given row of the output. The default name is _LEVEL_.

PREDICTED

PRED

PROB

P
specifies the predicted values for the response variable. It specifies the predicted probabilities of events for binary and nominal response variables and the cumulative predicted probabilities for ordinal response variables. For a response variable Y with three levels, 1, 2, and 3, the cumulative probabilities are $\Pr(\mathrm{Y} \leq 1)$ and $\Pr(\mathrm{Y} \leq 2)$, but by default the last level, $\Pr(\mathrm{Y} \leq 3) = 1$, is not output. The default name is _PRED_.

RESCHI

PEARSON
specifies the Pearson residual for identifying poorly fitted observations. The default name is _RESCHI_.

RESDEV
specifies the deviance residual for identifying poorly fitted observations. The default name is _RESDEV_. This statistic is not supported for GEEs.

RESLIK
specifies the likelihood residual for identifying poorly fitted observations. The default name is _RESLIK_. This statistic is not supported for GEEs.

RESRAW

RESIDUAL

R

specifies the raw residual for identifying poorly fitted observations. The default name is _RESRAW_.

RESWORK

specifies the working residual for identifying poorly fitted observations. The default name is _RESWORK_.

ROLE

specifies the numeric variable that indicates the role played by each observation in fitting the model. The default name is _ROLE_. Table 14.11 shows how this variable is interpreted for each observation.

Table 14.11 Role Interpretation

Value	Observation Role
0	Not used
1	Training
2	Validation
3	Testing

If you do not partition the input data by specifying a PARTITION statement, then the role variable value is 1 for observations that are used in fitting the model and 0 for observations that have at least one missing or invalid value for the response, regressor, frequency, or weight variables.

STDRESCHI

specifies the standardized Pearson (chi-square) residual for identifying observations that are poorly accounted for by the model. The default name is _STDRESCHI_.

STDRESDEV

specifies the standardized deviance residual for identifying poorly fitted observations. The default name is _STDRESDEV_. This statistic is not supported for GEEs.

STDXBETA

specifies the standard error estimates of XBETA. The default name is _STDXBETA_.

UCL

UPPERXBETA

specifies the variable that contains the upper confidence limits for the linear predictor. The default name is _UCL_. You can set the confidence level by specifying the ALPHA= option.

UCLM

UPPERMEAN

UPPER

specifies the variable that contains the upper confidence limits for the mean. The default name is _UCLM_. You can set the confidence level by specifying the ALPHA= option.

XBETA

LINP

specifies the linear predictor. The default name is _XBETA_.

The following additional output statistic is available for GEEs. For more information about diagnostic statistics, see the section "Diagnostics for Models Fit by Generalized Estimating Equations (GEEs)" on page 771.

CLEVERAGE

CH

CLEV

specifies the cluster leverage. The default name is _CLEVERAGE_.

PARTITION Statement

PARTITION *partition-option* **;**

The PARTITION statement specifies how observations in the input data set are logically partitioned into disjoint subsets for model training, validation, and testing. For more information, see the section "Using Validation and Test Data" on page 101 in Chapter 2, "Shared Concepts." Either you can designate a variable in the input data table and a set of formatted values of that variable to determine the role of each observation, or you can specify proportions to use for randomly assigning observations to each role.

You must specify exactly one of the following *partition-options*:

FRACTION(< TEST=*fraction* **> < VALIDATE=***fraction* **> < SEED=***number* **>)**

randomly assigns specified proportions of the observations in the input data table to the roles. You specify the proportions for testing and validation by using the TEST= and VALIDATE= suboptions. If you specify both the TEST= and VALIDATE= suboptions, then the sum of the specified fractions must be less than 1 and the remaining fraction is the proportion of the observations assigned to the training role. The SEED= option specifies an integer that is used to start the pseudorandom number generator for random partitioning of data for training, testing, and validation. If you do not specify SEED=*number* or if *number* is less than or equal to 0, the seed is generated by reading the time of day from the computer's clock.

ROLE=*variable* **(< TEST='***value***' > < TRAIN='***value***' > < VALIDATE='***value***' >)**

ROLEVAR=*variable* **(< TEST='***value***' > < TRAIN='***value***' > < VALIDATE='***value***' >)**

names the *variable* in the input data table whose values are used to assign roles to each observation. This *variable* cannot also appear as an analysis variable in other statements or options. The TEST=, TRAIN=, and VALIDATE= suboptions specify the formatted values of this variable that are used to assign observation roles. If you do not specify the TRAIN= suboption, then all observations whose role is not determined by the TEST= or VALIDATE= suboption are assigned to the training role.

For more information, see the section "Partition Fit Statistics" on page 767.

REPEATED Statement

REPEATED SUBJECT=<*subject-effect*> / < *repeated-options* > **;**

The REPEATED statement performs a repeated measures analysis by using generalized estimating equations (GEEs). GEE analysis is available for models that use the normal, gamma, inverse Gaussian, negative binomial, binomial, binary, and Poisson distributions. GEE analysis is not supported when you perform model selection, when you perform a partitioned analysis, or when you specify parameter restrictions.

The *repeated-options* specify the working correlation structure, output options, and controls for the iterative fitting algorithm.

Table 14.12 summarizes the *repeated-options* in the REPEATED statement by function.

Table 14.12 REPEATED Statement Options

Option	Description
Basic Options	
SUBJECT=	Identifies subjects in the input data
TYPE=	Specifies the working correlation type
WITHIN=	Specifies a within-subject effect
Options Related to Output	
CORRB	Produces both the model-based and empirical estimated regression parameter correlation matrices
CORRW	Produces the working correlation matrix
COVB	Produces both the model-based and empirical estimated regression parameter covariance matrices
ECORRB	Produces the estimated regression parameter empirical correlation matrix
ECOVB	Produces the estimated regression parameter empirical covariance matrix
MCORRB	Produces the estimated regression parameter model-based correlation matrix
MCOVB	Produces the estimated regression parameter model-based covariance matrix
MODELSE	Produces a parameter estimates table that displays and uses the model-based standard errors
PRINTMLE	Produces a parameter estimates table that displays the maximum likelihood estimates

Basic Options

SUBJECT=*subject-effect*

identifies subjects in the input data set. A *subject-effect* is required; it can contain a single variable, an interaction effect, a nested effect, or a combination of these. Each variable that the *subject-effect* contains must be listed in the CLASS statement. Each distinct level of the effect identifies a different subject. Responses from different subjects are assumed to be independent, but responses within subjects are assumed to be correlated.

You can specify the following *options* after a slash (/):

TYPE=*<type>*

specifies the working correlation type that is used in the repeated measures analysis. The working correlation structure describes the way that a subject's repeated measures are assumed to correlate with each other.

You can specify one of the following working correlation types:

AR | AR(1)

specifies the autoregressive-1 working correlation.

CS | EXCH

specifies the exchangeable working correlation.

IND

specifies the independence working correlation.

MDEP*(m)* **| TOEP***(m)*

specifies the m-dependent working correlation. By default, $m = 1$.

UN | UNSTR | UNSTRUCTURED

specifies the unstructured working correlation.

By default, TYPE=IND.

WITHIN=*within-effect*

WITHINSUBJECT=*within-effect*

specifies a within-subject effect for the repeated measures model. This effect must be composed of CLASS variables. The purpose of the within-subject effect is to define an order for the repeated measures that correspond to each subject. This definition can be critical for reproducibility when you use an order-sensitive correlation structure.

If you do not specify a within-subject effect, the order of the repeated measures is determined by the order in which they are read. This order can vary, depending on your computing resources. Therefore, a within-subject effect is required when you specify the autoregressive, m-dependent, or unstructured working correlation type. When you use one of these working correlation types, the levels of the within-subject effect must be unique within each subject.

If you use multiple variables to define the within-subject effect, the specification of the variables within this effect will have an impact on the analysis if the working correlation type depends on the ordering of the repeated measures. The autoregressive, m-dependent, and unstructured working correlation types use the levelization of the within-subject effect to locate observations within the correlation matrix, and depend on the order that is produced by the levelization.

In particular, if you define the within-subject effect by using the interaction of two or more variables, then the hierarchy of levelization and the resulting ordering of the repeated measures will depend on the order in which you list the variables in the CLASS statement.

For example, assume that you are using the interaction of the variables w1 and w2 as the within-subject effect in a repeated measures analysis. Furthermore, assume that w1 takes the values 'A' and 'B' and w2 takes the values 'L' and 'H'. The following statements list these variables in the order w2, w1 in the CLASS statement:

```
proc genselect data=mylib.health;
   class subject w2 w1;
   model outcome(event='1') = x1 x2 / dist=binary;
   repeated subject=subject / type=mdep(2) within=w1*w2;
run;
```

In this case, the order of the variables in the CLASS statement results in a levelization in which the value of w1 changes faster than the value of w2. Table 14.13 shows this levelization.

Table 14.13 Levelization Order When Variable Order Is w2, w1

Value of w1	Value of w2	Levelization Order
A	L	0
B	L	1
A	H	2
B	H	3

If you reverse the order of the variables in the CLASS statement, the levelization has a different hierarchy. In this hierarchy, the value of w2 changes faster than the value of w1. Table 14.14 shows that the levelization order that is assigned to each unique combination of the variables does not match the levelization order shown in Table 14.13.

Table 14.14 Levelization Order When Variable Order Is w1, w2

Value of w1	Value of w2	Levelization Order
A	L	0
B	L	2
A	H	1
B	H	3

Because the specified working correlation structure (*m*-dependent) relies on this ordering, the parameter estimates differ from those that are produced by the ordering shown in Table 14.13. The autoregressive and unstructured working correlation structures also result in different parameter estimates for the different orderings.

If you use two variables to define the within-subject effect, you can use a nested effect to enforce a specific levelization hierarchy. However, the best practice is to use a single variable in the within-subject effect.

Options Related to Output

CORRB
produces both the model-based and empirical estimated regression parameter correlation matrices.

CORRW
produces the estimated working correlation matrix. This matrix shows the estimated correlation between pairs of repeated measurements.

COVB
produces both the model-based and empirical estimated regression parameter covariance matrices.

ECORRB
produces the empirical estimated regression parameter correlation matrix.

ECOVB
produces the empirical estimated regression parameter covariance matrix.

MCORRB
produces the model-based estimated regression parameter correlation matrix.

MCOVB
produces the model-based estimated regression parameter covariance matrix.

MODELSE
produces a table that displays the parameter estimates from the repeated measures analysis, but using the model-based standard error estimates. By default, only a table that uses the empirical standard error estimates is shown.

PRINTMLE
produces a table that displays the parameter estimates and standard errors from maximum likelihood estimation, assuming no within-subject correlation. This table corresponds to the "Parameter Estimates" table that the procedure produces when you do not request a repeated measures analysis.

RESTRICT Statement

RESTRICT *constraint-specification* < **,** . . . **,** *constraint-specification* >
< *operator* < *value* > > < / *option* > **;**

The RESTRICT statement enables you to specify linear equality or inequality constraints among the parameters of a model. These restrictions are incorporated into the maximum likelihood analysis.

You can specify the following *option* after a slash (/):

DIVISOR=*value*
specifies a *value* by which all coefficients on the right-hand and left-hand sides of the restriction are divided.

Linear Constraints

There are several reasons why you might want to place constraints and restrictions on the model parameters:

- to fix a parameter at a particular value
- to equate parameters in a model
- to impose order conditions on the parameters in a model
- to specify contrasts among the parameters that the fitted model should honor

A restriction is composed of a left-hand side and a right-hand side, separated by an operator. If you do not specify the operator and right-hand side, the restriction is assumed to be an equality constraint against zero. If you do not specify the right-hand side, the value is assumed to be zero.

The *constraint-specification* takes the form

model-effect value-list < . . . *model-effect value-list* >

You must specify at least one *model-effect*, followed by one or more values in the *value-list*. The values in the list correspond to the multipliers of the corresponding parameter that is associated with the position in the model effect. If you specify more values in the *value-list* than the *model-effect* occupies in the model design matrix, the extra coefficients are ignored.

NOTE: You cannot use a specific *model-effect* more than once in a *constraint-specification*.

You can use the following operators to separate the left- and right-hand sides of a restriction: `=`, `>`, `<`, `>=`, `<=`. If there is no *operator*, then `=0` is the assumed *operator rhs-value* pair. If there is an *operator* but no *rhs-value*, then `0` is the assumed *rhs-value*.

Basic Syntax

The following SAS statements provide an example. Here, A is a CLASS variable that has three levels.

```
proc genselect data=mylib.binom;
   class A;
   model y/n = A x /  dist=binomial;
   restrict A 1 0 -1;
   restrict x 2 >= 0.5;
run;
```

The linear predictor for this model can be written as

$$\eta = \beta_0 + \beta_1 A_1 + \beta_2 A_2 + \beta_3 A_3 + x\beta_4$$

where A_k is the binary variable that is associated with the kth level of A.

The first RESTRICT statement specifies that the parameter estimates that are associated with the first and third levels of the A effect be identical. In terms of the linear predictor, the restriction can be written as

$$\beta_1 - \beta_3 = 0$$

In the default GLM parameterization, $\beta_3 = 0$, so this RESTRICT statement has the effect of setting $\beta_1 = 0$.

The second RESTRICT statement involves the regression parameter that is associated with the variable x and specifies that the parameter estimate satisfy $\beta_4 \geq 0.25$. In terms of the linear predictor, the restriction can be written as

$$2\beta_4 \geq \frac{1}{2}$$

PROC GENSELECT applies both of these restrictions when it computes the maximum likelihood estimates of the regression parameters of the model.

Generalized Logit Syntax

The generalized logit model for a nominal multinomial response consists of a regression model for each nonreference level of the response variable. To specify restrictions for effects in specific components of the model, you specify a restriction for each component to which you want to apply constraints. You specify each restriction in the sort order of the response variable and separate them with commas.

You must specify a restriction in which you set *value-list* to zero for each component model that has a lower response variable sort order than the component model to which you want to apply constraints.

The following SAS statements provide an example. In this example, a generalized logit regression model is fitted to the categorical response variable Y. The response variable has four different levels. The generalized logit model consists of a regression model that has a CLASS regressor variable named Visit and a continuous regressor variable named Lage for each level of the response variable Y. The RESTRICT statements constrain the model to have identical values of the estimated regression coefficient for Lage for all three nonreference categories of Y; that is, a common-slopes model is fitted. In the second RESTRICT statement, the *constraint-specification* of *Lage 0* is necessary as a placeholder and does not affect the regression coefficient of Lage for the first level of Y.

```
proc genselect data=mylib.mult;
   class Visit / Param=Ref;
   model Y=Visit Lage/dist=Multinomial link=Glogit;
   restrict Lage 1 , Lage -1;
   restrict Lage 0 , Lage  1, Lage -1;
run;
```

Some distributions involve a dispersion parameter (the parameter ϕ in the expressions for the log likelihood), and in the case of the Tweedie distribution, a power parameter. You cannot use the RESTRICT statement to constrain either of these parameters. Instead, you can use the MODEL statement option PHI= to set the dispersion to a fixed value. Likewise, you can use the MODEL statement option P= to set the Tweedie power parameter to a fixed value.

In multinomial models, you cannot constrain the intercept parameter to be equal to zero.

Linear constraints are not supported in repeated measures (GEE) analysis, in model selection, or when you have specified informative missingness. Furthermore, linear constraints are not supported for polynomial constructed effects.

SELECTION Statement

SELECTION < **METHOD=***method* < *(method-options)* > >< *options* > **;**

The SELECTION statement performs model selection by examining whether effects should be added to or removed from the model according to rules that are defined by model selection methods. The statement is fully documented in the section "SELECTION Statement" on page 42 in Chapter 2, "Shared Concepts."

The GENSELECT procedure supports the following effect-selection *methods* in the SELECTION statement:

BACKWARD performs backward elimination. This method starts with all effects in the model and deletes effects.

BACKWARD(FAST) performs fast backward elimination when SELECT=SL. This method starts with all effects in the model and deletes effects without refitting the model.

ELASTICNET performs elastic net selection. This method selects effects by fitting a sequence of candidate models with a grid of regularization parameters. For more information, see the section "Elastic Net Selection" on page 83 in Chapter 2, "Shared Concepts."

FORWARD performs forward selection. This method starts with no effects in the model and adds effects.

LASSO performs model selection by the group LASSO method. This method adds and removes effects by using a sequence of LASSO steps. For more information, see the section "Group LASSO Selection" on page 87 in Chapter 2, "Shared Concepts."

NONE results in no model selection. This method fits the full model.

STEPWISE performs stepwise selection. This method is similar to the FORWARD method except that effects already in the model do not necessarily stay there.

By default, METHOD=STEPWISE.

The SELECT=, CHOOSE=, and STOP= *method-options* default to SBC. These defaults differ from their corresponding defaults in the HPGENSELECT procedure in SAS/STAT software.

You can specify the following criteria in the SELECT=, CHOOSE=, and STOP= *method-options*:

AIC uses Akaike's information criterion (Akaike 1974).

AICC uses a small-sample bias corrected version of Akaike's information criterion, as promoted in Hurvich and Tsai (1989) and Burnham and Anderson (1998).

SBC | BIC uses the Schwarz Bayesian criterion (Schwarz 1978).

SL uses the significance level of the score test as the criterion (not available for the CHOOSE= option).

VALIDATE uses the average square error (ASE) that is computed on the validation data as the criterion (not available for the SELECT= option).

For more information, see the section "Information Criteria" on page 767. If you specify the PARTITION statement, then the AIC, AICC, SBC, and SL statistics are computed on the training data; otherwise they are computed on the full data table.

Forward selection where you also specify the NOINT option is available only for the following models: binary, binomial, exponential, geometric, and Poisson.

If you specify METHOD=LASSO, then the value of the CHOOSE= option is SBC by default, you can specify CHOOSE=NONE to select the model in the last LASSO step, and the STOPHORIZON= option has no effect.

If you specify METHOD=ELASTICNET, then the HIERARCHY= and STOPHORIZON= options have no effect. The elastic net method is not currently available with the lognormal distribution.

NOTE: If you use the fast backward elimination method, then the –2 log-likelihood, AIC, AICC, and SBC statistics are approximated at each step where the model is not refit, and hence they do not match the values that are computed when that model is fit outside the selection routine. Similarly, if you specify SELECT=AIC, AICC, or SBC, the selection criteria are estimated (Lawless and Singhal 1978), and hence they do not match the values that are computed when that model is fit outside the selection routine.

NOTE: The default model hierarchy method is HIERARCHY=NONE for the forward and fast backward selection methods. The backward elimination method always uses the HIERARCHY=SINGLE *method-option*.

The LASSO method produces a summary table that displays the effects that are added or removed at each step; the LASSO regularization parameter; and the AIC, AICC, and SBC fit statistics. The elastic net method produces a similar summary table that includes the selected effects, the regularization parameter, and the value of the CHOOSE= criterion. For the other methods, you can specify the following values for the DETAILS= option:

SUMMARY produces a summary table that shows the effect that is added or removed at each step along with the SELECT=, CHOOSE=, and STOP= criteria. The summary table is produced by default if the DETAILS= option is not specified.

STEPS produces the preceding summary table and displays the results from fitting each model at each step.

ALL produces the preceding tables and a detailed listing of all candidates at each step along with their ranking in terms of the selection criterion for addition to or removal from the model.

If you specify the PLOTS=CRITERIA or PLOTS=ALL option, then a plot of the fit criterion by the selection step is created for the AIC, AICC, and SBC statistics. If you also specify a PARTITION statement or the PARTFIT option, then the same type of plot is created for the ASE (for each role).

If you specify the PLOTS=FITBYROLE or PLOTS=ALL option and a PARTITION statement, then a plot of the ASE by the selection step for each role is created.

The PLOTS= option is not available for the LASSO method or the elastic net method.

Model selection is not available for repeated measures (GEE) models or when you specify linear restrictions.

STORE Statement

STORE < **OUT=***libref.data-table* > < / **NOTE** | **TEXT=***quoted-string* < *quoted-string . . .* > > **;**

The STORE statement saves the context and results of the statistical analysis. The resulting analytic store has a binary file format that cannot be modified. An example of analytic store technology is to perform a time-consuming analysis and to store its results by using the STORE statement. Later you can perform specific statistical analysis tasks based on the saved results of the previous analysis, without having to fit the model again. You can process the contents of the analytic store by specifying the RESTORE= option, as well as by specifying the ASTORE procedure and the **genmodCode**, **genmodScore**, and **genmodType3** actions in the **regression** action set (see Chapter 22, "Regression Action Set" (*SAS Visual Statistics: Programming Guide*)).

You specify the name of the analytic store by using the OUT= option. You can save notes to the analytic store by specifying them as quoted strings in the NOTE= or TEXT= option. Each quoted string begins a new line, preserves initial blanks, and can be at most 200 characters long. These text strings are displayed when you specify the item store in a RESTORE= option. *libref.data-table* is a two-level name, where *libref* refers to the library, and *data-table* specifies the name of the output data table. For more information about this two-level name, see the section "Using CAS Sessions and CAS Engine Librefs" on page 707.

WEIGHT Statement

WEIGHT *variable* < / **NORMALIZE** > **;**

The *variable* in the WEIGHT statement is used as a weight to perform a weighted analysis of the data. Observations that have nonpositive or missing weights are not included in the analysis. The values of the WEIGHT variable can be nonintegral and are not truncated. When you omit the WEIGHT statement, all observations that are used in the analysis are assigned a weight of 1.

You can specify the following option in the WEIGHT statement after a slash (/):

NORMALIZE

normalizes the weights that are specified by the WEIGHT *variable* so that they add up to the sample size. Weights are normalized by multiplying them by $\frac{\sum f_i t_i}{\sum w_i f_i t_i}$, where training observation i has a frequency equal to f_i and a weight equal to w_i, and t_i is the number of trials when you are using events/trials or multinomial-trial syntax; otherwise t_i is equal to 1. When you specify this option, the estimated covariance matrix of the parameter estimators is invariant to the scale of the WEIGHT *variable*.

Details: GENSELECT Procedure

Missing Values

Any observation that has missing values for the response, frequency, weight, offset, or explanatory variables is excluded from the analysis; however, missing values are valid for response and explanatory variables that are specified along with the MISSING option in the CLASS statement. Observations that have a nonpositive weight or a frequency less than 1 are also excluded.

The estimated linear predictor and the fitted probabilities are not computed for any observation that has missing offset or explanatory variable values. However, if only the response value is missing, the linear predictor and the fitted probabilities can be computed and output to a data set by using the OUTPUT statement.

You can also model the missing values by specifying the INFORMATIVE option in the MODEL statement. For more information about informative missingness, see the section "Informative Missingness" on page 99 in Chapter 2, "Shared Concepts."

Exponential Family Distributions

Many of the probability distributions that the GENSELECT procedure fits are members of an exponential family of distributions, which have probability distributions that are expressed as follows for some functions b and c that determine the specific distribution:

$$f(y) = \exp\left\{\frac{y\theta - b(\theta)}{\phi} + c(y,\phi)\right\}$$

For fixed ϕ, this is a one-parameter exponential family of distributions. The response variable can be discrete or continuous, so $f(y)$ represents either a probability mass function or a probability density function. A more useful parameterization of generalized linear models is by the mean and variance of the distribution:

$$\begin{aligned} \mathrm{E}(Y) &= b'(\theta) \\ \mathrm{Var}(Y) &= b''(\theta)\phi \end{aligned}$$

In generalized linear models, the mean μ of the response distribution is related to linear regression parameters through a link function,

$$g(\mu_i) = \mathbf{x}_i'\boldsymbol{\beta}$$

for the ith observation, where $\mathbf{x}_i$ is a fixed known vector of explanatory variables and $\boldsymbol{\beta}$ is a vector of regression parameters. The GENSELECT procedure parameterizes models in terms of the regression parameters $\boldsymbol{\beta}$ and either the dispersion parameter ϕ or a parameter that is related to ϕ, depending on the model. For exponential family models, the distribution variance is $\mathrm{Var}(Y) = \phi \mathrm{V}(\mu)$, where $\mathrm{V}(\mu)$ is a variance function that depends only on μ.

Other distributions that are not exponential family models but are sometimes useful in statistical modeling are also included in the GENSELECT procedure.

Response Distributions

The response distribution is the probability distribution of the response (target) variable. The GENSELECT procedure can fit data for the following exponential family distributions:

- binary distribution
- binomial distribution
- gamma distribution
- geometric distribution
- exponential distribution
- inverse Gaussian distribution
- lognormal distribution
- multinomial for ordinal or nominal responses
- negative binomial distribution
- normal (Gaussian) distribution
- Poisson distribution
- Tweedie distribution

The following distributions are not exponential family members but are also included in the GENSELECT procedure:

- beta distribution
- generalized Poisson distribution
- *t* distribution
- Weibull distribution

Expressions for the probability distributions (probability density functions for continuous variables or probability mass functions for discrete variables) are shown in the section "Response Probability Distribution Functions" on page 749. The expressions for the log-likelihood functions of these distributions are presented in the section "Log-Likelihood Functions" on page 754.

The binary (or Bernoulli) distribution is the elementary distribution of a discrete random variable that can take two values, which have the probabilities p and $1 - p$. Suppose the random variable is denoted as Y and

$$\Pr(Y = 1) = p$$
$$\Pr(Y = 0) = 1 - p$$

The value that is associated with probability p is often called the *event* or "success"; the complementary event is called the *nonevent* or "failure." A Bernoulli experiment is a random draw from a binary distribution and generates events with probability p.

If $Y_1, \ldots, Y_n$ are n independent Bernoulli random variables, then their sum follows a binomial distribution. In other words, if $Y_i = 1$ denotes an event (success) in the ith Bernoulli trial, a binomial random variable is the number of events (successes) in n independent Bernoulli trials. If you use the events/trials syntax in the MODEL statement and you specify the DISTRIBUTION=BINOMIAL option, the GENSELECT procedure fits the model as if the data had arisen from a binomial distribution. For example, the following statements fit a binomial regression model that has the regressors x1 and x2. The variables e and t represent the events and trials, respectively, for the binomial distribution:

```
proc genselect;
   model e/t = x1 x2 / distribution=Binomial;
run;
```

If the events/trials syntax is used, then both variables must be numeric and the value of the events variable cannot be less than 0 or exceed the value of the trials variable. A "Response Profile" table is not produced for binomial data, because the response variable is not subject to levelization.

If the response variable is listed in a CLASS statement and a response distribution is not specified in a DISTRIBUTION= option, then a binary distribution or a multinomial distribution that uses the default or specified link function is assumed. If the response variable is not listed in a CLASS statement and a response distribution is not specified in a DISTRIBUTION= option, then a normal distribution that uses the default or specified link function is assumed.

The multinomial distribution is a generalization of the binary distribution and allows for more than two outcome categories. Because there are more than two possible outcomes for the multinomial distribution, the terminology of "successes," "failures," "events," and "nonevents" no longer applies. With multinomial data, these outcomes are generically referred to as "categories" or levels.

Whenever the GENSELECT procedure determines that the response variable is listed in a CLASS statement and has more than two levels (unless the events/trials syntax is used), it fits the model as if the data had arisen from a multinomial distribution. By default, the procedure then assumes that the response categories are ordered, and it fits a cumulative link model by applying the default or specified link function.

If the response categories are unordered, then you should fit a generalized logit model by choosing LINK=GLOGIT in the MODEL statement.

Response Probability Distribution Functions

Beta Distribution

$$
\begin{aligned}
f(y) &= \frac{\Gamma(\phi)}{\Gamma(\mu\phi)\Gamma((1-\mu)\phi)} y^{\mu\phi-1}(1-y)^{(1-\mu)\phi-1} \quad \text{for } 0 < y < 1 \\
\mathrm{E}(Y) &= \mu \\
\mathrm{Var}(Y) &= \mu(1-\mu)/(1+\phi), \quad \phi > 0
\end{aligned}
$$

This parameterization of the beta distribution comes from Ferrari and Cribari-Neto (2004).

Binary Distribution

$$
\begin{aligned}
f(y) &= \begin{cases} p & \text{for } y = 1 \\ 1-p & \text{for } y = 0 \end{cases} \\
\mathrm{E}(Y) &= p \\
\mathrm{Var}(Y) &= p(1-p)
\end{aligned}
$$

Binomial Distribution

$$
\begin{aligned}
f(y) &= \binom{n}{r} \mu^r (1-\mu)^{n-r} \quad \text{for } y = \frac{r}{n},\ r = 0, 1, 2, \ldots, n \\
\mathrm{E}(Y) &= \mu \\
\mathrm{Var}(Y) &= \frac{\mu(1-\mu)}{n}
\end{aligned}
$$

Exponential Distribution

$$
\begin{aligned}
f(y) &= \frac{1}{\mu} \exp\left(-\frac{y}{\mu}\right) \quad \text{for } 0 < y < \infty \\
\mathrm{E}(Y) &= \mu \\
\mathrm{Var}(Y) &= \mu^2
\end{aligned}
$$

The exponential distribution is a special case of the gamma where $\nu = 1$.

Gamma Distribution

$$
\begin{aligned}
f(y) &= \frac{1}{\Gamma(\nu) y} \left(\frac{y\nu}{\mu}\right)^{\nu} \exp\left(-\frac{y\nu}{\mu}\right) \quad \text{for } 0 < y < \infty \\
\phi &= \frac{1}{\nu} \\
\mathrm{E}(Y) &= \mu \\
\mathrm{Var}(Y) &= \frac{\mu^2}{\nu}, \quad \nu > 0
\end{aligned}
$$

For the gamma distribution, $\nu = \frac{1}{\phi}$ is the estimated dispersion parameter that is displayed in the output. The parameter ν is sometimes called the gamma index parameter.

Generalized Poisson Distribution

$$
\begin{aligned}
\xi &= 1 - \exp(-\phi), \quad \phi \geq 0 \\
\mu^* &= \mu - \xi(\mu - y) \\
f(y) &= (\mu^*)^{(y-1)} \frac{((1-\xi)\mu)}{\Gamma(y+1)} \exp(-\mu^*) \quad \text{for } y = 0, 1, 2, \ldots \\
\mathrm{E}(Y) &= \mu \\
\mathrm{Var}(Y) &= \frac{\mu}{(1-\xi)^2}
\end{aligned}
$$

The generalized Poisson distribution provides an overdispersed alternative to the Poisson distribution; $\phi = \xi = 0$ produces the mass function of a regular Poisson random variable. For more information about the generalized Poisson distribution and a comparison with the negative binomial distribution, see Joe and Zhu (2005). For the generalized Poisson distribution, ϕ is the estimated dispersion parameter that is displayed in the output.

Geometric Distribution

$$
\begin{aligned}
f(y) &= \frac{\mu^y}{(1+\mu)^{y+1}} \quad \text{for } y = 0, 1, 2, \ldots \\
\mathrm{E}(Y) &= \mu \\
\mathrm{Var}(Y) &= \mu + \mu^2
\end{aligned}
$$

The geometric distribution is a special case of the negative binomial where $\phi = 1$.

Inverse Gaussian Distribution

$$
\begin{aligned}
f(y) &= \frac{1}{\sqrt{2\pi y^3 \sigma}} \exp\left[-\frac{1}{2y} \left(\frac{y-\mu}{\mu\sigma} \right)^2 \right] \quad \text{for } 0 < y < \infty \\
\phi &= \sigma^2 \\
\mathrm{Var}(Y) &= \phi\mu^3
\end{aligned}
$$

Lognormal Distribution

$$
\begin{aligned}
f(y) &= \frac{1}{\sqrt{2\pi}\sigma y}\exp\left[-\frac{1}{2}\left(\frac{\ln(y)-\mu}{\sigma}\right)^2\right] \quad \text{for } 0 < y < \infty \\
\phi &= \sigma^2 \\
\mathrm{E}(Y) &= \exp(\mu)\sqrt{\exp(\phi)} \\
\mathrm{Var}(Y) &= \exp(2\mu)\exp\{\phi\}(\exp(\phi)-1)
\end{aligned}
$$

If Y has the preceding lognormal distribution, then $X = \log(Y)$ has a normal distribution with mean μ and variance σ^2.

Multinomial Distribution

$$
f(y_1, y_2, \ldots, y_k) = \frac{m!}{y_1! y_2! \cdots y_k!} p_1^{y_1} p_2^{y_2} \cdots p_k^{y_k}
$$

Negative Binomial Distribution

$$
\begin{aligned}
f(y) &= \frac{\Gamma(y+1/k)}{\Gamma(y+1)\Gamma(1/k)}\frac{(k\mu)^y}{(1+k\mu)^{y+1/k}} \quad \text{for } y = 0, 1, 2, \ldots \\
\phi &= k \\
\mathrm{E}(Y) &= \mu \\
\mathrm{Var}(Y) &= \mu + \phi\mu^2
\end{aligned}
$$

For the negative binomial distribution, k is the estimated dispersion parameter that is displayed in the output.

Normal Distribution

$$
\begin{aligned}
f(y) &= \frac{1}{\sqrt{2\pi}\sigma}\exp\left[-\frac{1}{2}\left(\frac{y-\mu}{\sigma}\right)^2\right] \quad \text{for } -\infty < y < \infty \\
\phi &= \sigma^2 \\
\mathrm{E}(Y) &= \mu \\
\mathrm{Var}(Y) &= \phi
\end{aligned}
$$

Poisson Distribution

$$\begin{aligned} f(y) &= \frac{\mu^y \mathrm{e}^{-\mu}}{y!} \quad \text{for } y = 0, 1, 2, \ldots \\ \mathrm{E}(Y) &= \mu \\ \mathrm{Var}(Y) &= \mu \end{aligned}$$

(Shifted) *t* Distribution

$$\begin{aligned} z &= \frac{\Gamma(0.5(\nu+1))}{\sqrt{\pi\nu\phi}\,\Gamma(0.5\nu)} \\ f(y) &= z\left(1 + \frac{(y-\mu)^2}{\nu\phi}\right)^{-0.5(\nu+1)} \quad \text{for } -\infty < y < \infty \\ \mathrm{E}(Y) &= \mu \\ \mathrm{Var}(Y) &= \phi\nu/(\nu-2), \quad \nu \geq 3 \end{aligned}$$

You can specify the degrees of freedom parameter ν in the DISTRIBUTION=*t*(ν) option. If you do not specify it, a value of $\nu = 3$ is used.

Note that this form of the *t* distribution is not a noncentral distribution, but that of a shifted central *t* random variable.

Tweedie Distribution

The Tweedie model is a generalized linear model from the exponential family. The Tweedie distribution is characterized by three parameters: the mean μ, the dispersion ϕ, and the power p. The variance of the distribution is $\phi\mu^p$. For values of p in the range $1 < p < 2$, a Tweedie random variable can be represented as a Poisson sum of gamma distributed random variables. That is,

$$Y = \sum_{i=1}^{N} Y_i$$

where N has a Poisson distribution that has mean $\lambda = \frac{\mu^{2-p}}{\phi(2-p)}$ and the Y_is have independent, identical gamma distributions, each of which has an expected value $\mathrm{E}(Y_i) = \phi(2-p)\mu^{p-1}$ and an index parameter $\nu_i = \frac{2-p}{p-1}$.

In this case, Y has a discrete mass at 0, $\Pr(Y = 0) = \Pr(N = 0) = \exp(-\lambda)$, and the probability density of Y, $f(y)$, is represented by an infinite series for $y > 0$. The GENSELECT procedure restricts the power parameter to satisfy $1.1 <= p$ for numerical stability in model fitting. The Tweedie distribution does not have a general closed form representation for all values of p. For more information about the Tweedie distribution, see Frees (2010).

The distribution mean and variance are:

$$\begin{aligned} \mathrm{E}(Y) &= \mu \\ \mathrm{Var}(Y) &= \phi\mu^p \end{aligned}$$

Weibull Distribution

$$\begin{aligned} f(y) &= \frac{1}{\mu\phi}\left(\frac{y}{\mu}\right)^{\frac{1}{\phi}-1}\exp\left(-\frac{y}{\mu}\right)^{\frac{1}{\phi}} \quad \text{for } 0 < y < \infty \\ \mathrm{E}(Y) &= \mu\Gamma(1+\phi) \\ \mathrm{Var}(Y) &= \mu^2[\Gamma(1+2\phi) - \Gamma^2(1+\phi)], \quad \phi > 0 \end{aligned}$$

Log-Likelihood Functions

The GENSELECT procedure forms the log-likelihood functions of the various models as

$$L(\boldsymbol{\mu};\mathbf{y}) = \sum_{i=1}^{n} f_i\, l(\mu_i; y_i, w_i)$$

where $l(\mu_i; y_i, w_i)$ is the log-likelihood contribution of the ith observation that has weight w_i, and f_i is the value of the frequency variable. For the determination of w_i and f_i, see the WEIGHT and FREQ statements. The individual log-likelihood contributions for the various distributions are as follows.

In the following, the mean parameter μ_i for each observation i is related to the regression parameters $\boldsymbol{\beta}_i$ through the linear predictor $\eta_i = \mathbf{x}_i'\boldsymbol{\beta}$ by

$$\mu_i = g^{-1}(\eta_i)$$

where g is the link function.

Beta Distribution

$$\begin{aligned} l(\mu_i; y_i, w_i) = \log&\left\{\frac{\Gamma(\phi/w_i)}{\Gamma(\mu_i\phi/w_i)\Gamma((1-\mu_i)\phi/w_i)}\right\} \\ &+ (\mu_i\phi/w_i - 1)\log\{y_i\} \\ &+ ((1-\mu_i)\phi/w_i - 1)\log\{1-y_i\} \end{aligned}$$

where ϕ is the dispersion parameter that is displayed in the output.

Binary Distribution

$$\begin{aligned} \eta_i &= \mathbf{x}_i'\boldsymbol{\beta} \\ \mu_i &= g^{-1}(\eta_i) \\ l(\mu_i; y_i) &= y_i \log\{\mu_i\} + (1 - y_i) \log\{1 - \mu_i\} \end{aligned}$$

Here, μ_i is the probability of an event, and the variable y_i takes the value 1 for an event and 0 for a nonevent. The inverse link function $g^{-1}(\cdot)$ maps from the scale of the linear predictor η_i to the scale of the mean. For example, for the logit link (the default),

$$\mu_i = \frac{\exp\{\eta_i\}}{1 + \exp\{\eta_i\}}$$

You can control which binary outcome in your data is modeled as the event by specifying the *response-options* in the MODEL statement, and you can choose the link function by specifying the LINK= option in the MODEL statement.

If you specify a WEIGHT statement and w_i denotes the weight for the current observation, the log-likelihood function is computed as

$$l(\mu_i; y_i, w_i) = w_i l(\mu_i; y_i)$$

Binomial Distribution

$$\begin{aligned} \eta_i &= \mathbf{x}_i'\boldsymbol{\beta} \\ \mu_i &= g^{-1}(\eta_i) \\ l(\mu_i; y_i, w_i) &= w_i \left(y_i \log\{\mu_i\} + (n_i - y_i) \log\{1 - \mu_i\}\right) \\ &\quad + w_i \left(\log\{\Gamma(n_i + 1)\} - \log\{\Gamma(y_i + 1)\} - \log\{\Gamma(n_i - y_i + 1)\}\right) \end{aligned}$$

where y_i and n_i are the values of the events and trials of the ith observation, respectively. The value μ_i measures the probability of events (successes) in the underlying Bernoulli distribution whose aggregate follows the binomial distribution.

Exponential Distribution

$$l(\mu_i; y_i, w_i) = w_i \log\left\{\frac{w_i y_i}{\mu_i}\right\} - \frac{w_i y_i}{\mu_i} - \log\{y_i \Gamma(w_i)\}$$

Gamma Distribution

$$\begin{aligned}\eta_i &= \mathbf{x}_i'\boldsymbol{\beta}\\ \mu_i &= g^{-1}(\eta_i)\\ l(\mu_i; y_i, w_i) &= \frac{w_i}{\phi}\log\left(\frac{w_i y_i}{\phi\mu_i}\right) - \frac{w_i y_i}{\phi\mu_i} - \log(y_i) - \log\left(\Gamma\left(\frac{w_i}{\phi}\right)\right)\end{aligned}$$

For the gamma distribution, $\nu = \frac{1}{\phi}$ is the estimated dispersion parameter that is displayed in the output.

Generalized Poisson Distribution

$$\begin{aligned}\xi_i &= (1 - \exp\{-\phi\}/w_i)\\ \mu_i^* &= \mu_i - \xi(\mu_i - y_i)\\ l(\mu_i^*; y_i, w_i) &= \log\{\mu_i^* - \xi_i y_i\} + (y_i - 1)\log\{\mu_i^*\}\\ &\quad - \mu_i^* - \log\{\Gamma(y_i + 1)\}\end{aligned}$$

where ϕ is the dispersion parameter that is displayed in the output.

Geometric Distribution

$$\begin{aligned}l(\mu_i; y_i, w_i) &= y_i \log\left\{\frac{\mu_i}{w_i}\right\} - (y_i + w_i)\log\left\{1 + \frac{\mu_i}{w_i}\right\}\\ &\quad + \log\left\{\frac{\Gamma(y_i + w_i)}{\Gamma(w_i)\Gamma(y_i + 1)}\right\}\end{aligned}$$

Inverse Gaussian Distribution

$$\begin{aligned}\eta_i &= \mathbf{x}_i'\boldsymbol{\beta}\\ \mu_i &= g^{-1}(\eta_i)\\ l(\mu_i; y_i, w_i) &= -\frac{1}{2}\left[\frac{w_i(y_i - \mu_i)^2}{y_i\mu_i^2\phi} + \log\left(\frac{\phi y_i^3}{w_i}\right) + \log(2\pi)\right]\end{aligned}$$

where ϕ is the dispersion parameter that is displayed in the output.

Lognormal Distribution

$$\begin{aligned}\eta_i &= \mathbf{x}_i'\boldsymbol{\beta}\\ \mu_i &= g^{-1}(\eta_i)\\ l(\mu_i; y_i, w_i) &= -\frac{1}{2}\left[2\log\{y_i\} + \frac{w_i(y_i - \mu_i)^2}{\phi} + \log\left(\frac{\phi}{w_i}\right) + \log(2\pi)\right]\end{aligned}$$

where ϕ is the dispersion parameter that is displayed in the output. If $X = \log\{Y\}$ has a normal distribution with mean μ and variance ϕ, then Y has the log-likelihood function $l(\mu_i; y_i, w_i)$.

Multinomial Distribution

The multinomial distribution that the GENSELECT procedure models is a generalization of the binary distribution; it is the distribution of a single draw from a discrete distribution that has J possible values. Thus, the log-likelihood function for the ith observation is

$$l(\boldsymbol{\pi}_i; \mathbf{y}_i, w_i) = w_i \sum_{j=1}^{J} y_{ij} \log\{\pi_{ij}\}$$

In this expression, J denotes the number of response categories (the number of possible outcomes) and π_{ij} is the probability that the ith observation takes on the response value associated with category j. The category probabilities must satisfy

$$\sum_{j=1}^{J} \pi_j = 1$$

and the constraint is satisfied by modeling $J - 1$ categories.

If your response Y has more than two values and they can be ordered (for example, $Y \in \{1, \ldots, J-1, J\}$), then the GENSELECT procedure fits ordinal response models of the form

$$g(\pi_j \mid \mathbf{x}) = \alpha_j + \boldsymbol{\beta}'\mathbf{x}, \quad j = 1, \ldots, J-1$$

where g is the link function and $\pi_j = \Pr(Y \leq j)$ are cumulative probabilities of the ordered response categories.

If your response Y has more than two values and they have no natural ordering, then the GENSELECT procedure fits a generalized or baseline-category logit model, which has the form

$$\log\left(\frac{\Pr(Y = j \mid \mathbf{x})}{\Pr(Y = J \mid \mathbf{x})}\right) = \alpha_j + \boldsymbol{\beta}_j'\mathbf{x}, \quad j = 1, \ldots, J-1$$

where the $\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_{J-1}$ are $J - 1$ vectors of slope parameters. These models are a special case of the discrete choice or conditional logit models introduced by McFadden (1974).

In models that have ordered response categories, the probabilities are expressed in cumulative form, so the last category is redundant. In generalized logit models (multinomial models that have unordered categories), one category is chosen as the reference category, and the linear predictor in the reference category is set to 0. For more information, see the REF= *response-option* in the MODEL statement.

Negative Binomial Distribution

$$\begin{aligned}
\eta_i &= \mathbf{x}_i'\boldsymbol{\beta} \\
\mu_i &= g^{-1}(\eta_i) \\
l(\mu_i; y_i, w_i) &= y_i \log\left(\frac{k\mu_i}{w_i}\right) - (y_i + w_i/k)\log\left(1 + \frac{k\mu_i}{w_i}\right) + \log\left(\frac{\Gamma(y_i + w_i/k)}{\Gamma(y_i + 1)\Gamma(w_i/k)}\right)
\end{aligned}$$

where k is the negative binomial dispersion parameter that is displayed in the output.

Normal Distribution

$$\begin{aligned} \eta_i &= \mathbf{x}_i'\boldsymbol{\beta} \\ \mu_i &= g^{-1}(\eta_i) \\ l(\mu_i; y_i, w_i) &= -\frac{1}{2}\left[\frac{w_i(y_i-\mu_i)^2}{\phi} + \log\left(\frac{\phi}{w_i}\right) + \log(2\pi)\right] \end{aligned}$$

where ϕ is the dispersion parameter that is displayed in the output.

Poisson Distribution

$$\begin{aligned} \eta_i &= \mathbf{x}_i'\boldsymbol{\beta} \\ \mu_i &= g^{-1}(\eta_i) \\ l(\mu_i; y_i, w_i) &= w_i[y_i \log(\mu_i) - \mu_i - \log(y_i!)] \end{aligned}$$

t Distribution

$$\begin{aligned} z_i &= -0.5\log\{\phi/w_i\} + \log\{\Gamma(0.5(\nu+1)\} \\ &\quad - \log\{\Gamma(0.5\nu)\} - 0.5 \times \log\{\pi\nu\} \\ l(\mu_i; y_i, w_i) &= -\left(\frac{\nu+1}{2}\right)\log\left\{1 + \frac{w_i}{\nu}\frac{(y_i-\mu_i)^2}{\phi}\right\} + z_i \end{aligned}$$

where ϕ is the dispersion parameter that is displayed in the output.

Tweedie Distribution

The Tweedie distribution does not in general have a closed form log-likelihood function in terms of the mean, dispersion, and power parameters. The form of the log likelihood is

$$L(\boldsymbol{\mu}; \mathbf{y}) = \sum_{i=1}^{n} f_i\, l(\mu_i; y_i, w_i)$$

where

$$l(\mu_i, y_i, w_i) = \log(f(y_i; \mu_i, p, \frac{\phi}{w_i}))$$

and $f(y, \mu, p, \phi)$ is the Tweedie probability distribution, which is described in the section "Tweedie Distribution" on page 753. Evaluation of the Tweedie log likelihood for model fitting is performed numerically as described in Dunn and Smyth (2005, 2008).

Quasi-likelihood

The extended quasi-likelihood (EQL) is constructed according to the definition of McCullagh and Nelder (1989, Chapter 9) as

$$Q_p(\mathbf{y}, \boldsymbol{\mu}, \phi, p) = \sum_i q(y_i, \mu_i, \phi, p)$$

where the contribution from an observation is

$$q(y_i, \mu_i, \phi, p) = -0.5 \log(2\pi \frac{\phi}{w_i} y_i^p) - w_i \left(\frac{y_i^{2-p} - (2-p) y_i \mu_i^{1-p} + (1-p)\mu_i^{2-p}}{(1-p)(1-p)} \right) / \phi$$

where $1 < p < 2$. This EQL is used in computing initial values for the iterative maximization of the Tweedie log likelihood, as specified using the OPTMETHOD= Tweedie option in Table 14.7. If you specify the OPTMETHOD=EQL *Tweedie-optimization-option* in Table 14.7, then the parameter estimates are computed by using the EQL instead of the log likelihood.

Weibull Distribution

$$l(\mu_i; y_i, w_i) = -w_i \left\{ \frac{\phi - 1}{\phi} \log\left(\frac{y_i}{\mu_i}\right) + \log(\mu_i \phi) + \exp\left[\log\left(\frac{y_i}{\mu_i}\right) / \phi \right] \right\}$$

where ϕ is the dispersion parameter that is displayed in the output.

Existence of Maximum Likelihood Estimates

The likelihood equation for a logistic regression model does not always have a finite solution. Sometimes there is a nonunique maximum on the boundary of the parameter space, at infinity. The existence, finiteness, and uniqueness of maximum likelihood estimates for the logistic regression model depend on the patterns of data points in the observation space (Albert and Anderson 1984; Santner and Duffy 1986).

Consider a binary response model. Let Y_j be the response of the jth subject, and let $\mathbf{x}_j$ be the vector of explanatory variables (including the constant 1 that is associated with the intercept). There are three mutually exclusive and exhaustive types of data configurations:

Complete separation There is a complete separation of data points if there exists a vector $\mathbf{b}$ that correctly allocates all observations to their response groups; that is,

$$\begin{cases} \mathbf{b}'\mathbf{x}_j > 0 & Y_j = 0 \\ \mathbf{b}'\mathbf{x}_j < 0 & Y_j = 1 \end{cases}$$

This configuration produces nonunique infinite estimates. If the iterative process of maximizing the likelihood function is allowed to continue, then the log likelihood diminishes to 0 and the dispersion matrix becomes unbounded.

Quasi-complete separation The data are not completely separable, but there is a vector $\mathbf{b}$ such that

$$\begin{cases} \mathbf{b}'\mathbf{x}_j \geq 0 & Y_j = 0 \\ \mathbf{b}'\mathbf{x}_j \leq 0 & Y_j = 1 \end{cases}$$

and equality holds for at least one subject in each response group. This configuration also yields nonunique infinite estimates. If the iterative process of maximizing the likelihood function is allowed to continue, then the dispersion matrix becomes unbounded and the log likelihood diminishes to a nonzero constant.

Overlap If neither complete nor quasi-complete separation exists in the sample points, there is an overlap of sample points. In this configuration, the maximum likelihood estimates exist and are unique.

The GENSELECT procedure uses a simple empirical approach to recognize the data configurations that lead to infinite parameter estimates. The basis of this approach is that any convergence method of maximizing the log likelihood must yield a solution that indicates complete separation, if such a solution exists. Upon convergence, if the predicted response equals the observed response for every observation, there is a complete separation of data points.

If the data are not completely separated, if an observation is identified to have an extremely large probability (≥ 0.95) of predicting the observed response, and if there have been at least eight iterations, then there are two possible situations. First, there is overlap in the data set, the observation is an atypical observation of its own group, and the iterative process stopped when a maximum was reached. Second, there is quasi-complete separation in the data set, and the asymptotic dispersion matrix is unbounded. If any of the diagonal elements of the dispersion matrix for the standardized observation vector (all explanatory variables standardized to zero mean and unit variance) exceeds 5,000, then PROC GENSELECT declares quasi-complete separation; if any of the diagonal elements exceeds 1,000, then the procedure displays a message indicating that quasi-complete separation might be detectable by increasing the number of iterations. If either complete separation or quasi-complete separation is detected, a note is displayed in the procedure output.

Checking for quasi-complete separation is less foolproof than checking for complete separation. If neither type of separation is discovered and your parameter estimates have large standard errors, then your data might be separable. The NOCHECK option in the MODEL statement turns off the process of checking for infinite parameter estimates; the MINITER= option in the PROC GENSELECT statement increases the number of iterations.

The LASSO Method of Model Selection

LASSO Selection

The GENSELECT procedure implements the group LASSO method, which is described in the section "Group LASSO Selection" on page 87 in Chapter 2, "Shared Concepts." The current section provides some background about the LASSO method that you need in order to understand the group LASSO method.

LASSO (least absolute shrinkage and selection operator) selection arises from a constrained form of ordinary least squares regression in which the sum of the absolute values of the regression coefficients is constrained to be smaller than a specified parameter. More precisely, let $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m)$ denote the matrix of covariates, and let $\mathbf{y}$ denote the response. Then for a given parameter t, the LASSO regression coefficients $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_m)$ are the solution to the constrained least squares problem:

$$\min ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||^2 \qquad \text{subject to} \qquad \sum_{j=1}^{m} |\beta_j| \leq t$$

For generalized linear models, the LASSO regression coefficients $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_m)$ are the solution to the constrained optimization problem,

$$\min\{-L(\boldsymbol{\mu}; \mathbf{y})\} \qquad \text{subject to} \qquad \sum_{j=1}^{m} |\beta_j| \leq t$$

where L is the log-likelihood function defined in the section "Log-Likelihood Functions" on page 754.

Provided that the LASSO parameter t is small enough, some of the regression coefficients will be exactly zero. Hence, you can think of the LASSO method as selecting a subset of the regression coefficients for each LASSO parameter. By increasing the LASSO parameter in discrete steps, you obtain a sequence of regression coefficients for which the nonzero coefficients at each step correspond to selected parameters. For more information about the LASSO method, see, for example, Hastie, Tibshirani, and Friedman (2009).

Some distributions involve a dispersion parameter (the parameter ϕ in the expressions for the log likelihood). These parameters are not estimated by the LASSO optimization algorithm, and they are set to either the default value or a value that you specify. You can use the MODEL statement option PHI= to set the dispersion to a fixed value.

Generalized Estimating Equations

The analysis of correlated data that arise from repeated measurements when the measurements are assumed to be multivariate normal has been studied extensively. However, the normality assumption might not always be reasonable; for example, you must use a different methodology in the data analysis when the responses are discrete and correlated. Generalized estimating equations (GEEs) provide a practical method with reasonable statistical efficiency to analyze such data.

Liang and Zeger (1986) introduced GEEs as a method of dealing with correlated data when, except for the correlation among responses, the data can be modeled as a generalized linear model. For example, in many cases you can model correlated binary and count data in this way. For more information about GEEs, see Hardin and Hilbe (2003), Diggle, Liang, and Zeger (1994), and Lipsitz et al. (1994).

Let y_{ij}, $j = 1, \ldots, n_i$, $i = 1, \ldots, K$, represent the jth measurement on the ith subject. There are n_i measurements on subject i and $\sum_{i=1}^{K} f_i\, n_i$ total measurements, where f_i is the cluster frequency that you specify in the FREQ statement. If you omit the FREQ statement, then $f_i = 1$ for all observations. The frequencies must be the same for all observations within each cluster.

You model correlated data by using the same link function and linear predictor specification that you use to model independent data. You use the same variance functions as in the independent case, but you must also specify a structure for the correlation between the repeated measurements. In the following material, $v(\mu)$ denotes the variance function for the specified distribution. Note that for the negative binomial distribution, the variance function is defined as $v(\mu) = \mu + k\mu^2$, where k is the maximum likelihood estimate of the negative binomial dispersion parameter k or the fixed value of k that you specify using the PHI= option in the MODEL statement.

Let the vector of measurements on the ith subject be $\mathbf{Y}_i = [y_{i1}, \ldots, y_{in_i}]'$, where the corresponding vector of means is $\boldsymbol{\mu}_i = [\mu_{i1}, \ldots, \mu_{in_i}]'$, and let $\mathbf{V}_i$ be the covariance matrix of $\mathbf{Y}_i$. Let the vector of independent, or explanatory, variables for the jth measurement on the ith subject be $\mathbf{x}_{ij} = [x_{ij1}, \ldots, x_{ijp}]'$. The generalized estimating equation of Liang and Zeger (1986) for estimating the $p \times 1$ vector of regression parameters $\boldsymbol{\beta}$ is an extension of the independence estimating equation to correlated data and is given by

$$\mathbf{S}(\boldsymbol{\beta}) = \sum_{i=1}^{K} f_i \mathbf{D}_i' \mathbf{V}_i^{-1} (\mathbf{Y}_i - \boldsymbol{\mu}_i(\boldsymbol{\beta})) = \mathbf{0}$$

where

$$\mathbf{D}_i = \frac{\partial \boldsymbol{\mu}_i}{\partial \boldsymbol{\beta}}$$

The relationship between the linear predictor and the mean is specified by

$$g(\mu_{ij}) = \mathbf{x}_{ij}{}'\boldsymbol{\beta}$$

where g is the link function, and so the $p \times n_i$ matrix of partial derivatives of the mean with respect to the regression parameters for the ith subject is given by

$$\mathbf{D}_i' = \frac{\partial \boldsymbol{\mu}_i'}{\partial \boldsymbol{\beta}} = \begin{bmatrix} \frac{x_{i11}}{g'(\mu_{i1})} & \cdots & \frac{x_{in_i1}}{g'(\mu_{in_i})} \\ \vdots & & \vdots \\ \frac{x_{i1p}}{g'(\mu_{i1})} & \cdots & \frac{x_{in_ip}}{g'(\mu_{in_i})} \end{bmatrix}$$

Working Correlation Matrix

Let $\mathbf{R}_i(\boldsymbol{\alpha})$ be an $n_i \times n_i$ "working" correlation matrix that is fully specified by the vector of parameters $\boldsymbol{\alpha}$. The covariance matrix of $\mathbf{Y}_i$ is modeled as

$$\mathbf{V}_i = \phi \mathbf{A}_i^{\frac{1}{2}} \mathbf{W}_i^{-\frac{1}{2}} \mathbf{R}(\boldsymbol{\alpha}) \mathbf{W}_i^{-\frac{1}{2}} \mathbf{A}_i^{\frac{1}{2}}$$

where $\mathbf{A}_i$ is an $n_i \times n_i$ diagonal matrix whose jth diagonal element is $v(\mu_{ij})$, and $\mathbf{W}_i$ is an $n_i \times n_i$ diagonal matrix whose jth diagonal is w_{ij}, where w_{ij} is a weight specified in the WEIGHT statement. If there is no WEIGHT statement, then $w_{ij} = 1$ for all i and j. If $\mathbf{R}_i(\boldsymbol{\alpha})$ is the true correlation matrix of $\mathbf{Y}_i$, then $\mathbf{V}_i$ is the true covariance matrix of $\mathbf{Y}_i$.

The working correlation matrix is usually unknown and must be estimated. You estimate it in the iterative fitting process by using the current value of the parameter vector $\boldsymbol{\beta}$ to compute appropriate functions of the Pearson residual

$$e_{ij} = \frac{y_{ij} - \mu_{ij}}{\sqrt{v(\mu_{ij})/w_{ij}}}$$

If you specify the working correlation as $\mathbf{R_0} = \mathbf{I}$, which is the identity matrix, the GEE reduces to the independence estimating equation.

Table 14.15 shows the working correlation types that the GENSELECT procedure supports and the estimators that are used to estimate the working correlations.

Table 14.15 Working Correlation Structures and Correlation Parameter Estimators

Working Correlation Structure	Estimator
Autoregressive	
AR(1)	
$\text{Corr}(Y_{ij}, Y_{i,j+t}) = \alpha^t$ for $t = 0, 1, 2, \ldots, n_i - j$	$\hat{\alpha} = \frac{1}{(K_1 - p)\phi} \sum_{i=1}^{K} f_i \sum_{j \le n_i - 1} e_{ij} e_{i,j+1}$ $K_1 = \sum_{i=1}^{K} f_i (n_i - 1)$
Exchangeable	
$\text{Corr}(Y_{ij}, Y_{ik}) = \begin{cases} 1 & j = k \\ \alpha & j \neq k \end{cases}$	$\hat{\alpha} = \frac{1}{(N^* - p)\phi} \sum_{i=1}^{K} f_i \sum_{j<k} e_{ij} e_{ik}$ $N^* = 0.5 \sum_{i=1}^{K} f_i n_i (n_i - 1)$
Independence	
$\text{Corr}(Y_{ij}, Y_{ik}) = \begin{cases} 1 & j = k \\ 0 & j \neq k \end{cases}$	The working correlation is not estimated in this case.
***m*-dependent**	
$\text{Corr}(Y_{ij}, Y_{i,j+t}) = \begin{cases} 1 & t = 0 \\ \alpha_t & t = 1, 2, \ldots, m \\ 0 & t > m \end{cases}$	$\hat{\alpha}_t = \frac{1}{(K_t - p)\phi} \sum_{i=1}^{K} f_i \sum_{j \le n_i - t} e_{ij} e_{i,j+t}$ $K_t = \sum_{i=1}^{K} f_i (n_i - t)$
Unstructured	
$\text{Corr}(Y_{ij}, Y_{ik}) = \begin{cases} 1 & j = k \\ \alpha_{jk} & j \neq k \end{cases}$	$\hat{\alpha}_{jk} = \frac{1}{(K - p)\phi} \sum_{i=1}^{K} f_i e_{ij} e_{ik}$

Dispersion Parameter

The dispersion parameter ϕ is estimated by

$$\hat{\phi} = \frac{1}{N - p} \sum_{i=1}^{K} f_i \sum_{j=1}^{n_i} e_{ij}^2$$

where $N = \sum_{i=1}^{K} f_i n_i$ is the total number of measurements and p is the number of regression parameters.

PROC GENSELECT reports $\hat{\phi}$ as the dispersion parameter in the output table "Analysis of GEE Parameter Estimates Model-Based Standard Error Estimates." The dispersion parameter is not included for models that use the binomial, binary, or Poisson distribution. If a fixed-scale parameter is specified using the PHI= option in the MODEL statement, then the fixed value is used in estimating the model-based covariance matrix and standard errors.

Fitting Algorithm

The following algorithm fits the specified model by using GEEs. Note that this is not in general a likelihood-based method of estimation; inferences based on likelihoods are not possible for GEE methods.

1. Compute an initial estimate of $\boldsymbol{\beta}$ by using an ordinary generalized linear model, assuming independence.
2. Compute the working correlations $\mathbf{R}$ on the basis of the standardized residuals, the current $\boldsymbol{\beta}$, and the assumed structure of $\mathbf{R}$.
3. Compute an estimate of the covariance:

$$\mathbf{V}_i = \phi \mathbf{A}_i^{\frac{1}{2}} \mathbf{W}_i^{-\frac{1}{2}} \hat{\mathbf{R}}(\boldsymbol{\alpha}) \mathbf{W}_i^{-\frac{1}{2}} \mathbf{A}_i^{\frac{1}{2}}$$

4. Update $\boldsymbol{\beta}$:

$$\boldsymbol{\beta}_{r+1} = \boldsymbol{\beta}_r + \left[\sum_{i=1}^{K} f_i \frac{\partial \boldsymbol{\mu}_i}{\partial \boldsymbol{\beta}}' \mathbf{V}_i^{-1} \frac{\partial \boldsymbol{\mu}_i}{\partial \boldsymbol{\beta}} \right]^{-1} \left[\sum_{i=1}^{K} f_i \frac{\partial \boldsymbol{\mu}_i}{\partial \boldsymbol{\beta}}' \mathbf{V}_i^{-1} (\mathbf{Y}_i - \boldsymbol{\mu}_i) \right]$$

5. Repeat steps 2–4 until convergence.

Missing Data

See Diggle, Liang, and Zeger (1994, Chapter 11) for a discussion of missing values in longitudinal data. Suppose you intend to take measurements $Y_{i1}, \ldots, Y_{in}$ for the ith unit. Missing values for which Y_{ij} are missing whenever Y_{ik} is missing for all $j \geq k$ are called *dropouts*. Otherwise, missing values that are intermixed with nonmissing values are called *intermittent* missing values. PROC GENSELECT can estimate the working correlation from data that contain both types of missing values by using the *all available pairs* method, in which all nonmissing pairs of data are used in the moment estimators of the working correlation parameters that are defined previously. The resulting covariances and standard errors are valid under the missing completely at random (MCAR) assumption.

For example, for the unstructured working correlation model,

$$\hat{\alpha}_{jk} = \frac{1}{(K' - p)\phi} \sum f_i e_{ij} e_{ik}$$

where the sum is over the units that have nonmissing measurements at times j and k, and K' is the number of units that have nonmissing measurements at j and k. Estimates of the parameters for other working correlation types are computed in a similar manner, using available nonmissing pairs in the appropriate moment estimators.

The contribution of the ith unit to the parameter update equation is computed by omitting the elements of $(\mathbf{Y}_i - \boldsymbol{\mu}_\mathbf{i})$, the columns of $\mathbf{D}_i' = \frac{\partial \boldsymbol{\mu}}{\partial \boldsymbol{\beta}}'$, and the rows and columns of $\mathbf{V}_i$ that correspond to missing measurements.

Parameter Estimate Covariances

The *model-based* estimator of $\mathrm{Cov}(\hat{\boldsymbol{\beta}})$ is given by

$$\boldsymbol{\Sigma}_m(\hat{\boldsymbol{\beta}}) = \mathbf{I}_0^{-1}$$

where

$$\mathbf{I}_0 = \sum_{i=1}^{K} f_i \frac{\partial \boldsymbol{\mu}_i}{\partial \boldsymbol{\beta}}' \mathbf{V}_i^{-1} \frac{\partial \boldsymbol{\mu}_i}{\partial \boldsymbol{\beta}}$$

This is a consistent estimator of the covariance matrix of $\hat{\boldsymbol{\beta}}$ if the mean model and the working correlation matrix are correctly specified.

The estimator

$$\boldsymbol{\Sigma}_e = \mathbf{I}_0^{-1}\mathbf{I}_1\mathbf{I}_0^{-1}$$

is called the *empirical*, or *robust*, estimator of the covariance matrix of $\hat{\boldsymbol{\beta}}$, where

$$\mathbf{I}_1 = \sum_{i=1}^{K} f_i \frac{\partial \boldsymbol{\mu}_i}{\partial \boldsymbol{\beta}}' \mathbf{V}_i^{-1}\mathrm{Cov}(\mathbf{Y}_i)\mathbf{V}_i^{-1} \frac{\partial \boldsymbol{\mu}_i}{\partial \boldsymbol{\beta}}$$

It has the property of being a consistent estimator of the covariance matrix of $\hat{\boldsymbol{\beta}}$, even if the working correlation matrix is misspecified—that is, if $\mathrm{Cov}(\mathbf{Y}_i) \neq \mathbf{V}_i$. For further information about the robust variance estimate, see Zeger, Liang, and Albert (1988), Royall (1986), and White (1982). In computing $\boldsymbol{\Sigma}_e$, the parameters $\boldsymbol{\beta}$ and ϕ are replaced by estimates, and $\mathrm{Cov}(\mathbf{Y}_i)$ is replaced by the estimate

$$(\mathbf{Y}_i - \boldsymbol{\mu}_i(\hat{\boldsymbol{\beta}}))(\mathbf{Y}_i - \boldsymbol{\mu}_i(\hat{\boldsymbol{\beta}}))'$$

Quasi-likelihood Information Criterion

The quasi-likelihood information criterion (QIC) was developed by Pan (2001) as a modification of the Akaike information criterion (AIC) to apply to models that are fit by GEEs.

Define the quasi-likelihood under the independence working correlation assumption, which is evaluated using the parameter estimates under the working correlation R as

$$Q(\hat{\boldsymbol{\beta}}(R), \phi) = \sum_{i=1}^{K} f_i \sum_{j=1}^{n_i} Q(\hat{\boldsymbol{\beta}}(R), \phi; (Y_{ij}, \mathbf{X}_{ij}))$$

where the quasi-likelihood contribution of the jth observation in the ith cluster is defined in the section "Quasi-likelihood Functions" on page 766 and $\hat{\boldsymbol{\beta}}(R)$ are the parameter estimates obtained from GEEs that use the working correlation R.

QIC is defined as

$$\mathrm{QIC}(R) = -2Q(\hat{\boldsymbol{\beta}}(R), \phi) + 2\mathrm{trace}(\hat{\Omega}_I \hat{V}_R)$$

where $\hat{V}_R$ is the robust covariance estimate and $\hat{\Omega}_I$ is the inverse of the model-based covariance estimate under the independence working correlation assumption, evaluated at $\hat{\boldsymbol{\beta}}(R)$, which are the parameter estimates obtained from GEEs that use the working correlation R.

PROC GENSELECT also computes an approximation to QIC(R), which Pan (2001) defines as

$$\mathrm{QIC}_{\boldsymbol{u}}(R) = -2Q(\hat{\boldsymbol{\beta}}(R), \phi) + 2p$$

where p is the number of regression parameters.

Pan (2001) notes that QIC is appropriate for selecting regression models and working correlations, whereas $\mathrm{QIC}_{\boldsymbol{u}}$ is appropriate only for selecting regression models.

Quasi-likelihood Functions

For detailed discussions of quasi-likelihood functions, see McCullagh and Nelder (1989) and Hardin and Hilbe (2003). The contribution of observation j in cluster i to the quasi-likelihood function that is evaluated at the regression parameters $\boldsymbol{\beta}$ is given by $Q(\boldsymbol{\beta}, \phi; (Y_{ij}, \mathbf{X}_{ij})) = \frac{Q_{ij}}{\phi}$, where Q_{ij} is defined in the following list. These are used in the computation of the quasi-likelihood information criteria (QIC) for the goodness of fit of models that are fit using GEEs. The w_{ij} are prior weights, if any, that are specified in the WEIGHT or FREQ statement. Note that the definition here of the quasi-likelihood for the negative binomial differs from that given in McCullagh and Nelder (1989). The definition here allows the negative binomial quasi-likelihood to approach the Poisson as $k \to 0$.

- Binomial

$$Q_{ij} = w_{ij}[r_{ij}\log(p_{ij}) + (n_{ij} - r_{ij})\log(1 - p_{ij})]$$

- Gamma

$$Q_{ij} = -w_{ij}\left[\frac{y_{ij}}{\mu_{ij}} + \log(\mu_{ij})\right]$$

- Inverse Gaussian

$$Q_{ij} = \frac{w_{ij}(\mu_{ij} - .5y_{ij})}{\mu_{ij}^2}$$

- Negative binomial

$$Q_{ij} = w_{ij}\left[\log\Gamma\left(y_{ij} + \frac{1}{k}\right) - \log\Gamma\left(\frac{1}{k}\right) + y_{ij}\log\left(\frac{k\mu_{ij}}{1 + k\mu_{ij}}\right) + \frac{1}{k}\log\left(\frac{1}{1 + k\mu_{ij}}\right)\right]$$

- Normal

$$Q_{ij} = -\frac{1}{2}w_{ij}(y_{ij} - \mu_{ij})^2$$

- Poisson

$$Q_{ij} = w_{ij}(y_{ij}\log(\mu_{ij}) - \mu_{ij})$$

GEE analysis is not supported when you perform model selection, when you perform a partitioned analysis, or when you specify parameter restrictions.

Using a Preexisting Table Order

You can add group and order information to a table by using the **partition** action in the **table** action set. If you want to use the group and order information in a table to achieve a reproducible processing order and you also want to do a repeated measures analysis, the group information must follow specific rules. In particular, the variables that you specify in the **groupBy** parameter of the **partition** action must match the variables that you specify in the SUBJECT option of the REPEATED statement.

There are some variations to this requirement, depending on which type of subject effect you specify. If you specify a single-variable subject effect or an interaction subject effect, you must use all these variables in the **groupBy** parameter of the **partition** action. If you specify a nested subject effect, you must include the nested variable in the **groupBy** parameter. You can also include the nesting variables in the **groupBy** parameter, but you cannot include any other variables in the **groupBy** parameter.

If you use an order-sensitive working correlation structure, you do not need to use the variables that form the within-subject effect in the **orderBy** parameter of the **partition** action. These sets of variables can overlap, but it is not required that the preexisting table order correspond to the ordering for the within-subject effect.

Partition Fit Statistics

Specifying a PARTITION statement modifies the display of many tables by adding separate rows or columns for the training, validation, and test data tables. In addition, the "Fit Statistics" table displays the average square error, which is useful for assessing the model and which should be very similar for the different roles when the training data are representative of the other data partitions. For more information, see the section "Model Fit and Assessment Statistics" on page 767.

Model Fit and Assessment Statistics

The statistics that are defined in this section are useful for assessing the fit of the model to your data; they are displayed in the "Fit Statistics" table. The statistics are computed for each data role when you specify a PARTITION statement.

Information Criteria

The calculation of the information criteria uses the following formulas, where p denotes the number of effective parameters in the candidate model, F denotes the sum of frequencies used, and l is the log likelihood evaluated at the converged estimates:

$$\begin{aligned} \text{AIC} &= -2l + 2p \\ \text{AICC} &= \begin{cases} -2l + 2pF/(F-p-1) & \text{when } F > p+2 \\ -2l + 2p(p+2) & \text{otherwise} \end{cases} \\ \text{SBC} &= -2l + p\log(F) \end{aligned}$$

If you do not specify a FREQ statement, F equals n, the number of observations used.

These measures are most useful for comparing competing models that are not necessarily nested—that is, models that cannot be reduced to one another by simple constraints on the parameter space. Smaller values of the measures indicate better models.

Average Square Error

The average square error (ASE) is the average of the squared differences between the responses and the predictions. When you have a discrete number of response levels, the ASE is modified as shown in Table 14.16 (Brier 1950; Murphy 1973); it is also called the Brier score or Brier reliability.

Table 14.16 Average Square Error Computations

Response Type	ASE (Brier Score)
Binary	$\frac{1}{F}\sum_i f_i(y_i(1-\widehat{\pi}_i)^2 + (1-y_i)\widehat{\pi}_i^2)$
Binomial	$\frac{1}{F}\sum_i f_i(r_i/t_i - \widehat{\pi}_i)^2$
Multinomial	$\frac{1}{F}\sum_i f_i \sum_j (y_{ij} - \widehat{\pi}_{ij})^2$

In Table 14.16, $F = \sum_i f_i$, r_i is the number of events, t_i is the number of trials in binomial response models, $y_i = 1$ for events and 0 for nonevents in binary response models, and $\widehat{\pi}_i$ is the predicted probability of an event. For polytomous response models, $y_{ij} = 1$ if the ith observation has response level j, and π_{ij} is the model-predicted probability of response level j for observation i.

Predicted Values and Regression Diagnostics

You can produce observationwise predicted values, confidence limits, and regression diagnostics that were developed by Williams (1987) and Pregibon (1981) by specifying the OUTPUT statement. For multinomial response data, you can likewise produce observationwise predicted probabilities, confidence limits, and raw residuals.

Given a vector of covariates $\mathbf{x}_i$ for the ith observation in your data table, and the model-predicted parameter estimates $\hat{\boldsymbol{\beta}}$, you can write the linear predictor $\hat{\eta}_i = \mathbf{x}_i'\hat{\boldsymbol{\beta}}$. The predicted mean of the ith observation $\hat{\mu}_i(\hat{\boldsymbol{\beta}})$, or the model-predicted event probability $\hat{p}_i$, is $\mu_i(\hat{\boldsymbol{\beta}}) = \hat{p}_i = g^{-1}(\hat{\eta}_i)$, where you select the link function g by specifying the LINK= option. The variance function of the distribution is $V(\mu)$, and $\hat{\boldsymbol{\Sigma}}$ is the estimated covariance of $\hat{\boldsymbol{\beta}}$. Denote the frequency of the ith observation as f_i and the weight as w_i.

For ordinal response models, the predicted cumulative probabilities are computed by using the appropriate model-predicted intercept parameters $\hat{\alpha}_j$ and letting $\hat{\boldsymbol{\beta}}$ consist of the slope parameters: $\hat{\eta}_{ij} = g(\Pr(Y \le j\,|\mathbf{x}_i) = \hat{\alpha}_j + \mathbf{x}_i'\hat{\boldsymbol{\beta}}$ and $\hat{\pi}_{ij} = \Pr(Y \le j\,|\mathbf{x}_i) = g^{-1}(\hat{\eta}_{ij})$ for $1 \le j < J$.

For nominal response models, the predicted probabilities are computed by using the appropriate model-predicted intercept parameters $\hat{\alpha}_j$ and by letting $\hat{\boldsymbol{\beta}}_j$ consist of the slope parameters: $\hat{\eta}_{ij} = g(\Pr(Y = j\,|\mathbf{x}_i) = \hat{\alpha}_j + \mathbf{x}_i'\hat{\boldsymbol{\beta}}_j$ and $\hat{\pi}_{ij} = \Pr(Y = j\,|\mathbf{x}_i) = g^{-1}(\hat{\eta}_{ij})$ for $1 \le j < J$.

If you perform elastic net or LASSO selection, or if you specify the NOSTDERR or NOSTDERR option, then your analysis does not generate a covariance matrix, and you can produce only the predicted probabilities (individual, cumulative, and posterior when available) and the raw, Pearson, deviance, and working residuals.

Confidence Intervals

Approximate confidence intervals for predicted values can be computed as follows. You estimate the variance of the linear predictor by using the following formula:

$$\hat{\sigma}_x^2 = \mathbf{x}_i' \hat{\mathbf{\Sigma}} \mathbf{x}_i$$

For multinomial models, the variance also depends on the response function. Let $\boldsymbol{\delta}_j$ be a $(J-1)$ column vector whose jth entry is equal to 1 and all other entries are equal to 0. Redefine $\mathbf{x}_i = (\boldsymbol{\delta}_j', \mathbf{x}_i')'$, $\hat{\eta}_i = \hat{\eta}_{ij}$, and $\hat{\pi}_i = \hat{\pi}_{ij}$. Then

$$\hat{\sigma}^2(\hat{\eta}_i) = \mathbf{x}_i' \hat{\mathbf{\Sigma}} \mathbf{x}_i$$

The asymptotic $100(1-\alpha)\%$ confidence interval for η_i is

$$\left(\hat{\eta}_i - z_{\alpha/2}\hat{\sigma}(\hat{\eta}_i),\ \hat{\eta}_i + z_{\alpha/2}\hat{\sigma}(\hat{\eta}_i) \right)$$

where $z_{\alpha/2}$ is the $100(1-\alpha/2)$th percentile point of a standard normal distribution.

The predicted value and the $100(1-\alpha)\%$ confidence limits for p_i are obtained by back-transforming the corresponding measures for the linear predictor. So the confidence limits for p_i are

$$\left(g^{-1}\left[\hat{\eta}_i - z_{\alpha/2}\hat{\sigma}(\hat{\eta}_i)\right],\ g^{-1}\left[\hat{\eta}_i + z_{\alpha/2}\hat{\sigma}(\hat{\eta}_i)\right] \right)$$

Hat-Matrix Diagonals

The diagonal elements of the hat matrix are useful in detecting extreme points in the design space, where they tend to have larger values. For the generalized linear model, the variance of the ith individual observation is

$$v_i = \frac{\phi V(\mu_i)}{f_i w_i}$$

For the ith observation, let

$$w_{ei} = v_i^{-1}(g'(\mu_i))^{-2}$$

where $g'(\mu_i)$ is the derivative of the link function evaluated at μ_i. Let $\mathbf{W}$ be the diagonal matrix, with w_{ei} denoting the ith diagonal element, which is used in computing the expected information matrix. Define the leverage, or hat-matrix diagonal, h_i, as the ith diagonal element of the matrix

$$\mathbf{W}^{\frac{1}{2}}\mathbf{X}(\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}\mathbf{X}'\mathbf{W}^{\frac{1}{2}}$$

For binary or binomial models, if the estimated probability is extreme (less than 0.1 and greater than 0.9, approximately), then the hat-matrix diagonal might be greatly reduced in value. Consequently, when an observation has a very large or very small estimated probability, its leverage is not a good indicator of the observation's distance from the design space (Hosmer and Lemeshow 2000, p. 171).

Residuals

Residuals are useful in identifying observations that are not explained well by the model. For the binomial distribution, the raw residual is defined as

$$r_i = y_i/t_i - \hat{p}_i$$

where y_i is the number of event responses out of t_i trials for the ith observation. For single-trial syntax, $t_i = 1$ and $y_i = 1$ if the ordered response is 1 and $y_i = 0$ otherwise. For multinomial response data, the raw residual is

$$y_{ij} - \widehat{\pi}_{ij}$$

where $y_{ij} = 1$ if the ith observation has response level j and $y_{ij} = 0$ otherwise, and $\hat{\pi}_{ij}$ are the model-predicted probabilities of response level j for observation i.

For other generalized linear models, the raw residual is

$$r_i = y_i - \hat{p}_i$$

where y_i is the observed response and $\hat{p}_i$ is the predicted value. However, if you also specify the DIST=LOGNORMAL option, the raw residual is

$$r_i = \log(y_i) - \hat{p}_i$$

The Pearson residual is the square root of the ith observation's contribution to Pearson's chi-square:

$$r_{Pi} = r_i \sqrt{\frac{f_i w_i}{V(\mu_i)}}$$

The deviance residual is the square root of the contribution of the ith observation to the deviance, with the sign of the raw residual,

$$r_{Di} = (\text{sign}(r_i))\sqrt{d_i}$$

For example, for the binomial distribution,

$$d_i = 2 f_i w_i t_i \left[y_i/t_i \log\left(\frac{y_i/t_i}{\mu_i}\right) + (1 - y_i/t_i) \log\left(\frac{1 - y_i/t_i}{1 - \mu_i}\right)\right]$$

The working residual is

$$r_{Wi} = r_i \left(\frac{\partial \mu_i}{\partial \eta_i}\right)^{-1}$$

The Pearson residuals, standardized to have unit asymptotic variance, are

$$r_{SPi} = \frac{r_{Pi}}{\sqrt{1 - h_i}}$$

The deviance residuals, standardized to have unit asymptotic variance, are

$$r_{SDi} = \frac{r_{Di}}{\sqrt{1 - h_i}}$$

The likelihood residuals, which estimate components of a likelihood ratio test of deleting an individual observation, are a weighted combination of the standardized Pearson and deviance residuals,

$$r_{Li} = \text{sign}(r_i)\sqrt{h_i r_{SPi}^2 + (1 - h_i) r_{SDi}^2}$$

Other Regression Diagnostics

The CBAR statistic is a confidence interval displacement diagnostic that provides a scalar measure of the influence of an individual observation on $\widehat{\boldsymbol{\beta}}$. This diagnostic is based on the same idea as Cook's distance in linear regression theory (Cook and Weisberg 1982), but it uses the one-step estimate:

$$\overline{C}_i = r_{Pi}^2 h_i/(1-h_i)$$

The DIFDEV and DIFCHISQ statistics are diagnostics for detecting ill-fitted observations—observations that contribute heavily to the disagreement between the data and the predicted values of the fitted model. DIFDEV is the change in the deviance that results from deleting an individual observation, and DIFCHISQ is the change in the Pearson chi-square statistic that results from the same deletion. By using the one-step estimate, DIFDEV and DIFCHISQ for the ith observation are computed as follows:

$$\begin{aligned} \text{DIFDEV}_i &= r_{Di}^2 + \overline{C}_i \\ \text{DIFCHISQ}_i &= \overline{C}_i/h_i \end{aligned}$$

Diagnostics for Models Fit by Generalized Estimating Equations (GEEs)

The diagnostic statistics in this section were developed by Preisser and Qaqish (1996). See the section "Generalized Estimating Equations" on page 761 for further information and notation for generalized estimating equations (GEEs). The following additional notation is used in this section.

Partition the design matrix $\mathbf{X}$ and response vector $\mathbf{Y}$ by cluster; that is, let $\mathbf{X} = (X_1', \ldots, X_K')'$ and $\mathbf{Y} = (Y_1', \ldots, Y_K')'$, corresponding to the K clusters.

Let n_i be the number of responses for cluster i, and denote the total number of observations as $N = \sum_{i=1}^{K} n_i$. Denote the $n_i \times n_i$ diagonal matrix as A_i, where $V(\mu_{ij})$ is the jth diagonal element. If there is a WEIGHT statement, the diagonal element of A_i is $V(\mu_{ij})/w_{ij}$, where w_{ij} is the specified weight of the jth observation in the ith cluster. Let $\mathbf{B}$ be the $N \times N$ diagonal matrix with $g'(\mu_{ij})$ as diagonal elements, $i = 1, \ldots, K$, $j = 1, \ldots, n_i$. Let $\mathbf{B}_i$ be the $n_i \times n_i$ diagonal matrix that corresponds to cluster i, where $g'(\mu_{ij})$ is the jth diagonal element.

Let $\mathbf{W}$ be the $N \times N$ block diagonal weight matrix whose ith block, corresponding to the ith cluster, is the $n_i \times n_i$ matrix

$$\mathbf{W}_{ei} = \mathbf{B}_i^{-1}\mathbf{A}_i^{-\frac{1}{2}}\mathbf{R}_i^{-1}(\hat{\boldsymbol{\alpha}})\mathbf{A}_i^{-\frac{1}{2}}\mathbf{B}_i^{-1}$$

where $\mathbf{R}_i(\hat{\boldsymbol{\alpha}})$ is the working correlation matrix for cluster i that is computed using the estimated correlation parameters $\hat{\boldsymbol{\alpha}}$.

Let

$$\mathbf{Q}_i = \mathbf{X}_i(\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}\mathbf{X}_i'$$

where $\mathbf{X}_i$ is the $n_i \times p$ design matrix that corresponds to cluster i.

The cluster leverage statistic represents the leverage of cluster i and is contained in the matrix $\mathbf{H}_i = \mathbf{Q}_i\mathbf{W}_{ei}$. The leverage of cluster i is summarized by the trace of $\mathbf{H}_i$:

$$ch_i = \text{tr}(\mathbf{H}_i)$$

The leverage h_t of the tth observation in the ith cluster is the tth diagonal element of $\mathbf{H}_i$.

Joint Tests and Type 3 Tests

Linear hypotheses for $\boldsymbol{\beta}$ are expressed in matrix form as

$$H_0: \mathbf{L}\boldsymbol{\beta} = \mathbf{c}$$

where $\mathbf{L}$ is a matrix of coefficients for the linear hypotheses and $\mathbf{c}$ is a vector of constants. The vector of regression coefficients $\boldsymbol{\beta}$ includes slope parameters and intercept parameters. The Wald chi-square statistic for testing H_0 is computed as

$$\chi^2_W = (\mathbf{L}\widehat{\boldsymbol{\beta}} - \mathbf{c})'[\mathbf{L}\widehat{\mathbf{V}}(\widehat{\boldsymbol{\beta}})\mathbf{L}']^{-1}(\mathbf{L}\widehat{\boldsymbol{\beta}} - \mathbf{c})$$

where $\widehat{\mathbf{V}}(\widehat{\boldsymbol{\beta}})$ is the estimated covariance matrix. Under H_0, χ^2_W has an asymptotic chi-square distribution with r degrees of freedom, where r is the rank of $\mathbf{L}$.

For models that use less-than-full-rank parameterization (as specified by the PARAM=GLM option in the CLASS statement), a Type 3 test of an effect of interest (main effect or interaction) is a test of the Type III estimable functions that are defined for that effect. When the model contains no missing cells, the Type 3 test of a main effect is equivalent to testing the hypothesis of equal marginal means. For more information about Type III estimable functions, see the chapter "The GLM Procedure" and the section "The Four Types of Estimable Functions" in *SAS/STAT User's Guide*. Also see Littell, Freund, and Spector (1991).

For models that use full-rank parameterization, all parameters are estimable when there are no missing cells, so it is unnecessary to define estimable functions. The standard test of an effect of interest in this case is the joint test that the values of the parameters associated with that effect are 0. For a model that uses effects parameterization (as specified by the PARAM=EFFECT option in the CLASS statement), the joint test for a main effect is equivalent to testing the equality of marginal means. For a model that uses reference parameterization (as specified by the PARAM=REF option in the CLASS statement), the joint test is equivalent to testing the equality of cell means at the reference level of the other model effects. For more information about the coding scheme and the associated interpretation of results, see Muller and Fetterman (2002, Chapter 14).

If there is no interaction term, the Type 3 test of an effect for a model that uses GLM parameterization is the same as the joint test of the effect for the model that uses full-rank parameterization. In this situation, the joint test is also called the Type 3 test. For a model that contains an interaction term and no missing cells, the Type 3 test of a component main effect under GLM parameterization is the same as the joint test of the component main effect under effect parameterization. Both test the equality of cell means. But this Type 3 test differs from the joint test under reference parameterization, which tests the equality of cell means at the reference level of the other component main effect. If some cells are missing, you can obtain meaningful tests only by testing a Type III estimation function, so in this case you should use GLM parameterization.

The results of a Type 3 test or a joint test do not depend on the order in which you specify the terms in the MODEL statement.

A Type 3 test is not available for models that you fit by using the GEE method.

Multithreading

The GENSELECT procedure allocates data to different threads and calculates the likelihood function, gradient, and Hessian by accumulating the values from all threads. For more information about how PROC GENSELECT uses threads, see the section "Multithreading" on page 102 in Chapter 2, "Shared Concepts."

Optimization Algorithms

Several optimization techniques are available in PROC GENSELECT. You can choose a particular optimizer by using the TECHNIQUE= option in the PROC GENSELECT statement. Table 14.17 summarizes the optimization techniques available in PROC GENSELECT.

Table 14.17 Optimization Techniques

TECHNIQUE=	Algorithm
TRUREG	Trust region method
NEWRAP	Newton-Raphson method with line search
NRRIDG	Newton-Raphson method with ridging
QUANEW	Quasi-Newton methods
DBLDOG	Double-dogleg method
CONGRA	Conjugate gradient methods
LBFGS	Limited-memory BFGS quasi-Newton method
NMSIMP	Nelder-Mead simplex method

There is no algorithm for optimizing general nonlinear functions that always finds the global optimum for a general nonlinear optimization problem in a reasonable amount of time. Because no single optimization technique is always superior to others, PROC GENSELECT provides a variety of optimization techniques that work well in various circumstances. However, you can devise problems for which none of the techniques in PROC GENSELECT can find the correct solution. Moreover, nonlinear optimization can be computationally expensive in terms of time and memory, so you must be careful when matching an algorithm to a problem. The section "Choosing an Optimization Algorithm" on page 103 in Chapter 2, "Shared Concepts," is helpful in choosing a suitable optimization algorithm.

In a GEE analysis, these optimization options control the first stage of estimation that produces the initial values for the GEE fit.

Displayed Output

The following sections describe the output that PROC GENSELECT produces. The output is organized into various tables, which are discussed in their order of appearance.

Store Information

The "Store Information" table displays basic information about the stored model: the item store name, the names of the library and data table used to create the model, the names of the action and procedure that created the item store, and the date when the model was stored. This table is not displayed when you fit a model.

Store Text

The "Store Text" table displays notes that are stored along with the stored model. This table is not displayed when you fit a model.

Model Information

The "Model Information" table displays basic information about the model, such as the response variable, the frequency variable, the link function, and the model category that the GENSELECT procedure selected based on your input and options. The "Model Information" table also displays the distribution of the data that is assumed by the GENSELECT procedure. For information about how the procedure determines the response distribution, see the section "Response Distributions" on page 748. If you specify the CODE statement, then the table displays the name of the variable in the scoring code that contains the classifications. If you do not also specify the PCATALL option, then the table also displays the name of the variable in the scoring code that contains the predicted probabilities.

Number of Observations

The "Number of Observations" table displays the number of observations read from the input data table and the number of observations used in the analysis. If a FREQ statement is present, the table displays the sum of the frequencies that are read and used. If the events/trials syntax is used, the number of events and trials is also displayed. If you specify a PARTITION statement, the table displays the values for each role.

Response Profile

The "Response Profile" table displays the ordered value from which the GENSELECT procedure determines the probability being modeled as an event in a binary model and displays the ordering of categories in a multinomial model. For each response category level, the frequency used in the analysis is reported. You can affect the ordering of the response values by specifying *response-options* in the MODEL statement. For a binary model, the note that follows the "Response Profile" table indicates which outcome is modeled as the event. For an ordinal multinomial model, the note that follows the "Response Profile" table indicates how the ordered response levels are accumulated; for a nominal multinomial model, the note indicates the reference response level. If you specify a PARTITION statement, the table displays the values for each role.

Class Level Information

The "Class Level Information" table lists the levels of every variable specified in the CLASS statement. You should check this information to make sure that the data are correct. You can adjust the order of the CLASS variable levels by specifying the ORDER= option in the CLASS statement. You can suppress the "Class Level Information" table completely or partially by specifying the NOCLPRINT= option in the PROC GENSELECT statement.

If the classification variables use a nonsingular parameterization, the "Class Level Information" table also displays the reference value for each variable.

Selection Information

When you specify the SELECTION statement, the GENSELECT procedure by default produces a series of tables that display information about the model selection. The "Selection Information" table informs you about the model selection method, selection and stop criteria, and other parameters that govern the selection. You can suppress this table by specifying DETAILS=NONE in the SELECTION statement.

Iteration History

When you specify the ITHIST option in the PROC GENSELECT statement, the "Iteration History" table displays, for each iteration of the optimization, the number of function evaluations (including gradient and Hessian evaluations), the value of the objective function, the change in the objective function from the previous iteration, and the absolute value of the largest (projected) gradient element. The objective function that is used in the optimization in the GENSELECT procedure is normalized by default to enable comparisons across data tables that have different sampling intensity. You can control normalization by specifying the NORMALIZE= option in the PROC GENSELECT statement.

Convergence Status

The convergence status table is a small ODS table in the default output. In the listing, it appears as a message that indicates whether the optimization succeeded and which convergence criterion was met. If the optimization fails, the message indicates the reason for the failure. If you save the convergence status table to an output data set, a numeric Status variable is added that enables you to assess convergence programmatically. The values of the Status variable indicate the following:

0 Convergence was achieved, or an optimization was not performed (because TECHNIQUE=NONE is specified).

1 The objective function could not be improved.

2 Convergence was not achieved because of a user interruption or because a limit was exceeded, such as the maximum number of iterations or the maximum number of function evaluations. To modify these limits, see the MAXITER=, MAXFUNC=, and MAXTIME= options in the PROC GENSELECT statement.

3 Optimization failed to converge because function or derivative evaluations failed at the starting values or during the iterations or because a feasible point that satisfies the parameter constraints could not be found in the parameter space.

Entry and Removal Candidates

When you specify DETAILS=ALL or DETAILS=STEPS in the SELECTION statement, the GENSELECT procedure produces the "Entry Candidates" or "Removal Candidates" table, which displays the effect names and values of the criterion used to select entering or departing effects at each step of the selection process. For each step, the effects are displayed in sorted order from best to worst of the selection criterion.

Selection Summary

When you specify the SELECTION statement, the GENSELECT procedure produces the "Selection Summary" table, which displays information about which effects were added to or removed from the model in the various steps of the model selection process. The statistic that led to the entry or removal decision is also displayed. You can request further details about the model selection steps by specifying DETAILS=STEPS or DETAILS=ALL in the SELECTION statement. You can suppress the display of the "Selection Summary" table by specifying DETAILS=NONE in the SELECTION statement.

If you specify the LASSO selection method, then this table displays information about which effect was added to or removed from the model, the number of effects in the model, the lambda value, and the information criteria.

Stop Reason

When you specify the SELECTION statement, the GENSELECT procedure produces a simple table that tells you why model selection stopped.

Selection Reason

When you specify the SELECTION statement, the GENSELECT procedure produces a simple table that tells you why the final model was selected.

Selected Effects

When you specify the SELECTION statement, the GENSELECT procedure produces a simple table that tells you which effects were included in the final model.

Dimensions

The "Dimensions" table displays size measures that are derived from the model and the environment. It displays the number of columns in the current design matrix, the number of effects in the current design, the largest number of design columns associated with an effect, the rank of the matrix, and the number of parameters in the current model, including any dispersion parameters.

Fit Statistics

The "Fit Statistics" table displays a variety of likelihood-based measures of fit. All statistics are presented in "smaller is better" form. The values that the "Fit Statistics" table displays are not based on a normalized log-likelihood function. If you specify a PARTITION statement or the PARTFIT option, the table displays the values for each role along with statistics for comparing the training, validation, and testing results.

For more information about the statistics displayed in this table, see the section "Model Fit and Assessment Statistics" on page 767.

This table is not produced in a GEE analysis.

Parameter Estimates

The parameter estimates, their estimated (asymptotic) standard errors, and p-values for the hypothesis that the parameter is 0 are presented in the "Parameter Estimates" table. If you request confidence intervals by specifying the CLB option in the MODEL statement, confidence limits are produced for the estimates.

This table is not produced by default in a GEE analysis. You can produce this table by specifying the PRINTMLE option in the REPEATED statement. In this case, the table is labeled "Analysis of Initial Parameter Estimates."

Parameter Estimates Covariance Matrix

When you specify the COVB option in the PROC GENSELECT statement, the GENSELECT procedure displays the covariance matrix of the parameter estimates. The covariance matrix is computed as the inverse of the negative of the matrix of second derivatives of the log-likelihood function with respect to the model parameters (the Hessian matrix), evaluated at the parameter estimates.

This table is not produced in a GEE analysis.

Parameter Estimates Correlation Matrix

When you specify the CORRB option in the PROC GENSELECT statement, the GENSELECT procedure displays the correlation matrix of the parameter estimates.

This table is not produced in a GEE analysis.

GEE Model Information

The "GEE Model Information" table displays basic information about the GEE model, including the subject effects, any within-subject effects, the working correlation type, and details about the size of the working correlation matrix.

GEE Parameter Estimates Covariance Matrix Using Model-Based Covariance

The "GEE Parameter Estimates Covariance Matrix Using Model-Based Covariance" table displays the parameter estimates covariance matrix from the GEE analysis that uses the model-based covariance estimate. For more information about the covariance estimates that are used for GEE analysis, see the section "Parameter Estimate Covariances" on page 765.

This table is not displayed unless you specify the MCOVB or COVB option in the REPEATED statement.

GEE Parameter Estimates Correlation Matrix Using Model-Based Covariance

The "GEE Parameter Estimates Correlation Matrix Using Model-Based Covariance" table displays the parameter estimates correlation matrix from the GEE analysis that uses the model-based covariance estimate. For more information about the covariance estimates that are used for GEE analysis, see the section "Parameter Estimate Covariances" on page 765.

This table is not displayed unless you specify the MCORRB or CORRB option in the REPEATED statement.

GEE Parameter Estimates Covariance Matrix Using Empirical Covariance

The "GEE Parameter Estimates Covariance Matrix Using Empirical Covariance" table displays the parameter estimates covariance matrix from the GEE analysis that uses the empirical covariance estimate. For more information about the covariance estimates that are used for GEE analysis, see the section "Parameter Estimate Covariances" on page 765.

This table is not displayed unless you specify the ECOVB or COVB option in the REPEATED statement.

GEE Parameter Estimates Correlation Matrix Using Empirical Covariance

The "GEE Parameter Estimates Correlation Matrix Using Empirical Covariance" table displays the parameter estimates correlation matrix from the GEE analysis that uses the empirical covariance estimate. For more information about the covariance estimates that are used for GEE analysis, see the section "Parameter Estimate Covariances" on page 765.

This table is not displayed unless you specify the ECORRB or CORRB option in the REPEATED statement.

GEE Working Correlation Matrix

The "GEE Working Correlation Matrix" table displays the correlation matrix of the repeated measures. Its dimension depends on the number of repeated measures and any within-subject effects.

This table is not displayed unless you specify the CORRW option in the REPEATED statement.

GEE Exchangeable Correlation

The "GEE Exchangeable Correlation" table displays the exchangeable correlation parameter. For more information about the working correlation structures, see the section "Working Correlation Matrix" on page 762.

This table is not displayed unless you specify TYPE=EXCH and the CORRW option in the REPEATED statement.

GEE Fit Criteria

The "GEE Fit Criteria" table displays the information criteria for the GEE model. For more information about the fit statistics that are used with GEE, see the section "Quasi-likelihood Information Criterion" on page 765.

GEE Parameter Estimates Using Empirical Standard Errors

The "GEE Parameter Estimates Using Empirical Standard Errors" table presents the parameter estimates from the GEE analysis. The table displays the standard errors that are derived from the empirical covariance matrix. The table includes the parameter estimates, their estimated standard errors, and p-values for the hypothesis that the parameter is 0. If you request confidence intervals by specifying the CLB option in the MODEL statement, confidence limits are produced for the estimates.

For more information about the covariance estimates that are used for GEE analysis, see the section "Parameter Estimate Covariances" on page 765.

GEE Parameter Estimates Using Model-Based Standard Errors

The "GEE Parameter Estimates Using Model-Based Standard Errors" table presents the parameter estimates from the GEE analysis. The table displays the standard errors that are derived from the model-based covariance matrix. The table includes the parameter estimates, their estimated standard errors, and p-values for the hypothesis that the parameter is 0. If you request confidence intervals by specifying the CLB option in the MODEL statement, confidence limits are produced for the estimates.

This table is not produced unless you specify the MODELSE option in the REPEATED statement. For more information about the covariance estimates that are used for GEE analysis, see the section "Parameter Estimate Covariances" on page 765.

Model Analysis of Variance (Type III)

When you specify the TYPE3 option in the MODEL statement, the procedure produces the "Model Analysis of Variance (Type III)" table. This table displays tests that all parameters for a particular effect are 0. For more information, see the section "Joint Tests and Type 3 Tests" on page 772.

This table is not produced in a GEE analysis.

Postfit Information

The "Postfit Information" table displays the name of the data table that the stored model uses to create the results that are displayed in the Restore section of the output, the number of observations that are read from this data table, and the number of observations that are used in the analyses. If you specify a FREQ statement, the table displays the sum of the frequencies that are read and used. If you specify a WEIGHT statement, the table displays the sum of $f_i w_i$ that are read and used. If the stored model is created using a PARTITION statement, the table displays the values for each role. This table is not displayed when you fit a model. This table is followed by "Response Profile" and "Fit Statistics" tables that are computed by using the input data table.

Score Code Variables for Predicted Values

When you specify the PCATALL option in the CODE statement and you specify the DISTRIBUTION=BINARY option in the MODEL statement, the GENSELECT procedure produces the "Score Code Variables for Predicted Probability" table. For each level of the response variable, this table displays the corresponding variable in the scoring code that contains the predicted probability.

Timing

The "Timing" table displays the amount of time (in seconds) that PROC GENSELECT required to perform different tasks in the analysis.

OutputCasTables Table

The OutputCasTables table is a special table that has information about each CAS table that is created during a CAS action execution. The information for each CAS table consists of the CAS table name, the caslib in which the table resides, and the number of columns and rows in the CAS table. Because this table is not a typical ODS table that contains analytical results, you cannot include it in the *table-spec-list* in the DISPLAYOUT statement.

ODS Table Names

Each table that the GENSELECT procedure creates has a name associated with it. You must use this name to refer to the table when you use the DISPLAY statement, the DISPLAYOUT statement, or ODS statements. These names are listed in Table 14.18.

The EFFECT and LSMEANS statements also create tables that are not listed in Table 14.18. For information about these tables, see the corresponding sections of Chapter 2, "Shared Concepts."

Table 14.18 ODS Tables Produced by PROC GENSELECT

Table Name	Description	Statement	Options
ClassInfo	Level information from the CLASS statement	CLASS	Default
ConvergenceStatus	Status of optimization at conclusion of optimization	PROC GENSELECT	Default
CorrB	Correlation matrix of parameter estimates	PROC GENSELECT	CORRB
CovB	Covariance matrix of parameter estimates	PROC GENSELECT	COVB
Dimensions	Model dimensions	PROC GENSELECT	Default
EntryCandidates	Details about candidates for entry into the model	SELECTION	METHOD=FORWARD DETAILS=STEP
FitStatistics	Fit statistics	PROC GENSELECT	Default
GEEEmpPEst	GEE parameter estimates using empirical standard errors	REPEATED	Default
GEEExchCorr	Exchangeable correlation parameter	REPEATED	TYPE=EXCH
GEEFitCriteria	Measures of fit for GEE models	REPEATED	Default
GEEModInfo	Information about GEE model	REPEATED	Default
GEEModPEst	GEE parameter estimates using model-based standard errors	REPEATED	MODELSE
GEENCorr	Correlation matrix of GEE parameter estimates using model-based covariance	REPEATED	CORRB or MCORRB
GEENCov	Covariance matrix of GEE parameter estimates using model-based covariance	REPEATED	COVB or MCOVB
GEERCorr	Correlation matrix of GEE parameter estimates using empirical covariance	REPEATED	CORRB or ECORRB
GEERCov	Covariance matrix of GEE parameter estimates using empirical covariance	REPEATED	COVB or ECOVB
GEEWCorr	Working correlation matrix of repeated measures	REPEATED	CORRW

Table 14.18 *continued*

Table Name	Description	Statement	Options
IterHistory	Iteration history	PROC GENSELECT	ITHIST
ModelInfo	Information about the modeling environment	PROC GENSELECT	Default
ModelAnova	Model analysis of variance (Type III)	MODEL	TYPE3
NObs	Number of observations read and used, and number of events and trials, if applicable	PROC GENSELECT	Default
OutputCasTables	See the section "OutputCasTables Table" on page 779	OUTPUT DISPLAYOUT	OUT=
ParameterEstimates	Solutions for the parameter estimates associated with effects in MODEL statements	PROC GENSELECT REPEATED	Default PRINTMLE
PostfitInfo	Input data table name and number of observations read and used for the restored model	PROC GENSELECT	RESTORE= and DATA=
PredProbName	Displays the predicted probability variable in the scoring code associated with each response level	CODE	PCATALL
RemovalCandidates	Details about candidates for removal from the model	SELECTION	METHOD=BACKWARD DETAILS=STEP
ResponseProfile	Response categories for binary and multinomial data	Default	
SelectedEffects	List of effects selected for the model	SELECTION	Default
SelectionInfo	Information about the settings for model selection	SELECTION	Default
SelectionReason	Reason why the particular model was selected	SELECTION	Default
SelectionSummary	Summary information about model selection steps	SELECTION	Default
StopReason	Reason for termination of model selection	SELECTION	Default
StoreInfo	Information about the restored model	PROC GENSELECT	RESTORE=
StoreText	Notes stored along with the model	PROC GENSELECT	RESTORE=
Timing	Absolute and relative times for tasks performed by the procedure	PROC GENSELECT	Default

ODS Graphics

Statistical procedures use ODS Graphics to create graphs as part of their output. ODS Graphics is described in detail in the "Statistical Graphics Using ODS" chapter in *SAS/STAT User's Guide*.

Before you create graphs, ODS Graphics must be enabled (for example, by using the ODS GRAPHICS ON statement). For more information about enabling and disabling ODS Graphics, see the section "Enabling and Disabling ODS Graphics" in that chapter.

The overall appearance of graphs is controlled by ODS styles. Styles and other aspects of using ODS Graphics are discussed in the section "A Primer on ODS Statistical Graphics" in that chapter.

When ODS Graphics is enabled, the SELECTION statement can produce plots to help evaluate the selection process. For information about these plots, see the section "Model Selection Plots" on page 93 in Chapter 2, "Shared Concepts."

PROC GENSELECT assigns a name to each graph that it creates using ODS. You can use these names to reference the graphs when using ODS. The names are listed in Table 14.19.

Table 14.19 Graphs Produced by PROC GENSELECT

ODS Graph Name	Plot Description	PLOTS Option
CoefficientPanel	Coefficients and CHOOSE= criterion by step	COEFFICIENTS
ChooseCriterionPlot	CHOOSE= criterion by step	COEFFICIENTS(UNPACK)
CoefficientPlot	Coefficients by step	COEFFICIENTS(UNPACK)
CriterionPanel	Fit criteria by step	CRITERIA
AICCPlot	Corrected Akaike's information criterion by step	CRITERIA(UNPACK)
AICPlot	Akaike's information criterion by step	CRITERIA(UNPACK)
SBCPlot	Schwarz Bayesian information criterion by step	CRITERIA(UNPACK)
TEST_ASEPlot	Average square error on testing data by step	CRITERIA(UNPACK)
TRAIN_ASEPlot	Average square error on training data by step	CRITERIA(UNPACK)
VAL_ASEPlot	Average square error on validation data by step	CRITERIA(UNPACK)
FitByRolePlot	Progression of average square error by role	FITBYROLE

Examples: GENSELECT Procedure

The examples in this chapter are available in the GitHub repository located at https://github.com/sascommunities/doc-supplement-casstat.

Example 14.1: Model Selection

(View the complete code for this example (genex1.sas) in the example repository.)

The following statements examine the same data set that is used in the section "Getting Started: GENSELECT Procedure" on page 708, but they request model selection via the forward selection technique. Model effects are added in the order of their significance until no more effects significantly improve the current model. The DETAILS=ALL option in the SELECTION statement requests that all tables that are related to model selection be produced.

The data set getStarted is shown in the section "Getting Started: GENSELECT Procedure" on page 708. It contains 100 observations on a count response variable (Y), a continuous variable (Total) to be used in Example 14.2, and five categorical variables (C1–C5), each of which has four numerical levels.

A log-linked Poisson regression model is specified by using classification effects for variables C1–C5. The following statements request model selection by the forward selection method:

```
ods graphics on;

proc genselect data=mylib.getStarted;
   class C1-C5;
   model Y = C1-C5 / Distribution=Poisson;
   selection method=forward details=all plots=all;
run;

ods graphics off;
```

The model selection tables are shown in Output 14.1.1 through Output 14.1.3. Selection graphics produced by the PLOTS= option are displayed in Output 14.1.6 and Output 14.1.7.

The "Selection Information" table in Output 14.1.1 summarizes the settings for the model selection. The effect that produces the smallest approximate SBC is added to the model. The forward selection stops three steps after the smallest SBC is obtained, or when all effects have been added to the model.

Output 14.1.1 Selection Information

The GENSELECT Procedure

Selection Information	
Selection Method	Forward
Select Criterion	SBC
Stop Criterion	SBC
Effect Hierarchy Enforced	None
Stop Horizon	3

The "Selection Summary" table in Output 14.1.2 shows the effects that were added to the model and the approximate SBC. Step 0 refers to the null model that contains only an intercept. In the next step, effect C2 resulted in the smallest approximate SBC among the candidate effects. In step 2, the smallest SBC when an effect is added to a model that contains the intercept and C2 was achieved by adding C5 to the model. Similarly, in steps 3, 4, and 5, effects were added in order of the minimum SBC among remaining candidates. Finally, the model in step 2 was chosen as the best because it had the smallest SBC among all the candidate models.

Output 14.1.2 Selection Summary Information

Selection Summary

Step	Effect Entered	Number Effects In	SBC
0	Intercept	1	354.7980
1	C2	2	342.8795
2	C5	3	333.8182*
3	C1	4	341.7993
4	C4	5	351.8375
5	C3	6	363.8674

* Optimal Value Of Criterion

Selection stopped at a local minimum of the STOP criterion.

The model at step 2 is selected.

Selected Effects: Intercept C2 C5

For each step of the selection process, the DETAILS=ALL option displays the candidate effects for entering the model along with their SELECT= criterion. Output 14.1.3 displays this table for the first step; the other steps are not shown here.

Output 14.1.3 Step1 Entry Candidates

Entry Candidates

Rank	Effect	SBC
1	C2	342.8795
2	C5	347.5376
3	C1	355.7067
4	C4	361.7379
5	C3	365.7339

The DETAILS=ALL option also displays the dimensions, fit statistics, and parameter estimates at each step of the selection process; these details are not shown here.

Output 14.1.4 displays information about the selected model. Notice that the –2 log-likelihood value in the "Fit Statistics" table is larger than the value for the full model in Figure 14.6. This is expected because the selected model contains only a subset of the parameters. Because the selected model is more parsimonious than the full model, the information criteria AIC, AICC, and BIC are smaller than in the full model, indicating a better fit.

Output 14.1.4 Fit Statistics

Fit Statistics	
-2 Log Likelihood	303.58036
AIC (smaller is better)	317.58036
AICC (smaller is better)	318.79775
SBC (smaller is better)	335.81655

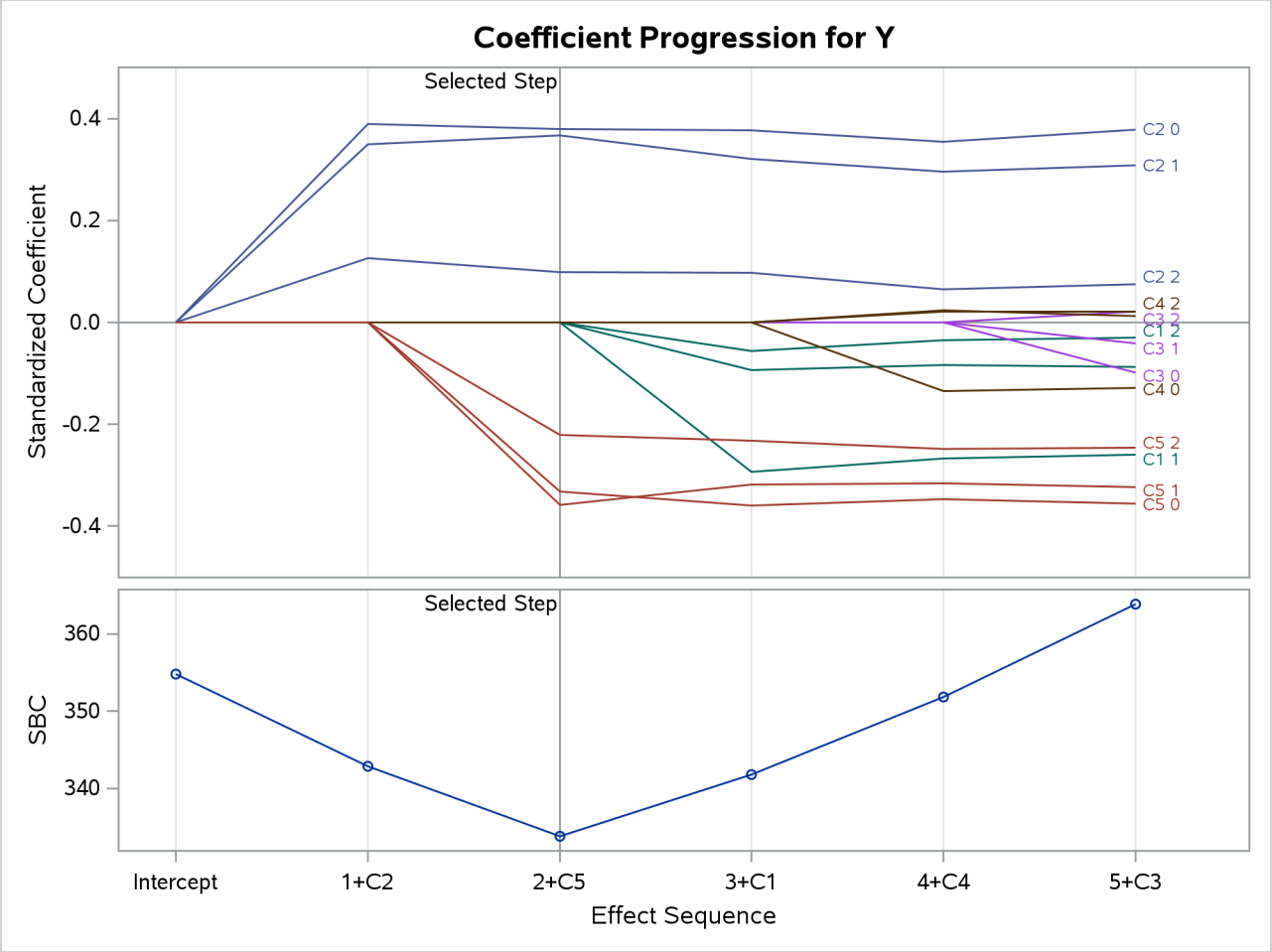
The parameter estimates of the selected model are shown in Output 14.1.5. Notice that the effects are listed in the "Parameter Estimates" table in the order in which they were specified in the MODEL statement and not the order in which they were added to the model.

Output 14.1.5 Parameter Estimates

Parameter Estimates					
Parameter	**DF**	**Estimate**	**Standard Error**	**Chi-Square**	**Pr > ChiSq**
Intercept	1	0.528024	0.215231	6.0186	0.0142
C2 0	1	0.962636	0.236921	16.5088	<.0001
C2 1	1	0.845132	0.233634	13.0851	0.0003
C2 2	1	0.213247	0.248896	0.7341	0.3916
C2 3	0	0	.	.	.
C5 0	1	-0.763491	0.207082	13.5933	0.0002
C5 1	1	-0.785915	0.201271	15.2471	<.0001
C5 2	1	-0.541212	0.208661	6.7275	0.0095
C5 3	0	0	.	.	.

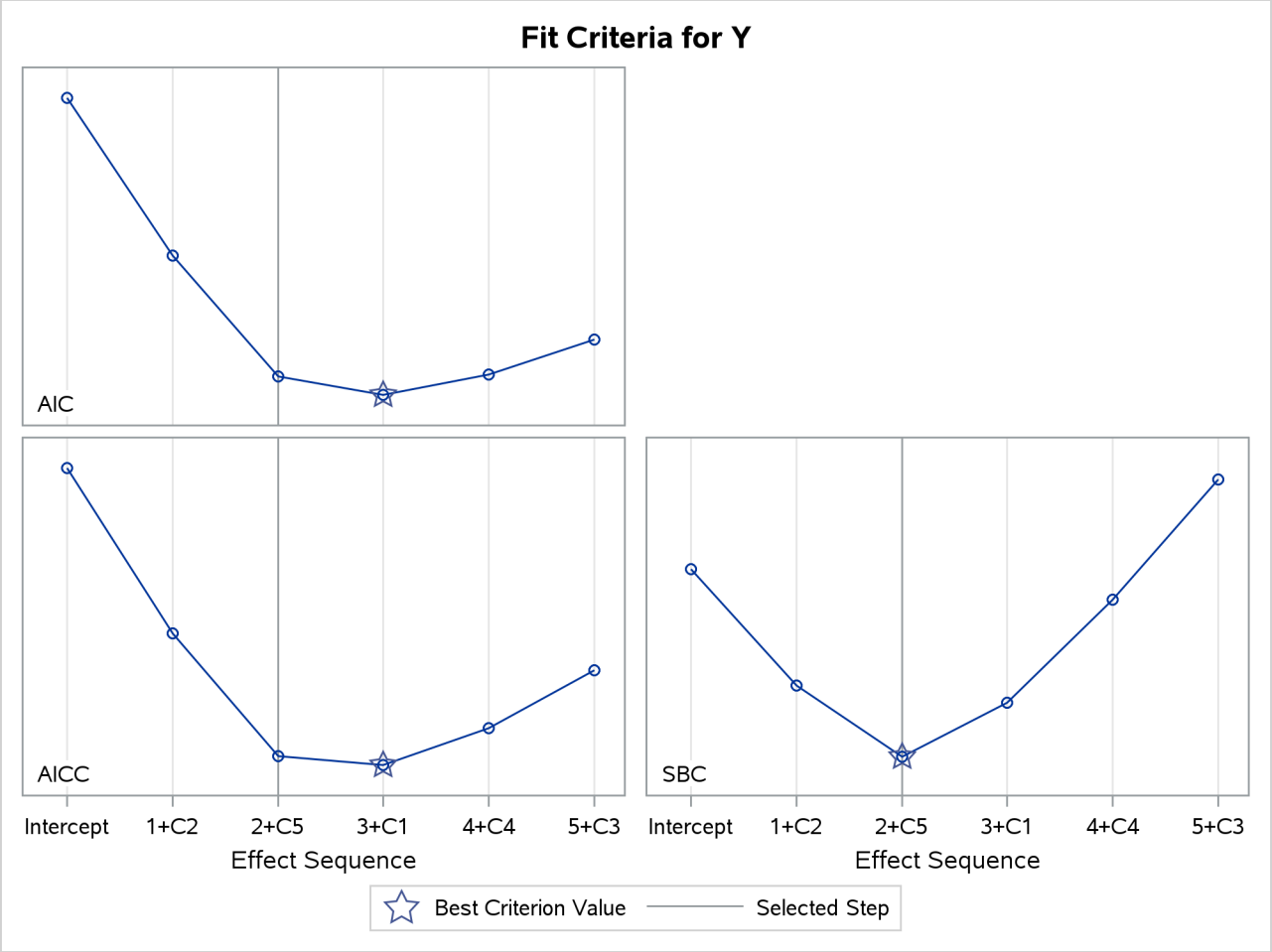
The coefficient panel in Figure 14.1.6 enables you to visualize the selection process. In this plot, standardized coefficients of all the effects that are selected at some step of the stepwise method are plotted as a function of the step number. This enables you to assess the relative importance of the effects that are selected at any step of the selection process and to know when effects entered the model. The lower plot in the panel shows how the criterion that is used to choose the selected model changes as effects enter or leave the model.

Output 14.1.6 Coefficient Progression



The criterion panel in Figure 14.1.7 provides a graphical view of the progression of the fit criteria as the selection process evolves.

Output 14.1.7 Criterion Panel



Example 14.2: Gamma Model

(View the complete code for this example (genex2.sas) in the example repository.)

The following statements examine the data set getStarted, which is used in the section "Getting Started: GENSELECT Procedure" on page 708, but they request that a log-linked gamma model be fit by using the continuous variable Total as the response instead of the count variable Y. The following statements fit a log-linked gamma model to these data by using classification effects for the variables C1–C5. The CLB MODEL statement option requests that 95% confidence limits be computed and displayed along with the parameter estimates. The CODE statement requests that a text file named "Scoring Parameters.txt" be created. This file contains a SAS program that has information from the model that allows scoring of a new data set based on the parameter estimates from the current model.

```
proc genselect data=mylib.getStarted;
   class C1-C5;
   model Total = C1-C5 / Distribution=Gamma Link=Log CLB;
   code File='ScoringParameters.txt';
run;
```

The "Parameter Estimates" table in Output 14.2.1 shows the resulting regression model parameter estimates and the estimated gamma dispersion parameter.

Output 14.2.1 Parameter Estimates

The GENSELECT Procedure

Parameter Estimates

Parameter	DF	Estimate	Standard Error	Chi-Square	Pr > ChiSq	95% Confidence Limits	
Intercept	1	4.028096	0.454883	78.4153	<.0001	3.13654	4.91965
C1 0	1	-0.064442	0.256719	0.0630	0.8018	-0.56760	0.43872
C1 1	1	-1.308470	0.318002	16.9305	<.0001	-1.93174	-0.68520
C1 2	1	0.295802	0.257834	1.3162	0.2513	-0.20954	0.80115
C1 3	0	0	.	.	.	.	.
C2 0	1	1.154556	0.295110	15.3060	<.0001	0.57615	1.73296
C2 1	1	0.824472	0.295273	7.7966	0.0052	0.24575	1.40320
C2 2	1	-0.287943	0.288421	0.9967	0.3181	-0.85324	0.27735
C2 3	0	0	.	.	.	.	.
C3 0	1	-0.008546	0.283509	0.0009	0.9760	-0.56421	0.54712
C3 1	1	-0.319783	0.276053	1.3419	0.2467	-0.86084	0.22127
C3 2	1	-0.071452	0.296418	0.0581	0.8095	-0.65242	0.50952
C3 3	0	0	.	.	.	.	.
C4 0	1	-0.143018	0.288101	0.2464	0.6196	-0.70769	0.42165
C4 1	1	-0.219348	0.290441	0.5704	0.4501	-0.78860	0.34991
C4 2	1	0.091763	0.278635	0.1085	0.7419	-0.45435	0.63788
C4 3	0	0	.	.	.	.	.
C5 0	1	-1.227558	0.267605	21.0425	<.0001	-1.75205	-0.70306
C5 1	1	-0.560699	0.252238	4.9413	0.0262	-1.05508	-0.06632
C5 2	1	-0.252965	0.259908	0.9473	0.3304	-0.76238	0.25645
C5 3	0	0	.	.	.	.	.
Dispersion	1	1.672305	0.238205			1.26494	2.21086

Now suppose you want to compute predicted values for some different data. If $\mathbf{x}$ is a vector of explanatory variables that might not be in the original data and $\hat{\boldsymbol{\beta}}$ is the vector of estimated regression parameters from the model, then $\mu = g^{-1}(\mathbf{x}'\hat{\boldsymbol{\beta}})$ is the predicted value of the mean, where g is the log link function in this case.

The following data contain new values of the regression variables C1–C5, from which you can compute predicted values based on information in the SAS program that is created by the CODE statement. This is called *scoring* the new data set.

```
data ScoringData;
   input C1-C5;
   datalines;
3 3 1 0 2
1 1 2 2 0
3 2 2 2 0
1 1 2 3 2
1 1 2 3 3
3 1 1 0 1
```

```
0 2 1 0 0
2 1 3 1 3
3 2 3 2 0
3 0 2 0 1
;
```

The following SAS DATA step creates the new data set Scores, which contains a variable P_Total that represents the predicted values of Total, along with the variables C1–C5. The resulting data are shown in Output 14.2.2.

```
data Scores;
   set ScoringData;
   %inc 'ScoringParameters.txt';
run;
proc print data=Scores;
run;
```

Output 14.2.2 Predicted Values for Scoring Data

Obs	C1	C2	C3	C4	C5	P_Total
1	3	3	1	0	2	27.449
2	1	1	2	2	0	10.349
3	3	2	2	2	0	12.590
4	1	1	2	3	2	25.020
5	1	1	2	3	3	32.222
6	3	1	1	0	1	46.020
7	0	2	1	0	0	7.282
8	2	1	3	1	3	138.244
9	3	2	3	2	0	13.523
10	3	0	2	0	1	82.063

Example 14.3: Binary Model with Repeated Measures

(View the complete code for this example (genex3.sas) in the example repository.)

This example uses data from the "Six Cities" study of the effects of air pollution on health (Ware et al. 1984). The analysis concerns the 16 selected cases in Lipsitz et al. (1994). The response is binary and indicates the wheezing status of 16 children. The children were assessed at ages 9, 10, 11, and 12 years. A value of 1 for wheezing status indicates the occurrence of wheezing at that age. The following statements create the data set:

```
data mylib.six;
     input case city$ @@;
   do i=1 to 4;
      input age smoke wheeze @@;
      output;
   end;
   datalines;
 1 portage   9 0 1  10 0 1  11 0 1  12 0 0
 2 kingston  9 1 1  10 2 1  11 2 0  12 2 0
 3 kingston  9 0 1  10 0 0  11 1 0  12 1 0
```

```
 4 portage   9 0 0  10 0 1  11 0 1  12 1 0
 5 kingston  9 0 0  10 1 0  11 1 0  12 1 0
 6 portage   9 0 0  10 1 0  11 1 0  12 1 0
 7 kingston  9 1 0  10 1 0  11 0 0  12 0 0
 8 portage   9 1 0  10 1 0  11 1 0  12 2 0
 9 portage   9 2 1  10 2 0  11 1 0  12 1 0
10 kingston  9 0 0  10 0 0  11 0 0  12 1 0
11 kingston  9 1 1  10 0 0  11 0 1  12 0 1
12 portage   9 1 0  10 0 0  11 0 0  12 0 0
13 kingston  9 1 0  10 0 1  11 1 1  12 1 1
14 portage   9 1 0  10 2 0  11 1 0  12 2 1
15 kingston  9 1 0  10 1 0  11 1 0  12 2 1
16 portage   9 1 1  10 1 1  11 2 0  12 1 0
;
```

The mean response is modeled by using a logistic regression model. The explanatory variables are city of residence (city), age in years (age), and maternal smoking status at each age (smoke). The responses for each child are assumed to be equally correlated, indicating that an exchangeable correlation structure is appropriate. The following statements fit the model by using generalized estimating equations (GEEs) and the chosen correlation structure:

```
proc genselect data=mylib.six;
   class case city;
   model wheeze(event='1') = city age smoke / dist=binary;
   repeated subject=case / type=exch covb corrw printmle;
run;
```

The MODEL statement specifies the mean model for the wheeze response variable as a logistic regression that uses city, age, and smoke as explanatory variables. This is the same specification that you use for an ordinary logistic regression.

The REPEATED statement uses GEE for parameter estimation, specifies the working correlation structure, and controls the output that is displayed. The SUBJECT=CASE option specifies that the case variable identifies the individual subjects in the data set. The case variable must be listed in the CLASS statement. The TYPE=EXCH option specifies that the procedure use the exchangeable working correlation structure. The exchangeable correlation structure does not rely on an order in the repeated measurements, so you do not need to specify a within-subject effect in the WITHIN= option. The COVB option displays the parameter estimates covariance matrix, and the CORRW option displays the final working correlation matrix. The PRINTMLE option displays a table of the initial parameter estimates that are produced by maximum likelihood estimation.

The initial parameter estimates for the GEE fit are computed by an ordinary logistic regression, which assumes no dependence between the repeated measurements.

Figure 14.3.1 displays general information about the GEE model fit.

Output 14.3.1 GEE Model Information

The GENSELECT Procedure

GEE Model Information	
Correlation Structure	Exchangeable
Subject Effect	case (16 levels)
Number of Clusters	16
Correlation Matrix Dimension	4
Maximum Cluster Size	4
Minimum Cluster Size	4

Figure 14.3.2 displays the estimated parameter covariance matrices. The COVB option produces both the model-based and empirical covariance matrices.

Output 14.3.2 GEE Parameter Estimate Covariance Matrices

GEE Parameter Estimates
Model-Based Covariance Matrix

	Intercept	city kingston	age	smoke
Intercept	5.7495	-0.2226	-0.5347	0.01655
city kingston	-0.2226	0.4548	-0.00241	0.01876
age	-0.5347	-0.00241	0.05300	-0.01658
smoke	0.01655	0.01876	-0.01658	0.1910

GEE Parameter Estimates
Empirical Covariance Matrix

	Intercept	city kingston	age	smoke
Intercept	9.3399	-0.8510	-0.8325	-0.1653
city kingston	-0.8510	0.4737	0.05736	0.04023
age	-0.8325	0.05736	0.07778	-0.00236
smoke	-0.1653	0.04023	-0.00236	0.1305

The working correlation matrix that is specified by the CORRW option is displayed in Figure 14.3.3.

Output 14.3.3 GEE Working Correlation Matrix

Working Correlation Matrix				
	Col1	**Col2**	**Col3**	**Col4**
Row1	1.0000	0.1648	0.1648	0.1648
Row2	0.1648	1.0000	0.1648	0.1648
Row3	0.1648	0.1648	1.0000	0.1648
Row4	0.1648	0.1648	0.1648	1.0000

The exchangeable working correlation structure is characterized by a single parameter. Figure 14.3.4 shows the table that contains this information.

Output 14.3.4 GEE Exchangeable Correlation

GEE Exchangeable Correlation	
Correlation	0.164816

Figure 14.3.5 displays the parameter estimates table. This table shows the parameter estimates, standard errors, chi-square values, and *p*-values for the parameter estimates. The table uses the empirical standard error estimates. You can also produce a table that uses the model-based standard error estimates by specifying the MODELSE option in the REPEATED statement.

Output 14.3.5 GEE Parameter Estimates Table

Analysis of GEE Parameter Estimates Using Empirical Standard Error Estimates					
Parameter	**DF**	**Estimate**	**Standard Error**	**Chi-Square**	**Pr > ChiSq**
Intercept	1	1.2751	3.0561	0.1741	0.6765
city kingston	1	0.1223	0.6882	0.0316	0.8589
city portage	0	0	.	.	.
age	1	-0.2036	0.2789	0.5327	0.4655
smoke	1	-0.0935	0.3613	0.0670	0.7957

You can compare the GEE parameter estimates to the parameter estimates from the initial model. The parameter estimates from the initial model are displayed in Figure 14.3.6.

Output 14.3.6 Initial Parameter Estimates Table

Analysis of Initial Parameter Estimates					
Parameter	**DF**	**Estimate**	**Standard Error**	**Chi-Square**	**Pr > ChiSq**
Intercept	1	1.259707	2.610423	0.2329	0.6294
city kingston	1	0.139098	0.552686	0.0633	0.8013
city portage	0	0	.	.	.
age	1	-0.200315	0.250815	0.6379	0.4245
smoke	1	-0.128364	0.410239	0.0979	0.7544

The GEE parameter estimates are not substantially different from the parameter estimates in the initial model.

References

Akaike, H. (1974). "A New Look at the Statistical Model Identification." *IEEE Transactions on Automatic Control* AC-19:716–723.

Albert, A., and Anderson, J. A. (1984). "On the Existence of Maximum Likelihood Estimates in Logistic Regression Models." *Biometrika* 71:1–10.

Brier, G. W. (1950). "Verification of Forecasts Expressed in Terms of Probability." *Monthly Weather Review* 78:1–3.

Burnham, K. P., and Anderson, D. R. (1998). *Model Selection and Inference: A Practical Information-Theoretic Approach*. New York: Springer-Verlag.

Cook, R. D., and Weisberg, S. (1982). *Residuals and Influence in Regression*. New York: Chapman & Hall.

Diggle, P. J., Liang, K.-Y., and Zeger, S. L. (1994). *Analysis of Longitudinal Data*. Oxford: Clarendon Press.

Dunn, P. K., and Smyth, G. K. (2005). "Series Evaluation of Tweedie Exponential Dispersion Model Densities." *Statistics and Computing* 15:267–280.

Dunn, P. K., and Smyth, G. K. (2008). "Series Evaluation of Tweedie Exponential Dispersion Model Densities by Fourier Inversion." *Statistics and Computing* 18:73–86.

Ferrari, S. L. P., and Cribari-Neto, F. (2004). "Beta Regression for Modelling Rates and Proportions." *Journal of Applied Statistics* 31:799–815.

Frees, E. W. (2010). *Regression Modeling with Actuarial and Financial Applications*. Cambridge: Cambridge University Press.

Hardin, J. W., and Hilbe, J. M. (2003). *Generalized Estimating Equations*. Boca Raton, FL: Chapman & Hall/CRC.

Hastie, T. J., Tibshirani, R. J., and Friedman, J. H. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd ed. New York: Springer-Verlag.

Hosmer, D. W., Jr., and Lemeshow, S. (2000). *Applied Logistic Regression*. 2nd ed. New York: John Wiley & Sons.

Hurvich, C. M., and Tsai, C.-L. (1989). "Regression and Time Series Model Selection in Small Samples." *Biometrika* 76:297–307.

Joe, H., and Zhu, R. (2005). "Generalized Poisson Distribution: The Property of Mixture of Poisson and Comparison with Negative Binomial Distribution." *Biometrical Journal* 47:219–229.

Lawless, J. F., and Singhal, K. (1978). "Efficient Screening of Nonnormal Regression Models." *Biometrics* 34:318–327.

Liang, K.-Y., and Zeger, S. L. (1986). "Longitudinal Data Analysis Using Generalized Linear Models." *Biometrika* 73:13–22.

Lipsitz, S. R., Fitzmaurice, G. M., Orav, E. J., and Laird, N. M. (1994). "Performance of Generalized Estimating Equations in Practical Situations." *Biometrics* 50:270–278.

Littell, R. C., Freund, R. J., and Spector, P. C. (1991). *SAS System for Linear Models*. 3rd ed. Cary, NC: SAS Institute Inc.

McCullagh, P., and Nelder, J. A. (1989). *Generalized Linear Models*. 2nd ed. London: Chapman & Hall.

McFadden, D. (1974). "Conditional Logit Analysis of Qualitative Choice Behavior." In *Frontiers in Econometrics*, edited by P. Zarembka, 105–142. New York: Academic Press.

Muller, K. E., and Fetterman, B. A. (2002). *Regression and ANOVA: An Integrated Approach Using SAS Software*. Cary, NC: SAS Institute Inc.

Murphy, A. H. (1973). "A New Vector Partition of the Probability Score." *Journal of Applied Meteorology* 12:595–600.

Pan, W. (2001). "Akaike's Information Criterion in Generalized Estimating Equations." *Biometrics* 57:120–125.

Pregibon, D. (1981). "Logistic Regression Diagnostics." *Annals of Statistics* 9:705–724.

Preisser, J. S., and Qaqish, B. F. (1996). "Deletion Diagnostics for Generalised Estimating Equations." *Biometrika* 83:551–562.

Royall, R. M. (1986). "Model Robust Inference Using Maximum Likelihood Estimators." *International Statistical Review* 54:221–226.

Santner, T. J., and Duffy, D. E. (1986). "A Note on A. Albert and J. A. Anderson's Conditions for the Existence of Maximum Likelihood Estimates in Logistic Regression Models." *Biometrika* 73:755–758.

Schwarz, G. (1978). "Estimating the Dimension of a Model." *Annals of Statistics* 6:461–464.

Ware, J. H., Dockery, S. A., III, Speizer, F. E., and Ferris, B. G., Jr. (1984). "Passive Smoking, Gas Cooking, and Respiratory Health of Children Living in Six Cities." *American Review of Respiratory Diseases* 129:366–374.

White, H. (1982). "Maximum Likelihood Estimation of Misspecified Models." *Econometrica* 50:1–25.

Williams, D. A. (1987). "Generalized Linear Model Diagnostics Using the Deviance and Single Case Deletions." *Journal of the Royal Statistical Society, Series C* 36:181–191.

Zeger, S. L., Liang, K.-Y., and Albert, P. S. (1988). "Models for Longitudinal Data: A Generalized Estimating Equation Approach." *Biometrics* 44:1049–1060.

Chapter 15
ICA Procedure

Contents

Overview: ICA Procedure **796**
PROC ICA Features 796
Using CAS Sessions and CAS Engine Librefs 797
Getting Started: ICA Procedure **798**
Syntax: ICA Procedure **803**
PROC ICA Statement 803
BY Statement 806
DISPLAY Statement 807
DISPLAYOUT Statement 808
OUTPUT Statement 809
VAR Statement 810
Details: ICA Procedure **810**
Computing Independent Components 810
FastICA Algorithm 810
InfoMax Algorithm 812
Whitening and Dimension Reduction 812
Missing Values 813
Displayed Output 813
Model Information 813
Dimensions 813
Number of Observations 813
Centering and Scaling Information 813
Eigenvalues 813
Whitening Transformation Matrix 813
Dewhitening Transformation Matrix 814
Demixing Matrix 814
Mixing Matrix 814
OutputCasTables Table 814
ODS Table Names 814
Examples: ICA Procedure **815**
Example 15.1: Extracting Independent Components with Dimension Reduction 815
References **819**

Overview: ICA Procedure

The ICA procedure performs independent component analysis in SAS Viya.

Independent component analysis attempts to extract from the observed multivariate data *independent components* (also called *factors* or *latent variables*) that are as statistically independent from each other as possible. The methods that PROC ICA implements, where statistical independence is defined as the maximization of non-Gaussianity, seek linear combinations of the observed random variables that maximize the non-Gaussianity of the estimated independent components.

Independent component analysis is one of the techniques for performing blind source separation or blind signal separation, where source means an original signal or independent component, and blind means that the mixing process of the source signals is unknown and few assumptions about the source signals are made. You can use independent component analysis to reveal the hidden structure of the data in many applications, including feature extraction and signal separation.

PROC ICA Features

The ICA procedure has the following main features:

- provides a VAR statement to specify the numeric variables for the analysis
- supports whitening and dimension reduction
- produces an output data table that contains independent components and whitened variables

PROC ICA implements the following methods in parallel, based on the FastICA algorithm of Hyvärinen and Oja (2000):

- symmetric decorrelation, which calculates all the independent components simultaneously
- deflationary decorrelation, which extracts the independent components successively

In addition, PROC ICA implements the following methods, based on the InfoMax algorithm of Bell and Sejnowski (1995):

- geodesic InfoMax, which uses geodesic-based gradient updates
- symmorth InfoMax, which uses a symmetric orthogonalized version of gradient updates
- unconstrained InfoMax, which uses Euclidean gradient updates

Because PROC ICA runs on SAS Cloud Analytic Services (CAS), it also does the following:

- enables you to run on a cluster of machines that distribute the data and the computations

- enables you to run in single-machine mode
- exploits all the available cores and concurrent threads. For information about how PROC ICA uses threads, see the section "Multithreading" on page 102 in Chapter 2, "Shared Concepts."

Using CAS Sessions and CAS Engine Librefs

SAS Cloud Analytic Services (CAS) is the analytic server and associated cloud services in SAS Viya. This section describes how to create a CAS session and set up a CAS engine libref that you can use to connect to the CAS session. It assumes that you have a CAS server already available; contact your system administrator if you need help starting and terminating a server. This CAS server is identified by specifying the host on which it runs and the port on which it listens for communications. To simplify your interactions with this CAS server, the host information and port information for the server are stored as SAS option values that are retrieved automatically whenever this CAS server needs to be accessed. You can examine the host and port values for the server at your site by using the following statements:

```
proc options option=(CASHOST CASPORT);
run;
```

In addition to starting a CAS server, your system administrator might also have created a CAS session and a CAS engine libref for your use. You can define your own sessions and CAS engine librefs that connect to the CAS server as shown in the following statements:

```
cas mysess;
libname mylib cas sessref=mysess;
```

The CAS statement creates the CAS session named mysess, and the LIBNAME statement creates the mylib CAS engine libref that you use to connect to this session. It is not necessary to explicitly name the CASHOST and CASPORT of the CAS server in the CAS statement, because these values are retrieved from the corresponding SAS option values.

If you have created the mysess session, you can terminate it by using the TERMINATE option in the CAS statement as follows:

```
cas mysess terminate;
```

For more information about the CAS and LIBNAME statements, see the section "Introduction to Shared Concepts" on page 6 in Chapter 2, "Shared Concepts."

Getting Started: ICA Procedure

(View the complete code for this example at https://github.com/sascommunities/doc-supplement-casstat/tree/main/Examples/icags.sas.)

NOTE: Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections "Using CAS Sessions and CAS Engine Librefs" on page 7 and "Loading a SAS Data Set onto a CAS Server" on page 7 in Chapter 2, "Shared Concepts."

The following DATA step creates the mylib.sample data table, which provides simulated signal data, in your CAS session. The data for this example are the result of multiplying the source signal matrix $\mathbf{S}$ and the mixing matrix $\mathbf{A}$; that is, the data matrix $\mathbf{X} = \mathbf{SA}$. Figure 15.1 shows the original source signals. Figure 15.2 shows the observed mixtures of the source signals. The problem is to recover the original source signals (s1, s2, and s3) shown in Figure 15.1 only from the observed signal mixtures (x1, x2, and x3) shown in Figure 15.2.

```
data mylib.sample;
   keep t x:;
   array S[200,3];     /* S: source signals */
   array A[3,3];       /* A: mixing matrix */
   array x[3] x1-x3;   /* X: mixed signals */

   N = 200;

   do i = 1 to 3;
      do j = 1 to 3;
         A[i,j] = 0.7*uniform(12345);
      end;
   end;

   do i = 1 to N;
      S[i,1] = cos(i/3);
      S[i,2] = 0.4*((mod(i,23)-11)/7)**5;
      S[i,3] = ((mod(i,29)-7)/11)-0.7;
   end;

   do i = 1 to N;
      t = i;
      do j = 1 to 3;
         x[j] = 0;
         do k = 1 to 3;
            x[j] = x[j] + S[i,k]*A[k,j];
         end;
      end;
      output;
   end;
run;
```

Figure 15.1 Original Source Signals

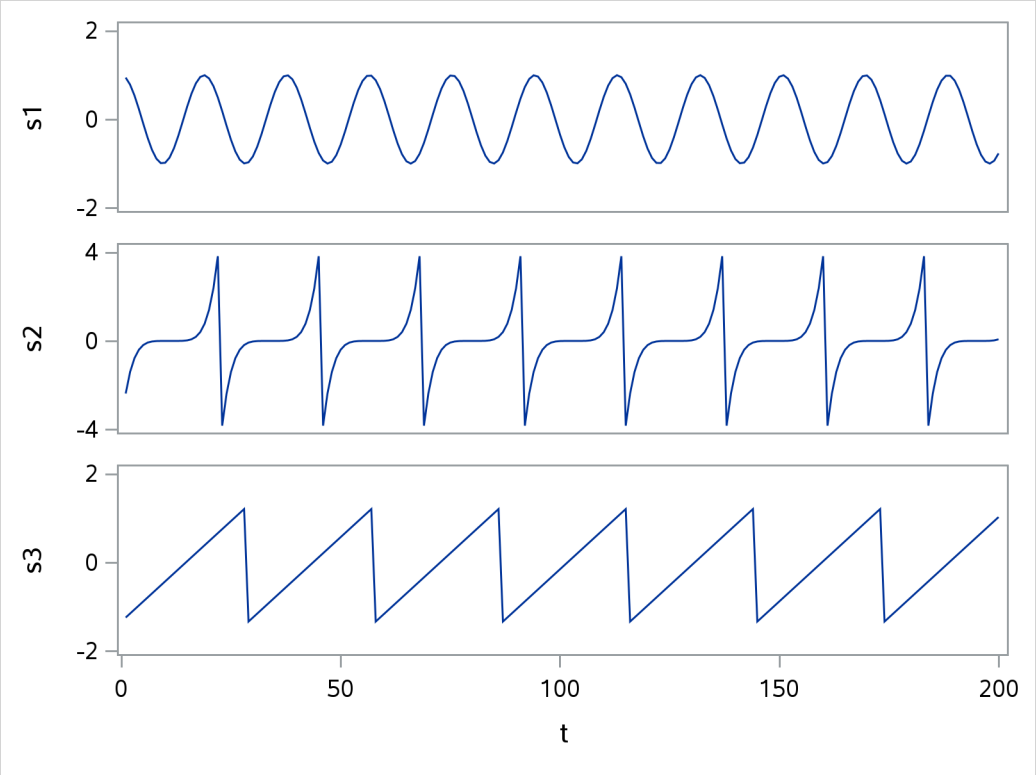
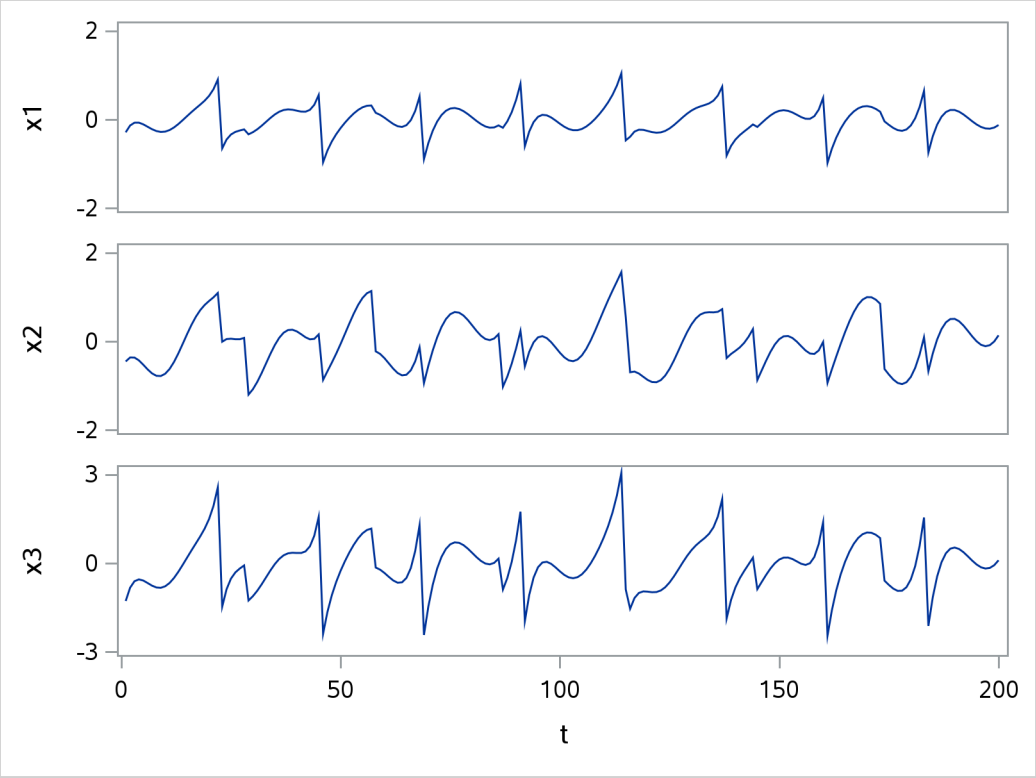


Figure 15.2 Observed Signal Mixtures



These statements assume that your CAS engine libref is named mylib, but you can substitute any appropriately defined CAS engine libref.

The following statements invoke the ICA procedure, which requests the independent component analysis of the data, outputs the computed independent components to an output data table, and produces the tables in Figure 15.3 through Figure 15.6:

```
proc ica data=mylib.sample seed=345;
   var x1-x3;
   output out=mylib.scores component=c copyvar=t;
run;
```

Figure 15.3 displays the "Model Information," "Dimensions," "Number of Observations," and "Centering and Scaling Information" tables.

The "Model Information" table identifies the data source and shows that the independent component extraction method is symmetric decorrelation, which is the default. The nonquadratic function $\log\cosh$ is used in the approximation of negentropy, and the eigenvalue proportion threshold is set to 0; these are the defaults. The random number seed is set to 345. Random number generation is used to initialize the demixing matrix. Changing the random number seed value changes the initial demixing matrix, which yields a different estimated demixing matrix.

The "Dimensions" table indicates that there are three variables to be analyzed and three independent components to be computed. If you omit the N= option in the PROC ICA statement, the default is the number of numeric variables to be analyzed. The table also shows that there are three whitened variables and that three independent components are actually extracted.

The "Number of Observations" table shows that all 200 of the sample observations in the input data are used in the analysis; all the samples are used because they all contain complete data.

The "Centering and Scaling Information" table displays the centering and scaling information of the analysis variables.

Figure 15.3 Model Information, Dimensions, Number of Observations, and Centering and Scaling Information

The ICA Procedure

Model Information	
Data Source	SAMPLE
Component Extraction Method	Symmetric Decorrelation
Negentropy Approximation Function	Log-cosh
Eigenvalue Proportion Threshold	0
Random Number Seed	345

Dimensions	
Number of Variables	3
Number of Whitened Variables	3
Number of Independent Components	3
Number of Independent Components Extracted	3

Number of Observations Read	200
Number of Observations Used	200

Figure 15.3 *continued*

Centering and Scaling Information		
Variable	Subtracted off	Divided by
x1	-0.01231	0.32312
x2	-0.04632	0.56436
x3	-0.05567	0.89319

Figure 15.4 displays the "Eigenvalues" table.

Figure 15.4 Eigenvalues Table

Eigenvalues				
	Eigenvalue	Difference	Proportion	Cumulative
1	2.663628	2.352835	0.8879	0.8879
2	0.310793	0.285215	0.1036	0.9915
3	0.025578		0.0085	1.0000

Figure 15.5 displays the "Whitening Transformation Matrix" and "Dewhitening Transformation Matrix" tables. In the whitening transformation matrix that is shown, the whitened variables are represented as a linear combination of the original variables that are centered and scaled. The dewhitening transformation matrix is the pseudoinverse of the whitening matrix.

Figure 15.5 Whitening and Dewhitening Transformation Matrices

Whitening Transformation Matrix			
Variable	White1	White2	White3
x1	-0.35446	-1.01630	3.66902
x2	-0.33541	1.44039	1.47348
x3	-0.37052	-0.33165	-4.84386

Dewhitening Transformation Matrix			
Whitened Variable	x1	x2	x3
White1	-0.94415	-0.89341	-0.98693
White2	-0.31586	0.44766	-0.10307
White3	0.09385	0.03769	-0.12390

Figure 15.6 displays the "Demixing Matrix" and "Mixing Matrix" tables. In the demixing matrix that is shown, the independent components are represented as a linear combination of the original variables that are standardized. The mixing matrix is the pseudoinverse of the demixing matrix.

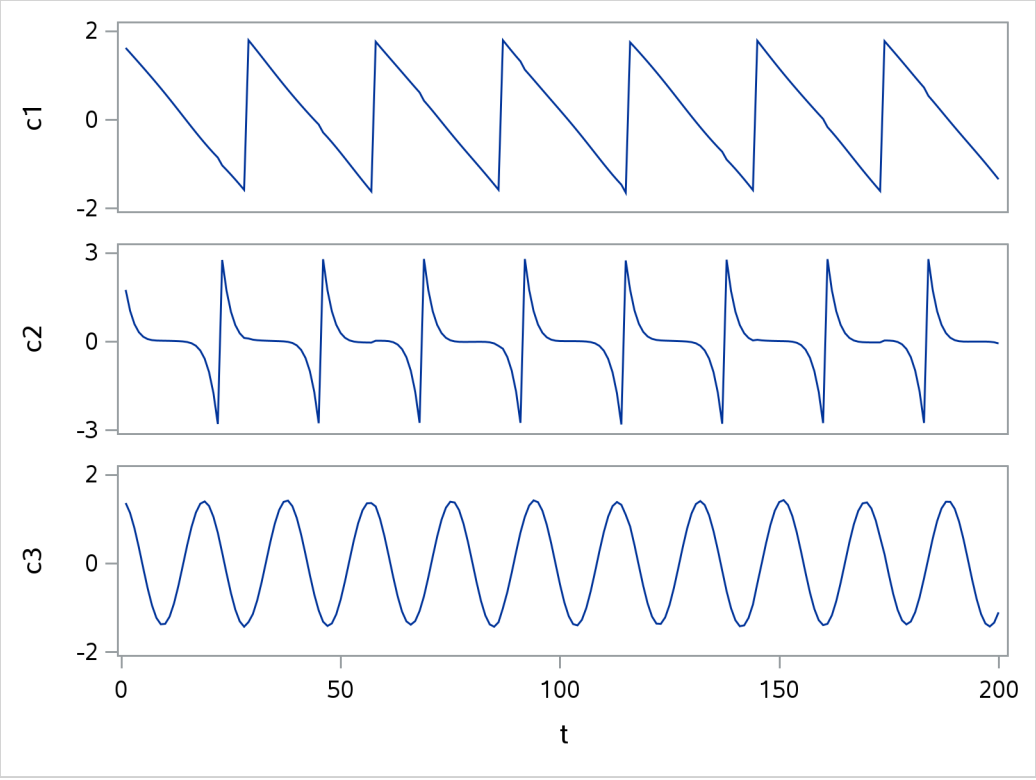
Figure 15.6 Demixing and Mixing Matrices

Demixing Matrix			
Variable	**Comp1**	**Comp2**	**Comp3**
x1	2.51490	0.32078	2.86227
x2	-0.43425	1.17723	1.66851
x3	-2.44766	-2.07527	-3.66231

Mixing Matrix			
Component	**x1**	**x2**	**x3**
Comp1	-0.12351	-0.69340	-0.41244
Comp2	-0.82569	-0.32078	-0.79147
Comp3	0.55043	0.64520	0.45108

Using the output data table mylib.scores and PROC SGPLOT or PROC SGRENDER (code not shown), you can plot the computed independent components. Figure 15.7 shows three independent components (c1, c2, and c3) that are the estimates of the three original source signals (s1, s2, and s3) in Figure 15.1. The original source signals are accurately estimated from the observed signal mixtures shown in Figure 15.2, up to multiplicative signed scalars. The order of the computed independent components cannot be determined in the independent component analysis model.

Figure 15.7 Estimated Source Signals



Syntax: ICA Procedure

The following statements are available in the ICA procedure:

```
PROC ICA < options > ;
    BY variables ;
    DISPLAY < table-list > < / options > ;
    DISPLAYOUT table-spec-list < / options > ;
    OUTPUT OUT=libref.data-table
            < COPYVARS=(variables) >
            < keyword < =prefix > >. . . < keyword < =prefix > > ;
    VAR variables ;
```

The PROC ICA statement is required. All other statements are optional. The following sections describe the PROC ICA statement and then describe the other statements in alphabetical order.

PROC ICA Statement

```
PROC ICA < options > ;
```

The PROC ICA statement invokes the ICA procedure. Table 15.1 summarizes the options available in the PROC ICA statement.

Table 15.1 PROC ICA Statement Options

Option	Description
Basic Options	
DATA=	Specifies the input data table
Analysis Options	
EIGTHRESH=	Specifies the threshold for the proportion of variance explained by eigenvalues
GFUNCTION=	Specifies the nonquadratic function to be used in the approximation of negentropy
METHOD=	Specifies the independent component extraction method to be used
N=	Specifies the number of independent components to be computed
NOCENTER	Suppresses centering of the numeric variables to be analyzed
NOSCALE	Suppresses scaling of the numeric variables to be analyzed
PREFIX=	Specifies a prefix for naming the independent components
SEED=	Specifies the seed value for pseudorandom number generation
WHITEPREFIX=	Specifies a prefix for naming the whitened variables

The following list provides details about these *options*.

DATA=*libref.data-table*

names the input data table for PROC ICA to use. The default is the most recently created data table. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to the data, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 797.

data-table specifies the name of the input data table.

EIGTHRESH=*p*

EIGENTHRESHOLD=*p*

specifies the threshold for the proportion of variance explained by eigenvalues, where $0 \leq p \leq 1$. If the proportion of variance that an eigenvalue explains is less than the threshold, the eigenvalue is discarded in the analysis. You can use this option to reduce the dimensionality of the input data. For more information, see the section "Whitening and Dimension Reduction" on page 812. By default, EIGTHRESH=0.

GFUNCTION=EXP | LOGCOSH

GFUNC=EXP | LOGCOSH

specifies the nonquadratic function $G(x)$ to be used in the approximation of negentropy. You can specify the following function types:

EXP

uses the exponential function, $G(u) = -e^{-u^2/2}$.

LOGCOSH

uses the $\log\cosh$ function, $G(u) = \log\cosh(u)$.

By default, GFUNCTION=LOGCOSH.

METHOD=*method***<(***options***)>**

specifies the independent component extraction method to be used. You can specify the following *methods*:

DEFLATION<(*options***)>**

requests the deflationary decorrelation method. You can specify the following *options*:

MAXITER=*n* specifies the maximum number of iterations to perform. By default, MAXITER=500.

TOL=*n* specifies the convergence criterion. By default, TOL=1E–4.

GEODESIC<(*options***)>**

requests the geodesic InfoMax method. You can specify the following *options*:

EXTENDED=0 | 1 | 2 specifies the extended option to use. All the sources are assumed to be super-Gaussians for option 0 and sub-Gaussians for option 2. For option 1, the types of sources are estimated. By default, EXTENDED=1.

LRATE=*n* specifies the learning rate to use. By default, LRATE=0.1.

MAXITER=*n* specifies the maximum number of iterations to perform. By default, MAXITER=500.

TOL=*n* specifies the convergence criterion. By default, TOL=1E–4.

SYMMETRIC< (*options***) >**
requests the symmetric decorrelation method. You can specify the following *options*:

MAXITER=*n* specifies the maximum number of iterations to perform. By default, MAXITER=500.

TOL=*n* specifies the convergence criterion. By default, TOL=1E–4.

SYMMORTH< (*options***) >**
requests the symmorth InfoMax method. You can specify the following *options*:

EXTENDED=0 | 1 | 2 specifies the extended option to use. All the sources are assumed to be super-Gaussians for option 0 and sub-Gaussians for option 2. For option 1, the types of sources are estimated. By default, EXTENDED=1.

LRATE=*n* specifies the learning rate to use. By default, LRATE=0.1.

MAXITER=*n* specifies the maximum number of iterations to perform. By default, MAXITER=500.

TOL=*n* specifies the convergence criterion. By default, TOL=1E–4.

UNCONSTRAINED< (*options***) >**
requests the unconstrained InfoMax method. You can specify the following *options*:

EXTENDED=0 | 1 | 2 specifies the extended option to use. All the sources are assumed to be super-Gaussians for option 0 and sub-Gaussians for option 2. For option 1, the types of sources are estimated. By default, EXTENDED=1.

LRATE=*n* specifies the learning rate to use. By default, LRATE=0.1.

MAXITER=*n* specifies the maximum number of iterations to perform. By default, MAXITER=500.

TOL=*n* specifies the convergence criterion. By default, TOL=1E–4.

By default, METHOD=SYMMETRIC.

N=*number*
specifies the number of independent components to be computed. The default is the number of numeric variables to be analyzed. The *number* must be an integer greater than or equal to 0. If N=0, the default is used.

NOCENTER
suppresses centering of the numeric variables to be analyzed. This option is useful if the analysis variables are already centered and scaled.

NOSCALE
suppresses scaling of the numeric variables to be analyzed. This option is useful if the analysis variables are already centered and scaled.

PREFIX=*name*
specifies a prefix for naming the independent components. By default, the *names* are Comp1, Comp2, . . . , Comp*n*. If you specify PREFIX=Abc, the components are named Abc1, Abc2, Abc3, and so on. The number of characters in the prefix plus the number of digits required to designate the variables should not exceed the current name length that is defined by the VALIDVARNAME= system option.

SEED=*number*
specifies an integer as the seed value for pseudorandom number generation. PROC ICA uses random values to initialize the demixing matrix. Changing the seed value changes the initial demixing matrix, resulting in a different estimated demixing matrix. If you do not specify this option or if *number* is less than or equal to 0, the seed is generated by reading the time of day from the computer's clock. By default, SEED=0.

WHITEPREFIX=*name*
WPREFIX=*name*
specifies a prefix for naming the whitened variables. By default, the *names* are White1, White2, . . . , White*n*. If you specify WHITEPREFIX=Abc, the components are named Abc1, Abc2, Abc3, and so on. The number of characters in the prefix plus the number of digits required to designate the variables should not exceed the current name length that is defined by the VALIDVARNAME= system option.

BY Statement

BY *variables* **;**

You can specify a BY statement in PROC ICA to obtain separate analyses of observations in groups that are defined by the values of the BY variables. If you specify more than one BY statement, only the last one specified is used. For more information, see the discussion of BY-group processing in the "Grouping Data" section of *SAS Programmers Guide: Essentials*.

DISPLAY Statement

DISPLAY < *table-list* > < / *options* > **;**

The DISPLAY statement enables you to specify a list of display tables to display or exclude. This statement is similar to the ODS SELECT, ODS EXCLUDE, and ODS TRACE statements. However, the DISPLAY statement can improve performance when a large number of tables could be generated (such as in BY-group processing). The procedure processes the DISPLAY statement on a CAS server and thus sends only a subset of ODS tables to the SAS client. Because ODS statements are processed on a SAS client, first all the generated display tables are sent to the client, and then the client creates a subset.

If you use both DISPLAY and ODS statements together, the DISPLAY statement takes precedence over the ODS statements. Note that the ODS EXCLUDE statement processes tables that are sent to the client after they have been filtered by the DISPLAY statement. In some cases, it might appear that the ODS EXCLUDE statement is taking precedence because it can further filter the tables. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

You can specify the *table-list* as a list of table names, paths, partial pathnames, and regular expressions.

The table names that you can specify are listed in the section "ODS Table Names" on page 814. A path is a table name that is prefixed with dot-separated grouping information. For example, a SelectionSummary table that a procedure produces during a selection routine might have the path *Bygroup1.Summary.SelectionSummary*. A partial pathname does not include all groups; for example, *SelectionSummary* and *Summary.SelectionSummary* are partial pathnames for *Bygroup1.Summary.SelectionSummary*.

When you specify a table name or partial pathname, all display tables whose paths end in the specified name are selected for display or exclusion. For example, both *SelectionSummary* and *Summary.SelectionSummary* select *Bygroup1.Summary.SelectionSummary*.

A regular expression is enclosed in forward slashes (/). For example, specifying "/tions/" selects all pathnames that contain the substring "tions"; in particular, the *Bygroup1.Summary.SelectionSummary* table is selected. Specifying "!/tions/" selects all pathnames that do not contain the substring "tions"; in particular, the *Bygroup1.Summary.SelectionSummary* table is not selected.

You can specify the following *options* after a slash (/):

CASESENSITIVE
performs a case-sensitive comparison of table names in the *table-list* to display table names when tables are subsetted for display. To preserve case, you must enclose table names in the *table-list* in quotation marks.

EXCLUDE
displays all display tables except those that you specify in the *table-list*.

EXCLUDEALL
suppresses display of all tables. This option takes precedence over the other options.

TRACE
displays the display table names, labels, and paths.

DISPLAYOUT Statement

DISPLAYOUT *table-spec-list* < / *options* > **;**

The DISPLAYOUT statement enables you to create output tables from your displayed output. This statement is similar to the ODS OUTPUT statement. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

The *table-spec-list* specifies a list of output tables to create. Each entry in the list has either a *key*=*value* format or a *key* format:

key=*value* specifies *key* as the ODS table name, path, or partial pathname, and specifies *value* as the output table name.

key specifies *key* as the ODS table name and also as the output table name.

The ODS table names that you can specify are listed in the section "ODS Table Names" on page 814. You cannot specify the ODS table named OutputCasTables in the *table-spec-list*.

Table names and partial pathnames are discussed under the DISPLAY statement. The DISPLAYOUT statement does not support regular expressions.

You can specify the following *options* after a slash (/):

INCLUDEALL
creates output tables for all display tables. The name of the created output table is the same as the name of the corresponding display table. If you specify this option, the *table-spec-list* specification is ignored.

NOREPLACE
does not replace any existing output table of the same name.

REPEATED
replicates all output tables on all nodes.

OUTPUT Statement

OUTPUT OUT=*libref.data-table*
< **COPYVARS=***(variables)* >
< *keyword* < **=***prefix* > > . . . < *keyword* < **=***prefix* > > **;**

The OUTPUT statement creates a data table that contains observationwise statistics, which are computed after PROC ICA fits the model. If you do not specify a *keyword*, then only the independent component values are included.

The variables in the input data table are *not* included in the output data table, in order to avoid data duplication for large data tables; however, variables that you specify in the COPYVARS= option are included.

You must specify the following option:

OUT=*libref.data-table*
names the output data table for PROC ICA to use. You must specify this option before any other options. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to where the data table is to be stored, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 797.

data-table specifies the name of the output data table.

You can also specify the following syntax elements:

COPYVAR=*variable*

COPYVARS=(*variables***)**
transfers one or more *variables* from the input data table to the output data table.

keyword **< =***prefix* **>**
specifies a statistic to include in the output data table and optionally a *prefix* for naming the output variables. If you do not provide a *prefix*, PROC ICA assigns a default prefix that is based on the type of statistic requested.

You can specify the following *keywords* to add statistics to the output data table:

COMP

COMPONENT
requests the source values for each independent component. The default prefix is the one that you specify in the PREFIX= option.

WHITE
requests the source values for each whitened variable. The default prefix is the one that you specify in the WHITEPREFIX= option.

The output variables that contain the requested statistic are named as follows, according to the *keyword* that you specify:

- The *keyword* COMP defines an output variable for each independent component, so the variables that correspond to each successive component are named by appending a number to the prefix. For example, if the model has three independent components, then COMP=C produces the variables C1, C2, and C3.
- The *keyword* WHITE defines an output variable for each whitened variable, so the variables that correspond to each successive whitened variable are named by appending a number to the prefix. For example, if the model has three whitened variables, then WHITE=W produces the variables W1, W2, and W3.

VAR Statement

VAR *variables* **;**

The VAR statement lists the numeric variables to be analyzed. If you omit the VAR statement, all numeric variables that are not specified in other statements are analyzed.

Details: ICA Procedure

Computing Independent Components

Independent component analysis defines a generative model for the observed data. In the model, the observed data variables are considered to be linear combinations of the latent variables that are non-Gaussian and maximally independent; that is, $\mathbf{X} = \mathbf{SA}$, where $\mathbf{X}$ is the data matrix, $\mathbf{S}$ are the latent variables or independent components, and $\mathbf{A}$ is the mixing matrix.

FastICA Algorithm

By applying the central limit theorem, you can estimate the independent components by maximizing the non-Gaussianity of $\mathbf{XW}$, where $\mathbf{W}$ is called the demixing matrix, the pseudoinverse of $\mathbf{A}$.

Let $\mathbf{X}$ be the centered and scaled data matrix that is whitened by using the eigenvalue decomposition, and let $\mathbf{w}_i$ be an orthonormal weight vector for the ith independent component. The FastICA algorithm of Hyvärinen and Oja (2000) that the ICA procedure implements is based on a fixed-point iteration scheme for finding a maximum of the non-Gaussianity of $\mathbf{Xw}_i$, which is measured by the approximation of negentropy,

$$J(u) \propto \Big(E[G(u)] - E[G(\nu)]\Big)^2$$

where the random variable u is assumed to be of zero mean and unit variance, ν is a Gaussian variable with zero mean and unit variance, and G is a nonquadratic function. PROC ICA provides the following choices of G as options: $G(u) = -e^{-u^2/2}$ and $G(u) = \log\cosh(u)$. The fixed-point iteration scheme in the FastICA algorithm is derived as an approximative Newton iteration,

$$\mathbf{w}_i \leftarrow E[\mathbf{X}^T g(\mathbf{X}\mathbf{w}_i)] - E[g'(\mathbf{X}\mathbf{w}_i)]\mathbf{w}_i$$

where g is the derivative of G.

To prevent the estimated weight vectors $\mathbf{w}_1, \ldots, \mathbf{w}_n$ from converging to the same maxima, the projections $\mathbf{X}\mathbf{w}_1, \ldots, \mathbf{X}\mathbf{w}_n$ must be decorrelated after every iteration. PROC ICA implements two methods to achieve this: the deflationary decorrelation method and the symmetric decorrelation method. The deflationary approach achieves decorrelation on the basis of the Gram-Schmidt orthogonalization method, which means that the independent components are computed sequentially:

$$\mathbf{w}_i \leftarrow \mathbf{w}_i - \sum_{j=1}^{i-1} (\mathbf{w}_j \mathbf{w}_j^T)\mathbf{w}_i$$

A problem with the deflationary decorrelation approach is the accumulation of numeric errors.

The symmetric approach estimates the independent components simultaneously by performing symmetric orthogonalization of the matrix $\mathbf{W} = (\mathbf{w}_1, \ldots, \mathbf{w}_n)$. This is accomplished by the classic method that involves matrix square roots:

$$\mathbf{W} \leftarrow \mathbf{W}\left(\mathbf{W}^T\mathbf{W}\right)^{-\frac{1}{2}}$$

The inverse square root is obtained from the eigenvalue decomposition of $\mathbf{W}^T\mathbf{W} = \mathbf{E}\mathbf{D}\mathbf{E}^T$ as $(\mathbf{W}^T\mathbf{W})^{-1/2} = \mathbf{E}\mathbf{D}^{-1/2}\mathbf{E}^T$, where $\mathbf{E}$ is an orthogonal matrix whose entries are the eigenvectors of $\mathbf{W}^T\mathbf{W}$ and $\mathbf{D}$ is a diagonal matrix whose entries are the eigenvalues of $\mathbf{W}^T\mathbf{W}$.

In the independent component analysis model, the variances and the order of the independent components cannot be determined. Each of the independent components is thus assumed to have unit variance in the computation because they are random variables. However, the sign is still indeterminable. By default, PROC ICA calculates the independent component scores by using the scaled input data and the estimated demixing matrix. You can use the NOSCALE option in the PROC ICA statement to suppress scaling; PROC ICA then calculates the independent component scores by using the original input data and the estimated demixing matrix.

Independent component analysis is closely related to projection pursuit, a technique that usually attempts to find the most non-Gaussian projections of multidimensional data. If the generative model holds for the data, optimizing the non-Gaussianity measures produces independent components; if the model is absent for the data, independent component analysis produces the projection pursuit directions.

InfoMax Algorithm

Given the data matrix X, you want to find the demixing matrix $\mathbf{W}$ so that the joint entropy of $\mathbf{Y} = \mathbf{XW}$ is maximized. It turns out that in this case, it also maximizes the mutual information between $\mathbf{X}$ and $\mathbf{Y}$. For this reason, this algorithm for extracting independent components is known as InfoMax (Bell and Sejnowski 1995). Essentially, the InfoMax algorithm is a parallel version of projection pursuit, but it uses the information maximization principle and maximum likelihood estimate instead of non-Gaussianity. The InfoMax algorithm uses gradient-based updates and learning rates, so it can be more adaptive in certain cases.

You can adapt the InfoMax algorithm to use the fact that the demixing matrix $\mathbf{W}$ must be orthogonal, such as by using geodesic-based gradient updates; see Nishimori (1999) for more information. To achieve this, PROC ICA implements three methods: geodesic InfoMax, symmorth InfoMax, and unconstrained InfoMax. The unconstrained InfoMax method uses the gradient updates of Bell and Sejnowski (1995), where the demixing matrix $\mathbf{W}$ can be any nonsingular matrix. The symmorth InfoMax method uses a symmetric orthogonalized version of the gradient updates, and the geodesic InfoMax method uses geodesic-based gradient updates on the orthogonal group.

Whitening and Dimension Reduction

Whitening can greatly simplify the complexity of the problem of independent component analysis by reducing the number of parameters to be estimated. A zero-mean random vector is said to be *white* if its elements are uncorrelated and have unit variances. A synonym for white is sphered.

By default, the input data variables are centered and scaled to have mean 0 and standard deviation 1. The centered and scaled data matrix $\mathbf{X}$ is then transformed linearly to become a new data matrix $\tilde{\mathbf{X}}$, which is *white*. In other words, the covariance matrix of $\tilde{\mathbf{X}}$ equals the identity matrix: $E[\tilde{\mathbf{X}}^T\tilde{\mathbf{X}}] = \mathbf{I}$. PROC ICA performs the whitening transformation by using the eigenvalue decomposition of the covariance matrix $E[\mathbf{X}^T\mathbf{X}] = \mathbf{EDE}^T$, where $\mathbf{E}$ is the orthogonal matrix of its eigenvectors and $\mathbf{D}$ is the diagonal matrix of its eigenvalues. Thus the whitening transformation matrix is given by $\mathbf{ED}^{-1/2}$, and whitening is done by $\tilde{\mathbf{X}} = \mathbf{XED}^{-1/2}$.

It can also be useful to reduce the dimensionality of the input data before extracting the independent components, which can usually reduce noise and prevent overlearning. PROC ICA performs dimension reduction and the whitening transformation at the same time by looking at the eigenvalues of $E[\mathbf{X}^T\mathbf{X}]$ and discarding those that are too small. You can use the EIGTHRESH= option in the PROC ICA statement to specify a threshold for the proportion of variance explained by the eigenvalues. An eigenvalue is discarded if the proportion of variance that it explains is less than the threshold.

When you use the EIGTHRESH= option for dimension reduction, you should suppress scaling by using the NOSCALE option in the PROC ICA statement. Scaling serves to place all dimensions of the space on an equal footing relative to their variation in the data. Performing dimension reduction with scaling poses the risk of dropping eigenvalues for the dimensions of the space that are spanned by the independent components but retaining those dimensions that are filled by noise.

Missing Values

Observations that have missing values for any variable in the VAR statement are omitted from the analysis and are given missing values for independent component values in the OUT= data table that is specified in the OUTPUT statement.

Displayed Output

The following sections describe the output that PROC ICA produces. The output is organized into various tables, which are discussed in their order of appearance.

Model Information

The "Model Information" table displays basic information about the model, including the input data table, the independent component extraction method, the nonquadratic function used in the negentropy approximation, and the eigenvalue proportion threshold.

Dimensions

The "Dimensions" table displays information about the number of VAR statement variables, the number of whitened variables after data whitening and dimension reduction, the number of independent components to extract, and the number of independent components that are actually extracted.

Number of Observations

The "Number of Observations" table displays the number of observations that are read from the input data table and the number of observations that are used in the analysis.

Centering and Scaling Information

The "Centering and Scaling Information" table displays the centering and scaling information for each analysis variable.

Eigenvalues

The "Eigenvalues" table displays eigenvalues of the covariance matrix that is calculated from the input data, along with the difference between successive eigenvalues, the proportion of variance explained by each eigenvalue, and the cumulative proportion of variance explained.

Whitening Transformation Matrix

The "Whitening Transformation Matrix" table displays the whitening transformation matrix, which is used to perform whitening and dimension reduction on the input data.

Dewhitening Transformation Matrix

The "Dewhitening Transformation Matrix" table displays the dewhitening transformation matrix, which is the pseudoinverse of the whitening matrix.

Demixing Matrix

The "Demixing Matrix" table displays the demixing matrix, which is used to extract the independent components from the input data.

Mixing Matrix

The "Mixing Matrix" table displays the mixing matrix, which is the pseudoinverse of the demixing matrix.

OutputCasTables Table

The OutputCasTables table is a special table that has information about each CAS table that is created during a CAS action execution. The information for each CAS table consists of the CAS table name, the caslib in which the table resides, and the number of columns and rows in the CAS table. Because this table is not a typical ODS table that contains analytical results, you cannot include it in the *table-spec-list* in the DISPLAYOUT statement.

ODS Table Names

Each table that the ICA procedure creates has a name associated with it. You must use this name to refer to the table when you use the DISPLAY statement, the DISPLAYOUT statement, or the Output Delivery System (ODS) statements. These names are listed in Table 15.2.

Table 15.2 ODS Tables Produced by PROC ICA

Table Name	Description	Required Statement / Option
CenScaleInfo	Centering and scaling information	Default output
Demixing	Demixing matrix	Default output
Dewhitening	Dewhitening transformation matrix	Default output
Dimensions	Model dimensions	Default output
Eigenvalues	Eigenvalues	Default output
Mixing	Mixing matrix	Default output
ModelInfo	Model information	Default output
NObs	Number of observations read and used	Default output
OutputCasTables	See the section "OutputCasTables Table" on page 814	DISPLAYOUT statement or OUTPUT statement
Whitening	Whitening transformation matrix	Default output
Timing	Absolute and relative times of tasks that are performed by the procedure	Default output

Examples: ICA Procedure

The examples in this chapter are available in the GitHub repository located at https://github.com/sascommunities/doc-supplement-casstat.

NOTE: Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections "Using CAS Sessions and CAS Engine Librefs" on page 7 and "Loading a SAS Data Set onto a CAS Server" on page 7 in Chapter 2, "Shared Concepts."

Example 15.1: Extracting Independent Components with Dimension Reduction

(View the complete code for this example (icaex1.sas) in the example repository.)

This example demonstrates that applying dimension reduction in PROC ICA provides a better estimate of the independent components by using simulated data, compared to extracting the independent components without reducing the dimension of the data. The data that this example uses are from the Getting Started section, and an extra noise signal (x4) is added to the observed signal mixtures. The following DATA step generates the data:

```
data mylib.ex1data;
   keep t x:;
   array S[200,3];      /* S: source signals */
   array A[3,3];        /* A: mixing matrix */
   array x[4] x1-x4;    /* X: observed signals */

   N = 200;

   do i = 1 to 3;
      do j = 1 to 3;
         A[i,j] = 0.7*uniform(12345);
      end;
   end;

   do i = 1 to N;
      S[i,1] = cos(i/3);
      S[i,2] = 0.4*((mod(i,23)-11)/7)**5;
      S[i,3] = ((mod(i,29)-7)/11)-0.7;
   end;

   do i = 1 to N;
      t = i;
      do j = 1 to 3;
         x[j] = 0;
         do k = 1 to 3;
            x[j] = x[j] + S[i,k]*A[k,j];
         end;
      end;
```

```
      x[4] = 0.1*uniform(67890);
      output;
   end;
run;
```

These statements assume that your CAS engine libref is named mylib, but you can substitute any appropriately defined CAS engine libref.

The following statements extract the independent components by using dimension reduction and output the computed independent components to an output data table:

```
proc ica data=mylib.ex1data eigthresh=0.004 noscale seed=345;
   var x1-x4;
   output out=mylib.scores1 component=c copyvar=t;
run;
```

Output 15.1.1 displays the PROC ICA output. The "Model Information" table shows that the eigenvalue proportion threshold is set to 0.004. The "Dimensions" table indicates that there are four variables to be analyzed and four independent components to be computed. It also shows that three whitened variables are generated by using whitening and three independent components are actually extracted, thanks to the use of dimension reduction.

Output 15.1.1 Results of Independent Component Analysis with Dimension Reduction

The ICA Procedure

Model Information	
Data Source	EX1DATA
Component Extraction Method	Symmetric Decorrelation
Negentropy Approximation Function	Log-cosh
Eigenvalue Proportion Threshold	0.004
Random Number Seed	345

Dimensions	
Number of Variables	4
Number of Whitened Variables	3
Number of Independent Components	4
Number of Independent Components Extracted	3

Number of Observations Read	200
Number of Observations Used	200

Centering and Scaling Information

Variable	Subtracted off	Divided by
x1	-0.01231	1.00000
x2	-0.04632	1.00000
x3	-0.05567	1.00000
x4	0.05224	1.00000

Output 15.1.1 *continued*

Eigenvalues

	Eigenvalue	Difference	Proportion	Cumulative
1	1.129934	1.045020	0.9251	0.9251
2	0.084915	0.079040	0.0695	0.9946
3	0.005874	0.005123	0.0048	0.9994
4	0.000752		0.0006	1.0000

Whitening Transformation Matrix

Variable	White1	White2	White3
x1	-0.26624	1.12655	11.73306
x2	-0.44684	-2.96560	2.16685
x3	-0.78387	1.30782	-5.22151
x4	-0.00124	0.04711	0.78387

Dewhitening Transformation Matrix

Whitened Variable	x1	x2	x3	x4
White1	-0.30083	-0.50490	-0.88572	-0.00140
White2	0.09566	-0.25182	0.11105	0.00400
White3	0.06892	0.01273	-0.03067	0.00460

Demixing Matrix

Variable	Comp1	Comp2	Comp3
x1	0.96285	7.75333	8.82969
x2	2.08133	-0.76730	2.96125
x3	-2.31137	-2.73321	-4.09587
x4	0.09540	0.50802	0.59118

Mixing Matrix

Component	x1	x2	x3	x4
Comp1	-0.26709	-0.18133	-0.70731	-0.00299
Comp2	-0.04024	-0.39162	-0.36922	0.00468
Comp3	0.17732	0.36367	0.40146	0.00289

You can produce a plot of the computed independent components by using the output data table mylib.scores1 and the SGRENDER procedure, as shown in the following statements:

```
proc template;
   define statgraph ScoresPanel;
      beginGraph;
         layout lattice / rows=3
                          columns=1
                          rowgutter=10
                          columndatarange=unionall
                          order=packed;

         columnaxes;
            columnaxis / label="t";
         endcolumnaxes;
```

```
            layout overlay / yaxisopts=
               (linearopts=(viewmin=-3 viewmax=3 tickvaluelist=(-3 0 3)));
               seriesplot x=t y=c1;
            endlayout;

            layout overlay / yaxisopts=
               (linearopts=(viewmin=-2 viewmax=2 tickvaluelist=(-2 0 2)));
               seriesplot x=t y=c2;
            endlayout;

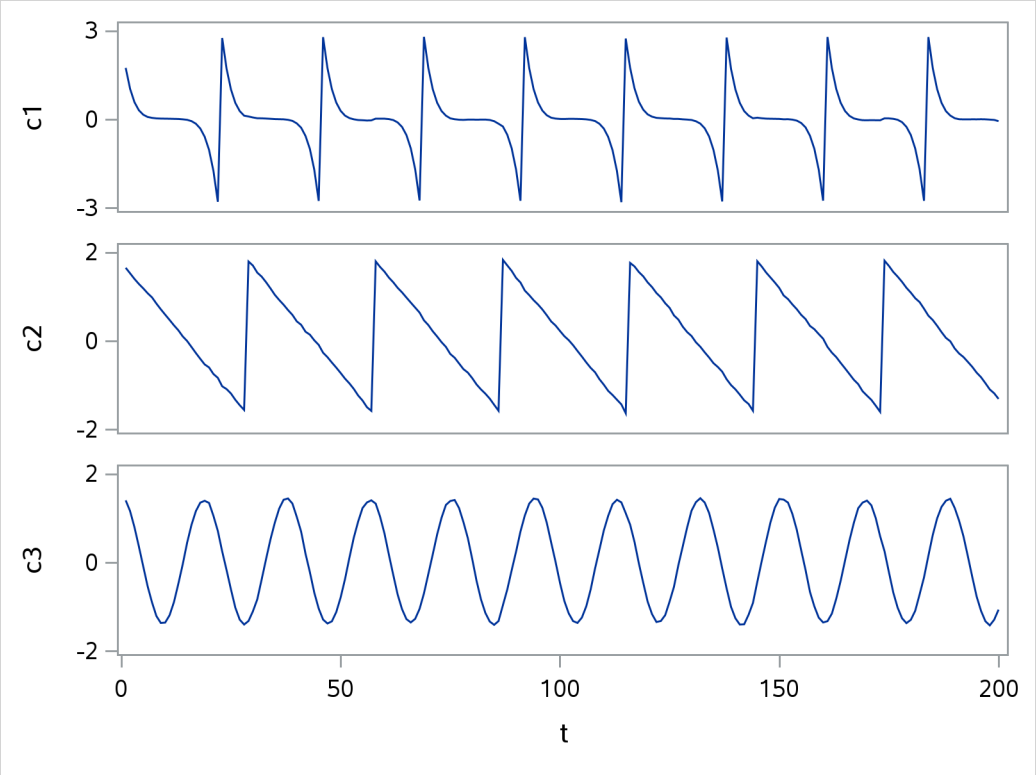
            layout overlay / yaxisopts=
               (linearopts=(viewmin=-2 viewmax=2 tickvaluelist=(-2 0 2)));
               seriesplot x=t y=c3;
            endlayout;

            endlayout;
         endGraph;
      end;
   run;

   proc sgrender data=mylib.scores1 template=ScoresPanel;
   run;
```

Output 15.1.2 displays the computed independent components. The original source signals (s1, s2, and s3) in Figure 15.1 are accurately estimated by the computed independent components (c1, c2, and c3) up to multiplicative signed scalars.

Output 15.1.2 Independent Components Computed with Dimension Reduction

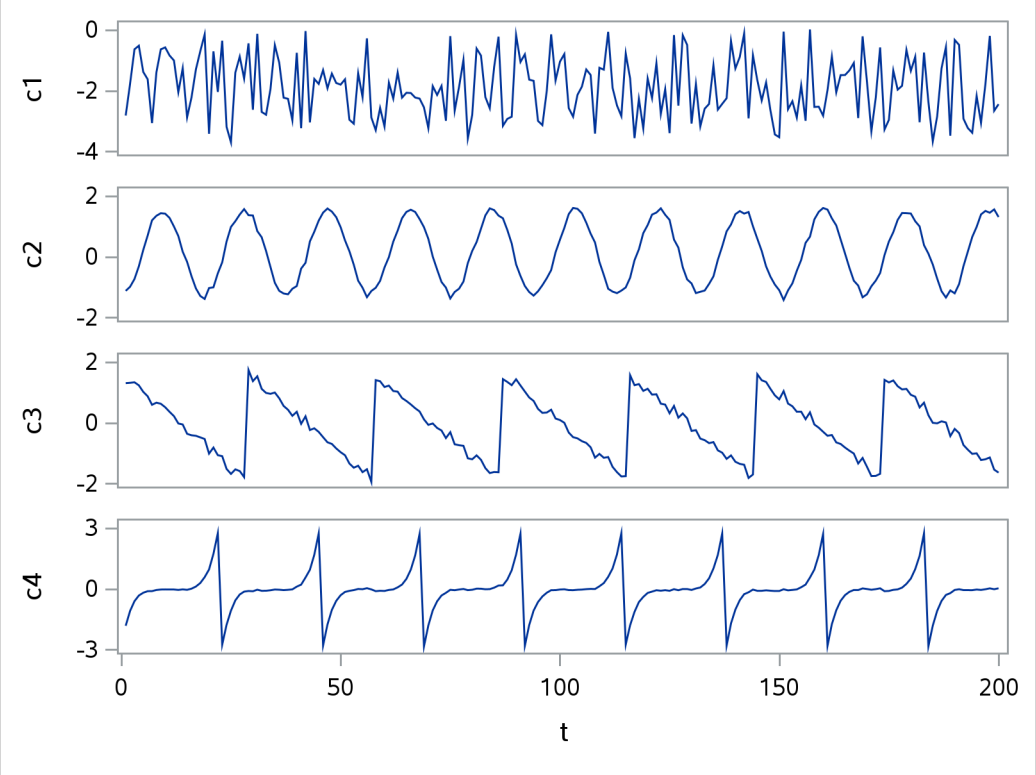


The following statements extract the independent components by using all dimensions of the input data (x1–x4) and output the computed independent components to an output data table:

```
proc ica data=mylib.ex1data noscale seed=345;
   var x1-x4;
   output out=mylib.scores2 component=c copyvar=t;
run;
```

Output 15.1.3 displays the computed independent components by using the output data table mylib.scores2 and PROC SGRENDER (code not shown). The original source signals (s1, s2, and s3) in Figure 15.1 are closely estimated by the three computed independent components (c2, c3, and c4) up to multiplicative signed scalars. However, you can clearly see that the estimates are affected by the addition of the noise signal (x4) compared to the independent components that are computed using dimension reduction in Output 15.1.2. Because there are fewer independent components than analysis variables in the input data, the computed component c1 is the projection pursuit direction.

Output 15.1.3 Independent Components Computed with Full Dimensions



References

Bell, A. J., and Sejnowski, T. J. (1995). "An Information-Maximization Approach to Blind Separation and Blind Deconvolution." *Neural Computation* 7:1129–1159.

Hyvärinen, A., Karhunen, J., and Oja, E. (2001). *Independent Component Analysis*. New York: John Wiley & Sons.

Hyvärinen, A., and Oja, E. (2000). "Independent Component Analysis: Algorithms and Applications." *Neural Networks* 13:411–430.

Nishimori, Y. (1999). "Learning Algorithm for Independent Component Analysis by Geodesic Flows on Orthogonal Group." In *Proceedings of the International Joint Conference on Neural Networks (IJCNN'99)*, vol. 2, 933–938. Piscataway, NJ: IEEE.

Chapter 16
KCLUS Procedure

Contents

Overview: KCLUS Procedure **822**
PROC KCLUS Features 823
Using CAS Sessions and CAS Engine Librefs 823
Getting Started: KCLUS Procedure **824**
Syntax: KCLUS Procedure **827**
PROC KCLUS Statement 828
CODE Statement 833
DISPLAY Statement 834
DISPLAYOUT Statement 835
FREQ Statement 836
INPUT Statement 836
SAVESTATE Statement 836
SCORE Statement 837
WEIGHT Statement 838
Details: KCLUS Procedure **838**
Obtaining the Statistics for Clustering 838
Missing Values 838
Initial Seed Selection 839
Standardization 839
Finding the Number of Clusters 839
Clustering Nominal Variables 840
Dissimilarity Measures 840
Computing Cluster Centers 841
Clustering Both Interval and Nominal Variables 842
Mixed Distance Measure 842
Weight of Nominal Distance 842
Score Output for Clustering Both Interval and Nominal Variables 842
Displayed Output 843
Number of Observations 843
Model Information 843
Cluster Summary 843
Iteration History 844
Descriptive Statistics 844
Within-Cluster Statistics 844
Cluster Summary for Nominal Variables 845
Frequencies for Nominal Variables 845

Cluster Summary for Mixed Variables 845
Standardization 846
Aligned Box Criterion Parameters 846
Aligned Box Criterion Statistics 846
Estimated Number of Clusters 846
OutputCasTables Table 847
ODS Table Names 847
Examples: KCLUS Procedure **848**
Example 16.1: Cluster Analysis 848
Example 16.2: Finding the Number of Clusters 852
Example 16.3: Clustering Nominal Variables 853
Example 16.4: Clustering Mixed Variables 855
References **856**

Overview: KCLUS Procedure

The KCLUS procedure performs clustering (a common step in data exploration) in SAS Viya.

You can use the KCLUS procedure to read and write data in distributed form, and to perform clustering and scoring in parallel by making full use of multicore computers or distributed computing environments.

The KCLUS procedure performs a cluster analysis on the basis of distances that are computed from quantitative or qualitative variables (or both). The observations are divided into clusters such that every observation belongs to one and only one cluster.

The KCLUS procedure uses the *k*-means algorithm for clustering interval input variables, uses the *k*-modes algorithm for clustering nominal input variables, and uses *k*-prototypes algorithm for clustering mixed input that contains both interval and nominal variables.

The KCLUS procedure accomplishes the clustering by updating the cluster centroids and the cluster membership of the data iteratively until the convergence criterion (for example, the least squares criterion for the Euclidean distance in *k*-means clustering) is satisfied or until the maximum iteration number is reached.

PROC KCLUS produces brief summaries of the cluster analysis in two output data tables:

- The OUT= data table is produced by the SCORE statement. This data table contains the cluster membership and distance variables for each observation in the input data table. It can be used for more detailed examination of the clusters.
- The OUTSTAT= data table is produced by the PROC KCLUS statement. This data table can be used for more detailed examination of between-cluster statistics.

PROC KCLUS Features

PROC KCLUS enables you to use parallel execution for clustering in a distributed computing environment. The following list summarizes the basic features of PROC KCLUS:

- uses the *k*-means method to cluster interval input variables
- uses the *k*-modes method to cluster nominal input variables
- uses the *k*-prototypes method to cluster mixed input variables
- provides a new technique called the aligned box criterion (ABC) for estimating the number of clusters in the data table
- enables you to run on a cluster of machines that distribute the data and the computations
- enables you to run in single-machine mode
- exploits all the available cores and concurrent threads

Using CAS Sessions and CAS Engine Librefs

SAS Cloud Analytic Services (CAS) is the analytic server and associated cloud services in SAS Viya. This section describes how to create a CAS session and set up a CAS engine libref that you can use to connect to the CAS session. It assumes that you have a CAS server already available; contact your system administrator if you need help starting and terminating a server. This CAS server is identified by specifying the host on which it runs and the port on which it listens for communications. To simplify your interactions with this CAS server, the host information and port information for the server are stored as SAS option values that are retrieved automatically whenever this CAS server needs to be accessed. You can examine the host and port values for the server at your site by using the following statements:

```
proc options option=(CASHOST CASPORT);
run;
```

In addition to starting a CAS server, your system administrator might also have created a CAS session and a CAS engine libref for your use. You can define your own sessions and CAS engine librefs that connect to the CAS server as shown in the following statements:

```
cas mysess;
libname mylib cas sessref=mysess;
```

The CAS statement creates the CAS session named mysess, and the LIBNAME statement creates the mylib CAS engine libref that you use to connect to this session. It is not necessary to explicitly name the CASHOST and CASPORT of the CAS server in the CAS statement, because these values are retrieved from the corresponding SAS option values.

If you have created the mysess session, you can terminate it by using the TERMINATE option in the CAS statement as follows:

```
cas mysess terminate;
```

For more information about the CAS and LIBNAME statements, see the section "Introduction to Shared Concepts" on page 6 in Chapter 2, "Shared Concepts."

Getting Started: KCLUS Procedure

(View the complete code for this example at https://github.com/sascommunities/doc-supplement-casstat/tree/main/Examples/kclus_gs.sas.)

NOTE: Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections "Using CAS Sessions and CAS Engine Librefs" on page 7 and "Loading a SAS Data Set onto a CAS Server" on page 7 in Chapter 2, "Shared Concepts."

This example shows how to use the KCLUS procedure to compute clusters of observations in an input data table.

Suppose you want to group the observations in the input CAS table mylib.inpData, in which the variables are raw measures on interval scales.

The following DATA step creates the input data table, mylib.inpData, in your CAS session. This data table contains four variables: the first two variables are the input variables among which x has missing values, the third variable is the frequency variable, and the last variable is an index variable.

```
data mylib.inpData;
   title 'Using PROC KCLUS to Analyze Data';
   drop n;
   id=1;
   do n=1 to 1000;
      x=2*rannor(12345)+20;
      y=4*rannor(12345)+20;
      freq = 1;
      id = id + 1;
      output;
   end;
   do n=1 to 1000;
      x=3*rannor(12345)+10;
      y=5*rannor(12345)+10;
      freq=2;
      id = id + 1;
      output;
   end;
   do n=1 to 700;
      x=10*rannor(12345);
      y=10*rannor(12345);
      freq=1;
      id = id + 1;
      output;
   end;
   do n=1 to 200;
```

```
      x=.;
      y=10*rannor(12345);
      freq=1;
      id = id + 1;
      output;
   end;
run;
```

These statements assume that your libref is named mylib, but you can substitute any appropriately defined libref.

The following statements run PROC KCLUS and output the results to ODS tables:

```
proc kclus data=mylib.inpData maxclusters=3;
   input x y;
   freq freq;
run;
```

Figure 16.1 shows that the "Number of Observations Used" is less than the "Number of Observations Read". By default, the KCLUS procedure ignores observations that have missing values, and it does not use them in the analysis. The two additional rows, "Sum of Frequencies Read" and "Sum of Frequencies Used" are displayed when the FREQ statement is specified. They provide information about the frequency values that are read and used.

Figure 16.1 Number of Observations

Using PROC KCLUS to Analyze Data

The KCLUS Procedure

Number of Observations Read	2900
Number of Observations Used	2700
Sum of Frequencies Read	3900
Sum of Frequencies Used	3700

Figure 16.2 shows the values of the parameters that are used in clustering. Because the number of clusters is not estimated by default and MAXCLUSTERS=3, three clusters are generated. Figure 16.2 shows the number of clusters and the default values for other options.

Figure 16.2 Model Information

Model Information	
Clustering Algorithm	K-means
Maximum Iterations	10
Stop Criterion	Cluster Change
Stop Criterion Value	0
Clusters	3
Initialization	Forgy
Seed	578406742
Distance for Interval Variables	Euclidean
Standardization	None
Interval Imputation	None

For each cluster, Figure 16.3 shows the number of observations; the maximum, minimum, and average distances from that cluster's centroid to the observations in that cluster; the sum of squares error; and the standard deviation. Figure 16.3 also displays information about the nearest cluster to that cluster and the distance between their centroids.

Figure 16.3 Cluster Summary

Cluster Summary for Interval Variables

		Distance from Cluster Centroid to Observation						
Cluster	Frequency	Minimum	Maximum	Average	SSE	Standard Deviation	Nearest Cluster	Distance to Nearest Cluster Centroid
1	2038	0.2289	31.7970	5.2019	78608.4	6.2106	2	13.8882
2	1175	0.0202	29.2304	4.4741	31968.1	5.2160	1	13.8882
3	487	0.5652	30.4447	10.7298	73156.7	12.2564	1	18.3687

Figure 16.4 shows the sum of squared errors (SSE) for each iteration. If the variables are interval, then the "Iteration History" table displays SSE Change and Stop Criterion columns. The SSE Change column displays the change in within-cluster distances. The Stop Criterion column displays the stopping criterion for each iteration. If the input variables are nominal, then the "Iteration History" table displays Within Distance Change and Stop Criterion columns. If the input variables are both interval and nominal, then the "Iteration History" table also displays Within Distance Change and Stop Criterion columns, but with the distance in the sense of mixed distance with both the interval and nominal parts as detailed in Clustering Both Interval and Nominal Variables.

Figure 16.4 Iteration History

Iteration History

Iteration Number	SSE	SSE Change	Stop Criterion
0	325431		
1	204706	-120725	8.851852
2	188187	-16519	3.407407
3	185097	-3089.291959	1.629630
4	184358	-739.528456	0.962963
5	184083	-275.000574	0.777778
6	183876	-206.393291	0.740741
7	183770	-106.591597	0.333333
8	183739	-30.920207	0.074074
9	183733	-5.722173	0
10	183733	0	0

Figure 16.5 and Figure 16.6 show statistics for each variable in the INPUT statement. Figure 16.5 shows the variable statistics for all the observations in the input data table, and Figure 16.6 shows the variable statistics for the observations that belong to a specific cluster.

Figure 16.5 Descriptive Statistics

Descriptive Statistics

Variable	Mean	Standard Deviation
x	11.020648	8.189686
y	10.246546	9.472050

Figure 16.6 Within-Cluster Statistics

Within Cluster Statistics

Variable	Cluster	Mean	Standard Deviation
x	1	9.5626	10.5049
	2	18.9930	6.0294
	3	-2.1130	9.0921
y	1	9.3852	11.0225
	2	19.5808	7.9047
	3	-4.7954	8.3923

Syntax: KCLUS Procedure

The following statements are available in the KCLUS procedure:

PROC KCLUS < *options* > **;**
CODE < *options* > **;**
DISPLAY < *table-list* > < */options* > **;**
DISPLAYOUT *table-spec-list* < */options* > **;**
FREQ *variable* **;**
INPUT *variables* < **LEVEL= NOMINAL** | **INTERVAL** > **;**
SCORE OUT=*libref.data-table* < *options* > **;**
SAVESTATE RSTORE=*libref.data-table* < *option* > **;**
WEIGHT *variable* **;**

The PROC KCLUS statement and an INPUT statement are required. You can specify multiple INPUT statements.

The following sections describe the PROC KCLUS statement and then describe the other statements in alphabetical order.

PROC KCLUS Statement

PROC KCLUS < *options* > **;**

The PROC KCLUS statement invokes the procedure. Table 16.1 summarizes the *options* available in this statement.

Table 16.1 PROC KCLUS Statement Options

Option	Description
Input and Output Data Set Options	
APPLYROWORDER	Specifies that the procedure use a prespecified data distribution
DATA=	Specifies the input data table
OUTSTAT(OUTITER)=	Specifies the output data table to contain cluster centroids
PRINTALLDISTANCES	Outputs the interval distance, nominal distance, and total distance (this option can be used only for mixed input variables)
Clustering Options	
DISTANCE=	Specifies the distance measure for similarity measurement (used for interval input variables)
DISTANCENOM=	Specifies the distance measure for similarity measurement (used for nominal input variables)
INIT=	Specifies the method for obtaining the initial estimate of cluster centers
MAXCLUSTERS=	Specifies the number of clusters
MAXITER=	Specifies the maximum number of iterations
SEED=	Specifies the seed that is used for pseudorandom number generation
STOPCRITERION=	Specifies the stop criterion method to use for convergence
KPROTOTYPEPARAMS=	Specifies the *k*-prototypes parameters for clustering interval and nominal variables at the same time (this option can be used only for mixed input variables)
Data Processing Options	
IMPUTE=	Specifies the imputation method for interval input variables
IMPUTENOM=	Specifies the imputation method for nominal input variables
STANDARDIZE=	Specifies the method for standardizing interval input variables
Number of Clusters Estimation Options	
NOC=	Specifies the method for estimating the number of clusters
Performance Options	
NTHREADS=	Specifies the number of threads for the computation

You can specify the following *options*:

APPLYROWORDER
uses a data distribution and row order as determined by a previous **partition** action call. For more information, see the section "The APPLYROWORDER Option" on page 107 in Chapter 2, "Shared Concepts."

DATA=*libref.data-table*
names the input data table for PROC KCLUS to use. The default is the most recently created data table. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to the data, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 823.

data-table specifies the name of the input data table.

DISTANCE=EUCLIDEAN | MANHATTAN
specifies the distance measure for similarity that is used for interval input variables.

You can choose from the following measures:

EUCLIDEAN calculates the Euclidean distance.

MANHATTAN calculates the Manhattan distance.

By default, DISTANCE=EUCLIDEAN.

DISTANCENOM=BINARY | GLOBALFREQ | RELATIVEFREQ
specifies the distance measure for similarity that is used for nominal input variables.

You can choose from the following measures:

BINARY calculates a simple matching distance.

GLOBALFREQ calculates the distance based on the frequency of levels of nominal input variables in the input data table.

RELATIVEFREQ calculates the distance based on the frequency of levels of nominal input variables in each cluster.

By default, DISTANCENOM=BINARY. For more information about these distance functions, see the section "Clustering Nominal Variables" on page 840.

INIT=FORGY | RAND
specifies the method for obtaining the initial estimate of cluster centers.

You can choose from the following methods:

FORGY selects the initial cluster centers randomly from observations.

RAND assigns all observations randomly to one of the clusters.

By default, INIT=FORGY.

IMPUTE=MEAN | NONE
specifies the imputation method to be used when the INPUT statement specifies interval variables.

You can choose from the following methods:

MEAN imputes missing values of variables in observations by using the mean value for that variable.

NONE does not do any imputation and excludes observations that have missing values.

If IMPUTE=NONE and the number of observations that have nonmissing values is less than the value of the MAXCLUSTERS= option, then the number of clusters is set to the number of observations that have nonmissing values.

By default, IMPUTE=NONE.

IMPUTENOM=MODE | NONE
specifies the imputation method to be used when the INPUT statement specifies nominal variables.

You can choose from the following methods:

MODE imputes missing values of variables in observations by using the mode value for that variable.

NONE does not do any imputation and excludes observations that have any missing values.

If IMPUTENOM=NONE and the number of observations that have nonmissing values is less than the value of the MAXCLUSTERS= option, then the number of clusters is set to the number of observations that have nonmissing values.

By default, IMPUTENOM=NONE.

MAXCLUSTERS=*number*
specifies either the number of clusters to be used (if the NOC= option is not specified or NOC=NONE) or the maximum number of clusters to be searched (if NOC=ABC).

By default, MAXCLUSTERS=6.

MAXITER=*number*
specifies the maximum number of iterations for the algorithm to perform. In each iteration of the *k*-means or *k*-modes clustering method, each observation is assigned to the nearest cluster centroid, and the centroids are recomputed.

By default, MAXITER=10.

NOC= NONE | ABC < *suboptions* **>**
specifies the method for estimating the number of clusters.

You can specify the following values:

ABC < *suboptions* **>** estimates the number of clusters by using the aligned box criterion (ABC) method. For more information about this method, see the section "Finding the Number of Clusters" on page 839.

You can specify one or more of the following *suboptions*:

ALIGN=*keyword* specifies the method for aligning the reference data based on the input data.

You can specify the following *keywords*:

NONE generates the reference data from a uniform distribution over the range of values for each subset of the input data table.

PCA generates the reference data from a uniform distribution over a box that is aligned with the principal components of each subset of the input data table.

By default, ALIGN=NONE.

B=*number* specifies the number of reference data to be created for each cluster candidate. By default, B=1.

CRITERION=*keyword* specifies the criterion to be used to estimate the number of clusters that use the statistics obtained by the ABC method.

You can specify the following *keywords*:

ALL uses all the following options and selects the number of clusters based on which number of clusters is chosen the most often. If each option selects a different number of clusters, then the number selected by GLOBALPEAK is used.

FIRSTMAXWITHSTD uses the smallest k such that the gap value for that k is greater than the one-standard-error adjusted gap value for k+1.

FIRSTPEAK uses the first peak value among the peak values in gap statistics.

GLOBALPEAK uses the maximum peak value among all the peak values in gap statistics.

By default, CRITERION=GLOBALPEAK.

MINCLUSTERS=*number* specifies the minimum number of clusters for searching for the best number of clusters. By default, MINCLUSTERS=2.

NONE does not estimate the number of clusters and uses the value specified in MAXCLUSTERS= option.

By default, NOC=NONE.

NTHREADS=*number-of-threads*
specifies the number of threads that are used in the computation. The default value is the number of CPUs available in the machine.

OUTSTAT <(OUTITER)>=*libref.data-table*
creates the output data table that contains the cluster centroids for each cluster. *libref.data-table* is a two-level name, where *libref* refers to the library, and *data-table* specifies the name of the output data table. For more information about this two-level name, see the DATA= option and the section "Using CAS Sessions and CAS Engine Librefs" on page 823.

This data table includes the iteration number as _ITERATION_, the cluster ID as _CLUSTER_ID_ and the cluster centroids, which consist of the variables that are specified in the INPUT statement. If you specify STANDARDIZE=RANGE or STANDARDIZE=STD, the data table also contains the standardized values of the cluster centroids, which are displayed by adding the S_ prefix to the variables in the INPUT statement.

You can also specify the following suboption:

OUTITER also outputs the cluster centroids to the OUTSTAT= data table for each iteration.

SEED=*number*
specifies an integer to be used to start the pseudorandom number generator. If you do not specify a seed or if you specify a value less than or equal to 0, the seed is generated from reading the time of day from the computer's clock.

STANDARDIZE=NONE | RANGE | STD
specifies the method for standardizing the interval input variables. PROC KCLUS uses the location and scale measures specific to the method. You can specify the following values:

NONE does not standardize and uses the actual values for the input variables.

RANGE standardizes the input variables by using the range method. PROC KCLUS uses the minimum as the location and the range as the scale.

STD standardizes the input variables by using the traditional standardization method. PROC KCLUS uses the mean as the location and the standard deviation as the scale.

By default, STANDARDIZE=NONE.

STOPCRITERION=*convergence_method***<(VALUE=***number***)>**
specifies the method to use for convergence. If you do not specify this option, the algorithm stops after it reaches the maximum number of iterations (which is specified in MAXITER= option).

You can specify the following *convergence_methods*:

CLUSTER_CHANGE <(VALUE=*number***)>**
uses the percentile of observations that do not change their cluster membership for that iteration. The optional VALUE= suboption enables you to specify the percentile of observations.

By default or if VALUE=0, PROC KCLUS runs until the cluster centroids no longer change.

WCSD_CHANGE < (VALUE=*number*) >
uses the within-cluster distance change as a convergence criterion. The optional VALUE= suboption enables you to specify the change in SSE for the k-means algorithm or specify the sum of within-cluster distances for the k-modes algorithm and the k-prototypes algorithm. For the k-prototypes algorithm, the distance is a mixed distance that includes both the interval and nominal parts.

KPROTOTYPEPARAMS=*gamma_method*
specifies the method for obtaining the γ parameter in the k-prototypes clustering algorithm (Huang 1997) for clustering mixed input data that contain both interval and nominal variables.

You can specify either of the following *gamma_methods*:

USERGAMMA(VALUE=*number*)
specifies the γ parameter value (in the VALUE= suboption) for the k-prototypes clustering algorithm. The value of γ should be an appropriate positive number that balances the difference between the scales of the nominal and interval input variables.

AUTOGAMMA
estimates the γ parameter value for the k-prototypes clustering algorithm. You can use this option when you do not have a clear idea of how large the γ value would be. When you specify the AUTOGAMMA option, the value of γ is inferred to be the average of the standard deviations of the interval input variables.

By default, KPROTOTYPEPARAMS=USERGAMMA(VALUE=0.5).

PRINTALLDISTANCES
prints the distance that is contributed by the interval variables (as _DISTANCEINT_) and the distance that is contributed by the nominal variables (as _DISTANCENOM_) in the mixed distance that is used by the k-prototypes algorithm to the output table that is specified in the SCORE statement. For more information, see the section "Score Output for Clustering Both Interval and Nominal Variables" on page 842.

CODE Statement

CODE < *options* > **;**

The CODE statement generates the SAS DATA step code that mimics the computations that are done by the SCORE statement.

You can specify the following *options*:

FILE=*filename*
specifies the filename of the file to write the SAS score code to.

LABELID=*number*
specifies a number used to construct array names and statement labels in the generated code. You can specify a value in the range 0 to 1024; by default, LABELID=0.

DISPLAY Statement

DISPLAY < *table-list* > < / *options* > **;**

The DISPLAY statement enables you to specify a list of display tables to display or exclude. This statement is similar to the ODS SELECT, ODS EXCLUDE, and ODS TRACE statements. However, the DISPLAY statement can improve performance when a large number of tables could be generated (such as in BY-group processing). The procedure processes the DISPLAY statement on a CAS server and thus sends only a subset of ODS tables to the SAS client. Because ODS statements are processed on a SAS client, first all the generated display tables are sent to the client, and then the client creates a subset.

If you use both DISPLAY and ODS statements together, the DISPLAY statement takes precedence over the ODS statements. Note that the ODS EXCLUDE statement processes tables that are sent to the client after they have been filtered by the DISPLAY statement. In some cases, it might appear that the ODS EXCLUDE statement is taking precedence because it can further filter the tables. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

You can specify the *table-list* as a list of table names, paths, partial pathnames, and regular expressions.

The table names that you can specify are listed in the section "ODS Table Names" on page 847. A path is a table name that is prefixed with dot-separated grouping information. For example, a SelectionSummary table that a procedure produces during a selection routine might have the path *Bygroup1.Summary.SelectionSummary*. A partial pathname does not include all groups; for example, *SelectionSummary* and *Summary.SelectionSummary* are partial pathnames for *Bygroup1.Summary.SelectionSummary*.

When you specify a table name or partial pathname, all display tables whose paths end in the specified name are selected for display or exclusion. For example, both *SelectionSummary* and *Summary.SelectionSummary* select *Bygroup1.Summary.SelectionSummary*.

A regular expression is enclosed in forward slashes (/). For example, specifying "/tions/" selects all pathnames that contain the substring "tions"; in particular, the *Bygroup1.Summary.SelectionSummary* table is selected. Specifying "!/tions/" selects all pathnames that do not contain the substring "tions"; in particular, the *Bygroup1.Summary.SelectionSummary* table is not selected.

You can specify the following *options* after a slash (/):

CASESENSITIVE
performs a case-sensitive comparison of table names in the *table-list* to display table names when tables are subsetted for display. To preserve case, you must enclose table names in the *table-list* in quotation marks.

EXCLUDE
displays all display tables except those that you specify in the *table-list*.

EXCLUDEALL
suppresses display of all tables. This option takes precedence over the other options.

TRACE
displays the display table names, labels, and paths.

DISPLAYOUT Statement

DISPLAYOUT *table-spec-list* < / *options* > **;**

The DISPLAYOUT statement enables you to create output tables from your displayed output. This statement is similar to the ODS OUTPUT statement. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

The *table-spec-list* specifies a list of output tables to create. Each entry in the list has either a *key*=*value* format or a *key* format:

key=*value*	specifies *key* as the ODS table name, path, or partial pathname, and specifies *value* as the output table name.
key	specifies *key* as the ODS table name and also as the output table name.

The ODS table names that you can specify are listed in the section "ODS Table Names" on page 847. You cannot specify the ODS table named OutputCasTables in the *table-spec-list*.

Table names and partial pathnames are discussed under the DISPLAY statement. The DISPLAYOUT statement does not support regular expressions.

You can specify the following *options* after a slash (/):

INCLUDEALL
creates output tables for all display tables. The name of the created output table is the same as the name of the corresponding display table. If you specify this option, the *table-spec-list* specification is ignored.

NOREPLACE
does not replace any existing output table of the same name.

REPEATED
replicates all output tables on all nodes.

FREQ Statement

FREQ *variable* **;**

The *variable* in the FREQ statement identifies a numeric variable in the input data table that contains the frequency of occurrence of each observation. PROC KCLUS treats each observation as if it appears f times, where f is the value of the FREQ *variable* for the observation. If f is not an integer, it is truncated to an integer. If f is less than 1 or missing, the observation is not used in the analysis. When you omit the FREQ statement, each observation is assigned a frequency of 1.

INPUT Statement

INPUT *variables* < **LEVEL= NOMINAL | INTERVAL** > **;**

The INPUT statement specifies the names of the *variables* to be used in clustering. It names one or more input variables that use common options. If you want to use different options for different variables, you can specify multiple INPUT statements.

To trigger the *k*-prototypes clustering algorithm for clustering mixed input data that contain both interval and nominal input variables, either you can use two INPUT statements with one INPUT statement that includes the LEVEL=INTERVAL option for interval input variables and another INPUT statement that includes the LEVEL=NOMINAL option for nominal input variables, or you can use a single INPUT statement without the LEVEL= option specified. In the latter case, PROC KCLUS detects the types of the input variables and uses the *k*-prototypes clustering algorithm if the input variables include both interval and nominal types.

You can include the following option in each INPUT statement:

LEVEL= NOMINAL | INTERVAL

specifies the level of measurement of the variables. For clustering, only interval, binary, and nominal variables are accepted.

By default, LEVEL=INTERVAL for numeric variables, and LEVEL=NOMINAL for categorical character variables.

SAVESTATE Statement

SAVESTATE RSTORE=*libref.data-table* **;**

The SAVESTATE statement creates an analytic store for the model and saves it as a binary object in a data table. You can then use the analytic store in the ASTORE procedure to score new data. The analytic store inherits the same parameters that the procedure uses.

RSTORE=*libref.data-table*

specifies a data table in which to save the analytic store for the model. *libref.data-table* is a two-level name, where *libref* refers to the library, and *data-table* specifies the name of the output data table. For more information about this two-level name, see the DATA= option and the section "Using CAS Sessions and CAS Engine Librefs" on page 823.

SCORE Statement

SCORE OUT=*libref.data-table* < *option* > **;**

OUTPUT OUT=*libref.data-table* < *option* > **;**

The SCORE statement writes the cluster membership information of each observation to the output data table that is specified in the OUT= option. This information includes the variables that are specified in the COPYVARS= option and additional variables, as described in the section "Score Output for Clustering Both Interval and Nominal Variables" on page 842.

You must specify the following option:

OUT=*libref.data-table*

names the output data table for PROC KCLUS to use. You must specify this option before any other options. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to where the data table is to be stored, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 823.

data-table specifies the name of the output data table.

The output data table contains the scored data. When you specify this option, all variables that are specified in the COPYVARS= option, followed by the _CLUSTER_ID_ variable and the _DISTANCE_ variable, are added to the output data table that is specified in this option.

You can also specify the following *option*:

COPYVAR=*variable*

COPYVARS=(*variables***)**

lists one or more variables from the input data table that are transferred to the scored output data table, provided that the output data table produces one or more records per input observation. By default, the KCLUS procedure does not transfer any variables from the input data table to the output data table. The COPYVARS= option accepts numeric and character variables. You can also use COPYVARS=(_ALL_) option to include all the input variables.

WEIGHT Statement

WEIGHT *variable* **;**

The *variable* in the WEIGHT statement is used as a weight to perform a weighted analysis of the data. Observations that have nonpositive or missing weights are not included in the analysis. When you omit the WEIGHT statement, all observations that are used in the analysis are assigned a weight of 1.

Details: KCLUS Procedure

Obtaining the Statistics for Clustering

The KCLUS procedure calculates the following cluster-based statistics (in addition to the variable-based statistics) about the data in the input data table.

- summary statistics for each cluster (for example, the number of observations)
- summary statistics for the variables in each cluster (for example, the mean value of an interval input variable in a cluster)

You can use these statistics in addition to the output data tables to further analyze the clustering results.

Missing Values

Observations that have missing values affect the clustering results because the distance measures that are computed for these observations do not reflect the actual values. To solve this problem, you can use the IMPUTE= option for interval input variables and the IMPUTENOM= option for nominal input variables in the PROC KCLUS statement. If IMPUTE=NONE, observations that have missing values for interval input variables are not considered in cluster analysis. If IMPUTE=MEAN, observations that have missing values for interval input variables are considered by replacing the missing values with the average value for that missing variable, and the mean is calculated based on all nonmissing values of this variable. Similarly, if IMPUTENOM=NONE, observations that have missing values for nominal input variables are not considered in cluster analysis. If IMPUTENOM=MODE, observations that have missing values for nominal input variables are considered by replacing the missing values with the mode value for that missing variable. However, observations that have all missing values are not considered in cluster analysis.

Initial Seed Selection

When INIT=FORGY in the PROC KCLUS statement (the default), random observations from the input data table are selected as initial cluster centroids by default. The initial aim of PROC KCLUS is to select the observations that have no missing values.

You can change the observations that are selected from the input data table by using the SEED= option. You can use this option to set the seed value for the random number generator, which is used for generating random observation indices.

Because the random number generator creates indices for the observations, the initial cluster centroids will be different depending on where and how you run the procedure and where the input data reside. When the data are distributed or the computation is done in parallel, the indices of the observations will change depending on the number of nodes and number of threads that are used. Thus, you might get different initial cluster centroids.

Standardization

Because variables that have large variances tend to affect the distance measure more than variables that have small variances, it is recommended that you standardize the variables before clustering the observations. The STANDARDIZE= option in the PROC KCLUS statement provides a convenient way to standardize the INPUT variables. This option standardizes all the variables by using the method that is specified in that option and adds the standardized values to the output data tables.

Finding the Number of Clusters

To estimate the number of clusters (NOC), you can specify NOC=ABC in the PROC KCLUS statement. This option uses the aligned box criterion (ABC) method to estimate an interim number of clusters and then runs the *k*-means clustering method to produce the final clusters. If the NOC= option is not specified, PROC KCLUS uses the MAXCLUSTERS= option in the PROC KCLUS statement to set the number of the clusters, and then runs the *k*-means clustering method to produce the final clusters.

The ABC method estimates the number of clusters for well-separated clusters. Similar to the gap statistics method that is presented in Tibshirani, Walther, and Hastie (2001), the ABC method uses within-cluster dispersion from the results of clustering as an error measure, making the ABC method independent of the method that is used for clustering. In order to estimate the number of clusters, the ABC method compares the change in the error measure with the change that is expected under an appropriate reference null distribution.

The cubic clustering criterion (CCC), proposed in Sarle (1983), is based on the assumption that a uniform distribution on a hyperrectangle is divided into clusters that are shaped approximately like hypercubes. In recent and widely accepted work, Tibshirani, Walther, and Hastie (2001) propose a gap statistics method that uses Monte Carlo simulation (instead of a heuristic) to generate a hypercube reference distribution. The ABC method improves on the CCC and gap statistics methods by generating more restricted and aligned reference distributions that consider different features in the input data table.

Although the ABC method calculates the reference distribution differently than the other methods do, the computation of the error measure is similar to the method presented in Tibshirani, Walther, and Hastie (2001).

The error measure—also called the gap and represented as *Gap(k)*—is obtained by subtracting the logarithm of the within-cluster sum of squares error from the logarithm of its expectation for clustering solutions over a range of possible k values. The simulation-adjusted standard deviation of the reference distribution—represented as *s(k)*—is obtained by using the standard deviation of the logarithm of the within-cluster sum of squares errors for reference data. The one-standard-error adjusted gap is obtained by subtracting *s(k)* from the gap. For more information, see Tibshirani, Walther, and Hastie (2001).

A notable feature of the ABC algorithm is that it also applies to clustering both interval and nominal inputs. In the k-prototypes clustering algorithm, the ABC algorithm is performed with a mixed distance to which both the interval and nominal inputs contribute.

The number of clusters is estimated by considering the *Gap(k)* and *s(k)* values for each cluster candidate k. You can choose one of four methods for estimating the number of clusters by specifying the CRITERION= suboption in the NOC=ABC option in the PROC KCLUS statement:

- The FIRSTPEAK option selects the first peak value among all the values in *Gap(k)*.
- The GLOBALPEAK option selects the peak value that has the maximum value among the peak values in *Gap(k)*.
- The FIRSTMAXWITHSTD option considers the standard deviation in each cluster in addition to the values in *Gap(k)*. It considers the values in the one-standard-error adjusted gap to estimate the number of clusters. It selects the smallest k such that the gap value for that k is greater than the one-standard-error adjusted gap value for k+1.
- The ALL option uses all the preceding options and selects the number of clusters based on which number of clusters is chosen the most often. If each option selects a different number, then the number selected by the GLOBALPEAK option is used.

Clustering Nominal Variables

The k-means algorithm works only with interval inputs. One way to apply the k-means algorithm to nominal data is to use data transformation methods to transform the nominal data into a new feature space. However, this approach can be very inefficient, and it does not produce good results. For clustering nominal inputs, the KCLUS procedure implements the k-modes clustering algorithm, which extends the k-means algorithm by using different dissimilarity measures and a different method for computing cluster centers (Huang 1997).

Dissimilarity Measures

In the k-modes clustering algorithm, distance measures depend on the level of nominal variables. Let X and Y be two observations, which are described by F nominal variables. Each nominal variable j in F has a different number of distinct values, which are called levels (or sometimes called categories).

The dissimilarity measure between observations X and Y is

$$d(X,Y) = \sum_{j=1}^{F} \delta(x_j, y_j)$$

where $\delta(x_j, y_j)$ is the dissimilarity measure between X and Y on the jth nominal variable.

PROC KCLUS supports the following dissimilarity measures:

- The simple matching dissimilarity measure calculates the total number of mismatches of the corresponding variables of two observations. If the number of mismatches is small, then the observations are similar to each other. The dissimilarity measure between X and Y on the jth nominal variable is

$$\delta(x_j, y_j) = \begin{cases} 0 & \text{if } x_j = y_j \\ 1 & \text{if } x_j \neq y_j \end{cases}$$

  This algorithm is very efficient. However, it might lead to clusters that have weak intrasimilarity, depending on the number of levels and the number of observations that have these levels. For more information, see Huang (1997). Specify DISTANCENOM=BINARY in the PROC KCLUS statement to use this distance measure.

- The frequency-based dissimilarity measure takes into account the frequencies of levels of each variable in the input data table. The dissimilarity measure between X and Y on the jth nominal variable is

$$\delta(x_j, y_j) = \begin{cases} 0 & \text{if } x_j = y_j \\ \frac{(n_{x_j} + n_{y_j})}{n_{x_j} * n_{y_j}} & \text{if } x_j \neq y_j \end{cases}$$

  where n_{x_j} is the number of observations that have level x_j of variable j in the input data table and n_{y_j} is the number of observations that have level y_j of variable j in the input data table. You can use this measure to find underrepresented clusters because it assigns more importance to rare categories than to frequent ones. For more information, see Huang (1997). Specify DISTANCENOM=GLOBALFREQ in the PROC KCLUS statement to use this distance measure.

- The cluster frequency-based dissimilarity measure takes into account the relative frequencies of levels of each variable in each cluster. The dissimilarity measure between X and Y on the jth nominal variable is

$$\delta(x_j, y_j) = \begin{cases} 1 - \frac{n_{x_j^c}}{n_c} & \text{if } x_j = y_j \\ 1 & \text{if } x_j \neq y_j \end{cases}$$

  where n_c is the number of observations in cluster c and $n_{x_j^c}$ is the number of objects that have level x_j of variable j in cluster c.

  This distance measure enables the algorithm to assign less similar observations to the clusters that have weak intrasimilarity. Thus, the generated clusters have strong intrasimilarities. For more information, see Ng et al. (2007). Specify DISTANCENOM=RELATIVEFREQ in the PROC KCLUS statement to use this distance measure.

Computing Cluster Centers

The k-modes algorithm uses modes instead of means as cluster centers. The KCLUS procedure uses a frequency-based method to update the modes after each iteration in order to minimize the clustering cost function. Cluster centers are updated using the same approach for all the distance measures.

Clustering Both Interval and Nominal Variables

For clustering inputs that include only interval or only nominal variables, the KCLUS procedure implements the *k*-means or *k*-modes clustering algorithm, respectively. For clustering inputs that include both interval and nominal variables, the KCLUS procedure implements the *k*-prototypes clustering algorithm (Huang 1997). The *k*-prototypes clustering algorithm uses a mixed distance that combines the distance contributed by the interval inputs and the distance contributed by the nominal inputs. This mixed distance is characterized with a weight parameter, γ, to balance the scales of the interval and nominal distances.

Mixed Distance Measure

The mixed distance measure d_{mixed} is defined as

$$d_{mixed} = d_{int}^2 + \gamma \times d_{nom}$$

where d_{int} is the distance of the interval input variables and d_{nom} is the distance of the nominal input variables. The interval distance measure, d_{int}, is specified in the DISTANCE= option, and the nominal distance measure, d_{nom}, is specified in the DISTANCENOM= option.

Weight of Nominal Distance

A weight parameter, γ, is used to balance any possible difference between the scales of the nominal variables and the interval variables. You can specify this value in the KPROTOTYPEPARAMS= option. By default, the weight is 0.5.

Score Output for Clustering Both Interval and Nominal Variables

The KCLUS procedure writes the cluster membership information to the output table that is specified in the OUT= option in the SCORE statement. This information always includes the variables that are specified in the COPYVARS= option and the following variables:

- _CLUSTER_ID_ (the ID of the closest cluster)
- _DISTANCE_ (the distance between the observation and the centroid of that cluster)

In addition, cluster membership information can include one or more of the following variables:

- _STANDARDIZED_DISTANCE_ (the distance between the standardized values of the observation and the standardized values of cluster centroid). This variable is included if you specify STANDARDIZE=RANGE or STANDARDIZE=STD in the PROC KCLUS statement.
- _DISTANCEINT_ (the interval part of the distance between the observation and the centroid of that cluster). This variable is included if the *k*-prototypes clustering algorithm is used and you specify the PRINTALLDISTANCES option in the PROC KCLUS statement.
- _DISTANCENOM_ (the nominal part of the distance between the observation and the centroid of that cluster). This variable is included if the *k*-prototypes clustering algorithm is used and you specify the PRINTALLDISTANCES option in the PROC KCLUS statement.

- _DISTANCE_, which is equal to _DISTANCEINT_^{2} + $\gamma \times$ _DISTANCENOM_.
- _STANDARDIZED_DISTANCEINT_ (the interval part of the distance between the standardized values of the observation and the standardized values of cluster centroid). This variable is included if the *k*-prototypes clustering algorithm is used and you specify the both the PRINTALLDISTANCES and the STANDARDIZE= options in the PROC KCLUS statement.
- _STANDARDIZED_DISTANCE_, which is equal to _STANDARDIZED_DISTANCEINT_^{2} + $\gamma \times$ _DISTANCENOM_.

Displayed Output

The KCLUS procedure displays various tables for cluster analysis.

The following sections describe the output in the order of their appearance when the related options are specified.

Number of Observations

The "Number of Observations" table displays the number of observations that are read from the input data table and the number of observations that are used in the clustering. The Number of Observations Read column displays the count of raw data that are read from the input data table, and the Number of Observations Used column displays the count of data that are actually used in the clustering. The latter count is affected by the missing values in the raw data and the setting of the IMPUTE= and IMPUTENOM= options in the PROC KCLUS statement. When the FREQ statement is specified, then the Sum of Frequencies Read and Sum of Frequencies Used are also displayed.

Model Information

The "Model Information" table provides basic information about the parameters that are used in the cluster analysis. This information includes the clustering algorithm, maximum number of iterations, stop criterion method and the value for that criterion, number of clusters, initialization technique, seed value, distance method, standardization method, imputation method, and the value of γ when the *k*-prototypes clustering algorithm is used.

Cluster Summary

The "Cluster Summary" table displays the following for each cluster when the variables in INPUT statement are interval:

- cluster number
- frequency (the number of observations in the cluster)
- maximum, minimum, and average distances from the cluster centroid to the observations in the cluster
- SSE (sum of squared errors between the observations and the closest cluster centroids to them)

- standard deviation (the root mean square distances between the observations and the cluster centroids that are closest to them)
- nearest cluster (the ID of the cluster whose centroid is closest to the current cluster's)
- distance between the cluster centroid of the current cluster and the nearest cluster

If the number of observations in a cluster is zero, then this cluster is not displayed in the "Cluster Summary" table.

Iteration History

The "Iteration History" table displays the following when the variables that are specified in the INPUT statement are interval:

- iteration number
- SSE (sum of squared distances between the observations and the closest cluster centroids)
- SSE change
- stop criterion

The "Iteration History" table displays the following when the variables that are specified in the INPUT statement are nominal:

- iteration number
- WithinDist (sum of distances between the observations and the closest cluster centroids)
- within distance change
- stop criterion

Descriptive Statistics

The "Descriptive Statistics" table displays the mean and standard deviation for each interval variable that is specified in the INPUT statement. The calculation of the mean and standard deviation for an interval variable uses all the data in its own column and discards the missing values. So the calculation of mean and standard deviation for one interval variable is independent of the calculation for other interval variables.

Within-Cluster Statistics

The "Within Cluster Statistics" table displays the following for each interval variable that is specified in the INPUT statement in each cluster:

- mean of the values for each variable in that cluster
- standard deviations of the values for each variable in that cluster

Cluster Summary for Nominal Variables

The "Cluster Summary for Nominal Variables" table displays the following for each cluster when the variables that are specified in the INPUT statement are nominal:

- cluster number
- frequency (the number of observations in the cluster)
- maximum, minimum, and average distances from the cluster centroid to the observations in the cluster
- within-cluster distance
- nearest cluster (the ID of the cluster whose centroid is closest to the current cluster's centroid)
- distance between the cluster centroid of the current cluster and the nearest cluster

If the number of observations in a cluster is zero, then this cluster is not displayed in the "Cluster Summary for Nominal Variables" table.

Frequencies for Nominal Variables

The "Frequencies for Nominal Variables" table displays the following for each nominal variable in the INPUT statement in each cluster:

- levels of each variable in the input data table, including any missing levels (if present)
- frequencies of the levels for each variable in the input data table, including frequencies of missing levels (if present)
- frequencies of the levels for each variable within the specific cluster, excluding frequencies of missing levels when the value of the IMPUTENOM= option is NONE

Cluster Summary for Mixed Variables

The "Cluster Summary for Mixed Variables" table displays the following for each cluster when the *k*-prototypes clustering algorithm is performed:

- cluster number
- frequency (the number of observations in the cluster)
- maximum, minimum, and average distances from the cluster centroid to the observations in the cluster
- within-cluster distance
- nearest cluster (the ID of the cluster whose centroid is closest to the current cluster's centroid)
- distance between the cluster centroid of the current cluster and the nearest cluster

If the number of observations in a cluster is 0, then that cluster is not displayed in the "Cluster Summary for Mixed Variables" table.

Standardization

The "Standardization" table displays the following:

- variable name
- location value that is used for standardization
- scale value that is used for standardization

Aligned Box Criterion Parameters

The "ABC Parameters" table displays the following:

- minimum number of clusters that are searched
- maximum number of clusters that are searched
- reference distribution count
- alignment method used for determining the region for generating the reference data

Aligned Box Criterion Statistics

The "ABC Statistics" table displays the following:

- number of clusters (k)
- logarithm of within-cluster sum of squares (W_k^*) for the input data for k number of clusters
- logarithm of within-cluster sum of squares (W_k) for the reference data for k number of clusters
- gap between the error measure from the reference data and the input data
- simulation-adjusted standard deviation (s) that is obtained from the reference data
- one-standard-error adjusted gap

Estimated Number of Clusters

The "Estimated Number of Clusters" table displays the following:

- criterion used for estimating the number of clusters from the measures that are calculated in the specified method
- estimated number of clusters for the input data table

OutputCasTables Table

The OutputCasTables table is a special table that has information about each CAS table that is created during a CAS action execution. The information for each CAS table consists of the CAS table name, the caslib in which the table resides, and the number of columns and rows in the CAS table. Because this table is not a typical ODS table that contains analytical results, you cannot include it in the *table-spec-list* in the DISPLAYOUT statement.

ODS Table Names

Each table created by the KCLUS procedure has a name associated with it. You must use this name to refer to the table when you use ODS statements. These names are listed in Table 16.2.

Table 16.2 ODS Tables Produced by PROC KCLUS

Table Name	Description	Statement	Option
ABCParameters	Parameters that are specified in the NOC= option	PROC KCLUS	NOC=ABC
ABCResults	Estimated number of clusters and the criteria that are used to calculate them	PROC KCLUS	NOC=ABC
ABCStats	Statistics that are computed for estimating the number of clusters	PROC KCLUS	NOC=ABC
ClusterSum	Cluster summary for interval input variables	INPUT	LEVEL=INTERVAL
ClusterSumIntNom	Cluster summary for mixed input variables	INPUT	LEVEL=INTERVAL and LEVEL=NOMINAL
ClusterSumNom	Cluster summary for nominal input variables	INPUT	LEVEL=NOMINAL
DescStats	Descriptive statistics for interval input variables	INPUT	LEVEL=INTERVAL
FreqNom	Frequencies for nominal variables in the input table and within each cluster	INPUT	LEVEL=NOMINAL
IterStats	Iteration history	PROC KCLUS	Default output
ModelInfo	Model information	PROC KCLUS	Default output
NObs	Number of observations	PROC KCLUS	Default output
OutputCasTables	See the section "OutputCasTables Table" on page 847	PROC KCLUS	Default output

Table 16.2 *continued*

Table Name	Description	Statement	Option
Standardization	Information about the standardization method and its parameters	PROC KCLUS	STANDARDIZE=RANGE or STANDARDIZE=STD
WithinClusStats	Statistics for interval input variables within clusters	INPUT	LEVEL=INTERVAL

Examples: KCLUS Procedure

The examples in this chapter are available in the GitHub repository located at `https://github.com/sascommunities/doc-supplement-casstat`.

NOTE: Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections "Using CAS Sessions and CAS Engine Librefs" on page 7 and "Loading a SAS Data Set onto a CAS Server" on page 7 in Chapter 2, "Shared Concepts."

Example 16.1: Cluster Analysis

(View the complete code for this example (kclus_ex1.sas) in the example repository.)

This example uses the Iris data set in the Sashelp library to demonstrate how to use PROC KCLUS to perform cluster analysis. The iris data published by Fisher (1936) have been widely used for examples in discriminant and cluster analyses. The sepal length, sepal width, petal length, and petal width are measured in millimeters on 50 iris specimens from each of three species: *Iris setosa, I. versicolor, and I. virginica*. Mezzich and Solomon (1980) discuss a variety of cluster analyses that use the Iris data.

You can load the Sashelp.Iris data set into your CAS session by naming your libref in the first statement of the following DATA step:

```
data mylib.iris;
   set sashelp.iris;
run;
```

These statements assume that your libref is named mylib, but you can substitute any appropriately defined libref.

The following statements perform clustering:

```
proc kclus
   data=mylib.iris
   seed=12345
   maxclusters=3
   outstat(outiter)=mylib.kclusOutstat1;
```

```
    input SepalLength SepalWidth PetalLength PetalWidth;
    score out=mylib.kclusOut1
    copyvars=(SepalLength SepalWidth PetalLength PetalWidth Species);
run;
```

In this example, PROC KCLUS generates the data table mylib.kclusOut1, which contains the cluster membership information for each observation in the input data table. For each observation, the mylib.kclusOut1 data table includes the variables that are specified in the COPYVARS= option in the SCORE statement and two new variables: _CLUSTER_ID_, which is the ID of the closest cluster, and _DISTANCE_, which is the distance between the observation and the centroid of the closest cluster. This example uses the variables in both the INPUT statement and the COPYVARS= option in order to transfer these variables to the output data table to do further analysis.

PROC KCLUS generates several ODS tables, some of which are shown in Output 16.1.1 through Output 16.1.6.

Output 16.1.1 Number of Observations

The KCLUS Procedure

Number of Observations Read	150
Number of Observations Used	150

Output 16.1.2 Model Information

Model Information	
Clustering Algorithm	K-means
Maximum Iterations	10
Stop Criterion	Cluster Change
Stop Criterion Value	0
Clusters	3
Initialization	Forgy
Seed	12345
Distance for Interval Variables	Euclidean
Standardization	None
Interval Imputation	None

Output 16.1.3 Cluster Summary

Cluster Summary for Interval Variables

		Distance from Cluster Centroid to Observation						
Cluster	**Frequency**	**Minimum**	**Maximum**	**Average**	**SSE**	**Standard Deviation**	**Nearest Cluster**	**Distance to Nearest Cluster Centroid**
1	62	2.1994	16.6064	7.3815	3982.1	8.0142	2	17.9718
2	38	2.5958	15.2971	7.1984	2387.9	7.9272	1	17.9718
3	50	0.6618	12.4803	4.8171	1515.1	5.5047	1	33.5693

Output 16.1.4 Iteration History

Iteration History			
Iteration Number	SSE	SSE Change	Stop Criterion
0	75216		
1	19644	-55572	25.333333
2	9352.340043	-10292	8.000000
3	8148.103759	-1204.236284	4.000000
4	7919.714264	-228.389495	2.000000
5	7885.144143	-34.570121	0

Output 16.1.5 Descriptive Statistics

Descriptive Statistics		
Variable	Mean	Standard Deviation
SepalLength	58.433333	8.280661
SepalWidth	30.573333	4.358663
PetalLength	37.580000	17.652982
PetalWidth	11.993333	7.622377

Output 16.1.6 Within-Cluster Statistics

Within Cluster Statistics			
Variable	Cluster	Mean	Standard Deviation
SepalLength	1	59.0161	4.6641
	2	68.5000	4.9416
	3	50.0600	3.5249
SepalWidth	1	27.4839	2.9628
	2	30.7368	2.9009
	3	34.2800	3.7906
PetalLength	1	43.9355	5.0889
	2	57.4211	4.8859
	3	14.6200	1.7366
PetalWidth	1	14.3387	2.9750
	2	20.7105	2.7987
	3	2.4600	1.0539

The following statements extract the first 10 observations from the output data table; they are shown in Output 16.1.7.

```
proc print noobs data=mylib.kclusOut1(obs=10);
run;
```

Output 16.1.7 First 10 Observations in the Output Data Table

SepalLength	SepalWidth	PetalLength	PetalWidth	Species	_CLUSTER_ID_	_DISTANCE_
50	33	14	2	Setosa	3	1.4959946524
46	34	14	3	Setosa	3	4.1518670499
46	36	10	2	Setosa	3	6.4029680618
51	33	17	5	Setosa	3	3.8259639308
55	35	13	2	Setosa	3	5.2685861481
48	31	16	2	Setosa	3	4.1373904819
52	34	14	2	Setosa	3	2.1066561181
49	36	14	1	Setosa	3	2.5686572368
44	32	13	2	Setosa	3	6.6901420015
50	35	16	6	Setosa	3	3.8675573687

PROC KCLUS creates the output statistics data table, which contains the cluster centroids. This data table includes the iteration number as _ITERATION_, the cluster ID as _CLUSTER_ID_, and the cluster centroids, which consist of the variables that are specified in the INPUT statement. Because the OUTITER= suboption is included in the OUTSTAT= option in the PROC KCLUS statement, cluster centroids for each iteration are added to the kclusOutstat1 data table.

The following statements extract the centroids before the first iteration and after the last iteration:

```
proc print noobs data=mylib.kclusOutstat1(firstobs=1 obs=3);
run;

proc print noobs data=mylib.kclusOutstat1(firstobs=16 obs=18);
run;
```

Output 16.1.8 and Output 16.1.9 show the results.

Output 16.1.8 Cluster Centroids before the First Iteration

ITERATION	_CLUSTER_ID_	SepalLength	SepalWidth	PetalLength	PetalWidth
0	1	63	25	49	15
0	2	63	33	60	25
0	3	61	28	47	12

Output 16.1.9 Cluster Centroids after the Last Iteration

ITERATION	_CLUSTER_ID_	SepalLength	SepalWidth	PetalLength	PetalWidth
5	1	59.0161	27.4839	43.9355	14.3387
5	2	68.5000	30.7368	57.4211	20.7105
5	3	50.0600	34.2800	14.6200	2.4600

Example 16.2: Finding the Number of Clusters

(View the complete code for this example (kclus_ex2.sas) in the example repository.)

This example uses the same data table that is loaded into your CAS session in Example 16.1.

You can find the number of clusters in the data table by specifying NOC=ABC in the PROC KCLUS statement as follows:

```
data mylib.iris;
   set sashelp.iris;
run;

proc kclus data=mylib.iris maxclusters=9 seed=1234
           NOC=ABC(B=10 minclusters=2 align=PCA criterion=FIRSTPEAK);
   input SepalLength SepalWidth PetalLength PetalWidth;
   ods output ABCStats=ABCStats1;
run;
```

PROC KCLUS generates several ODS tables, some of which are shown in Output 16.2.1 through Output 16.2.4.

Output 16.2.1 shows the parameters that are used in the aligned box criterion (ABC) method.

Output 16.2.1 Aligned Box Criterion Parameters

The KCLUS Procedure

ABC Parameters

Minimum Cluster	Maximum Cluster	Reference Distribution Count	Alignment Method
2	9	10	PCA

Output 16.2.2 shows the statistics that are obtained for each candidate number of clusters.

Output 16.2.2 Aligned Box Criterion Statistics

ABC Statistics

	Logarithm of Within-Cluster SSE				
Number of Clusters	Input	Reference	Gap	Simulation Adjusted Standard Deviation	One Standard Error Adjusted Gap
2	9.6313	10.1237	0.4923	0.0531	0.4392
3	8.9727	9.4894	0.5167	0.3678	0.1489
4	8.6811	8.9211	0.2400	0.0408	0.1992
5	8.5528	8.8012	0.2485	0.0536	0.1948
6	8.2784	8.4730	0.1945	0.0587	0.1359
7	8.2133	8.3702	0.1569	0.0517	0.1053
8	8.0750	8.5884	0.5134	0.0419	0.4715
9	8.0444	8.1240	0.0796	0.0639	0.0158

Output 16.2.3 Estimated Number of Clusters

Estimated Number of Clusters	
Criterion	Number of Clusters
FirstPeak	3

Output 16.2.4 Cluster Summary Table for Each Cluster

Cluster Summary for Interval Variables								
		Distance from Cluster Centroid to Observation						
Cluster	Frequency	Minimum	Maximum	Average	SSE	Standard Deviation	Nearest Cluster	Distance to Nearest Cluster Centroid
1	62	2.1994	16.6064	7.3815	3982.1	8.0142	2	17.9718
2	38	2.5958	15.2971	7.1984	2387.9	7.9272	1	17.9718
3	50	0.6618	12.4803	4.8171	1515.1	5.5047	1	33.5693

When you use the NOC= option, the KCLUS procedure first estimates the number of clusters, *k*, and then it displays the cluster analysis results for each of the *k* clusters as shown in Output 16.2.4.

Example 16.3: Clustering Nominal Variables

(View the complete code for this example (kclus_ex3.sas) in the example repository.)

In this example, PROC KCLUS clusters nominal variables in the Baseball data set. The Baseball data set includes 322 observations, and each observation has 24 variables. Among these 24 variables, the 5 nominal ones are selected as the input data to show an example of running *k*-modes clustering on a nominal data set. You can load the Sashelp.Baseball data set into your CAS session by naming your libref in the first statement of the following DATA step.

```
data mylib.baseball;
  Set sashelp.baseball;
  Keep Team League Division Position Div;
Run;
```

These statements assume that your libref is named mylib, but you can substitute any appropriately defined libref.

The following statements run the *k*-modes clustering algorithm with a frequency-based distance measure (DISTANCENOM=RELATIVEFREQ) and verify whether the clusters that the procedure obtains match the labels of the observations in the data table:

```
proc kclus data=mylib.baseball maxiter=10 maxc=5 DISTANCENOM=RELATIVEFREQ
   outstat(outiter)=mylib.kclusOutstat2;
input Team League Division Position Div / level=nominal;
```

```
score out=mylib.kclusOut2 copyvars=(Team League Division Position Div);
ods output FreqNom=FreqNom1;
run;
```

Output 16.3.1 shows the cluster summary table that is produced for five clusters.

Output 16.3.1 Cluster Summary Table for Five Clusters

The KCLUS Procedure

Cluster Summary for Nominal Variables

Cluster	Frequency	Distance from Cluster Centroid to Observation			Within Cluster Distance	Nearest Cluster	Distance to Nearest Cluster Centroid
		Minimum	Maximum	Average			
1	90	1.7111	2.0000	1.9580	176.2	3	3.8824
2	85	1.7059	2.0000	1.9550	166.2	1	3.8667
3	72	1.6806	2.0000	1.9466	140.2	3	3.8824
4	75	1.6800	2.0000	1.9474	146.1	1	4.0000

Output 16.3.2 shows the frequencies of levels for the nominal input variable Team and information about how the levels of variables are distributed in each cluster; this information is important for revealing intracluster similarity. The following statement prints the observations from the frequency table, as shown in Output 16.3.2:

```
proc print noobs data=FreqNom1(obs=12);
run;
```

Output 16.3.2 Frequencies for Nominal Variables

Variable	Level	FrequencyRead	_1	_2	_3	_4
Team	Atlanta	11	0	0	0	11
Team	Baltimore	15	0	15	0	0
Team	Boston	10	0	10	0	0
Team	California	13	13	0	0	0
Team	Chicago	24	13	0	11	0
Team	Cincinnati	12	0	0	0	12
Team	Cleveland	12	0	12	0	0
Team	Detroit	12	0	12	0	0
Team	Houston	11	0	0	0	11
Team	Kansas City	14	14	0	0	0
Team	Los Angeles	14	0	0	0	14
Team	Milwaukee	14	0	14	0	0

Example 16.4: Clustering Mixed Variables

(View the complete code for this example (kclus_ex4.sas) in the example repository.)

In this example, PROC KCLUS uses the *k*-prototypes clustering algorithm to cluster mixed input data that contain both interval and nominal variables in the Baseball data set, which is the same data set that is used in Example 16.3. You can execute the following SAS code to load the input data table, mylib.baseball.

```
data mylib.baseball;
  Set sashelp.baseball;
  Keep CrAtBat CrHits CrRuns CrRbi CrBB Team League Division Position Div;
Run;
```

These statements assume that your libref is named mylib, but you can substitute any appropriately defined libref.

The following statements run the *k*-prototypes clustering algorithm on mixed input data. The first INPUT statement uses the LEVEL=INTERVAL option to specify the input variables CrAtBat, CrHits, CrRuns, CrRbi, and CrBB as the interval input variables; the second INPUT statement uses the LEVEL=NOMINAL option to specify the input variables Team, League, Division, Position, and Div as the nominal input variables.

The KPROTOTYPEPARAMS=USERGAMMA(VALUE=10) option specifies $\gamma = 10$ for the *k*-prototypes clustering algorithm. The NOC=ABC option uses the ABC algorithm to search for the best number of clusters in a range of 2 to 10. The PRINTALLDISTANCES option prints two additional variables, _DISTANCEINT_ and _DISTANCENOM_, in the output table kclusout4.

```
proc kclus data=mylib.baseball maxiter=10 maxc=10 distancenom=relativefreq
   outstat(outiter)=mylib.kclusOutstat4 printalldistances
   noc=abc(B=10 minclusters=2 align=none criterion=all)
   kprototypeparams=usergamma(value=10);
   input CrAtBat CrHits CrRuns CrRbi CrBB / level=interval;
   input Team League Division Position Div / level=nominal;
   score out=mylib.kclusOut4 copyvars=
   (CrAtBat CrHits CrRuns CrRbi CrBB Team League Division Position Div);
   ods output FreqNom=FreqNom4;
run;
```

Output 16.4.1 shows the "Model Information" table. The clustering algorithm in use is the *k*-prototypes, and the γ value is 10. Since PROC KCLUS is implementing the *k*-prototypes algorithm on mixed input data, the distance measures for both the interval and nominal variables are displayed in the table, as Euclidean and RelativeFreq, respectively.

Output 16.4.1 Model Information for k-Prototypes Clustering Algorithm

The KCLUS Procedure

Model Information	
Clustering Algorithm	K-prototypes
Gamma Value	10
Maximum Iterations	10
Stop Criterion	Cluster Change
Stop Criterion Value	0
Clusters	2
Initialization	Forgy
Seed	1065607171
Distance for Interval Variables	Euclidean
Distance for Nominal Variables	RelativeFreq
Number of Clusters Estimation	ABC
Standardization	None
Interval Imputation	None
Nominal Imputation	None

Output 16.4.2 shows the cluster summary table that is produced for the k-prototypes algorithm.

Output 16.4.2 Cluster Summary Table for k-Prototypes Clustering Algorithm

Cluster Summary for Mixed Variables

		Distance from Cluster Centroid to Observation					
Cluster	**Frequency**	**Minimum**	**Maximum**	**Average**	**Within Cluster Distance**	**Nearest Cluster**	**Distance to Nearest Cluster Centroid**
1	94	16703.3	77673932	3377958	3.1753E8	2	21203061
2	228	4081.1	5148754	1081023	2.4647E8	1	21203059

References

Fisher, R. A. (1936). "The Use of Multiple Measurements in Taxonomic Problems." *Annals of Eugenics* 7:179–188.

Huang, Z. (1997). "A Fast Clustering Algorithm to Cluster Very Large Categorical Data Sets in Data Mining." In *Proceedings of the SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery*, 1–8. New York: ACM Press.

Mezzich, J. E., and Solomon, H. (1980). *Taxonomy and Behavioral Science*. New York: Academic Press.

Ng, M. K., Li, M. J., Huang, J. Z., and He, Z. (2007). "On the Impact of Dissimilarity Measure in k-Modes Clustering Algorithm." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29:503–507.

Sarle, W. S. (1983). *Cubic Clustering Criterion*. Technical Report A-108, SAS Institute Inc., Cary, NC.

Tibshirani, R., Walther, G., and Hastie, T. (2001). "Estimating the Number of Clusters in a Dataset via the Gap Statistic." *Journal of the Royal Statistical Society, Series B* 63:411–423.

Chapter 17
LMIXED Procedure

Contents

Overview: LMIXED Procedure **860**
PROC LMIXED Features 861
Notation for the Mixed Model 863
PROC LMIXED Compared to Other SAS Procedures 864
Using CAS Sessions and CAS Engine Librefs 864
Getting Started: LMIXED Procedure **865**
Syntax: LMIXED Procedure **868**
PROC LMIXED Statement 870
BLUP Statement 875
BY Statement 876
CLASS Statement 876
CONTRAST Statement 877
DISPLAY Statement 880
DISPLAYOUT Statement 881
EFFECT Statement 882
ESTIMATE Statement 883
FREQ Statement 886
LSMEANS Statement 887
LSMESTIMATE Statement 894
MARGINS Statement 899
MODEL Statement 905
OPTIMIZATION Statement 907
OUTPUT Statement 911
PARMS Statement 913
RANDOM Statement 916
REPEATED Statement 925
SLICE Statement 927
STORE Statement 933
TEST Statement 934
WEIGHT Statement 934
Details: LMIXED Procedure **935**
Linear Mixed Models Theory 935
Matrix Notation for General Linear Model 935
Formulation of the Mixed Model 936
Estimating Covariance Parameters in the Mixed Model 939
Computing Methodologies for REML in the Mixed Model 940

Estimating Fixed and Random Effects in the Mixed Model 941
Statistical Properties 942
Common Subject Effect 943
Computational Method 944
Distributed Computing 944
Multithreading 944
Displayed Output 944
Model Information 944
Class Level Information 944
Dimensions 944
Number of Observations 945
Optimization Information 945
Descriptive Statistics 945
Iteration History 945
Convergence Status 946
Covariance Parameter Estimates 946
Fit Statistics 946
Timing Information 947
OutputCasTables Table 947
ODS Table Names 947
Examples: LMIXED Procedure **950**
Example 17.1: Mixed Model Analysis of Microarray Data 950
References **954**

Overview: LMIXED Procedure

The LMIXED procedure fits a variety of linear mixed models to data and enables you to use these fitted models to make statistical inferences about the data. A *linear mixed model* is a generalization of the standard linear model that is used in the GLM procedure in SAS/STAT software; the generalization is that the data are permitted to exhibit correlation and nonconstant variability. Therefore, the linear mixed model provides you with the flexibility of modeling not only the means of your data (as in the standard linear model) but also their variances and covariances.

The primary assumptions underlying the analyses that the LMIXED procedure performs are as follows:

- The data are normally distributed (Gaussian).
- The means (expected values) of the data are linear in terms of a certain set of parameters.
- The variances and covariances of the data are conditioned on a different set of parameters, with a structure that the LMIXED procedure supports.

Because Gaussian data can be modeled entirely in terms of their means, variances and covariances, the two sets of parameters in a linear mixed model specify the complete probability distribution of the data. The parameters of the mean model are called *fixed-effects parameters*, and the parameters of the variance-covariance model are called *covariance parameters*.

The fixed-effects parameters are associated with known explanatory variables, as in the standard linear model. These variables can be either qualitative (as in the traditional analysis of variance) or quantitative (as in standard linear regression).

The covariance parameters are what distinguishes the linear mixed model from the standard linear model. The need for covariance parameters arises quite frequently in applications. A typical application follows a scenario in which the experimental units that the data are measured on can be grouped into clusters and the data from a common cluster are correlated. This scenario can be generalized to include one set of clusters nested within another. For example, if students are the experimental unit, they can be clustered into classes, which in turn can be clustered into schools. Each level of this hierarchy can introduce an additional source of variability and correlation.

The LMIXED procedure provides a variety of covariance structures. The most common covariance structures arise from the use of *random effects*, which are additional unknown random variables that are assumed to affect the variability of the data. The variances of the random effects, commonly known as *variance components*, become the covariance parameters for this particular structure. Traditional linear mixed models contain both fixed and random effects; in fact, it is the combination of these two types of effects that led to the name *mixed model*. The LMIXED procedure fits not only these traditional variance component models but also numerous other covariance structures.

The LMIXED procedure fits the structure you select by using the method of *restricted maximum likelihood (REML)*, also known as *residual maximum likelihood*. It is here that the Gaussian assumption for the data is exploited. Other estimation methods are also available, including *maximum likelihood (ML)* and *minimum variance quadratic unbiased estimation (MIVQUE0)*. The details of these estimation methods are discussed in subsequent sections.

PROC LMIXED Features

The LMIXED procedure provides easy accessibility to numerous linear mixed models that are useful in many common statistical analyses.

Here are the main features of the LMIXED procedure:

- The RANDOM statement supports many covariance structures, including variance components, compound symmetry, unstructured, AR(1), Toeplitz, factor analytic, and so on.
- Both the MODEL statement and the RANDOM statement are supported for model specification, as in the MIXED procedure.
- The REPEATED statement supports many residual covariance structures, including variance components, compound symmetry, unstructured, AR(1), and so on.
- Inference features include standard errors and *t* tests for fixed and random effects.
- A subject effect for blocking is supported.

- Both REML and ML estimation methods are supported; they are implemented with a variety of optimization algorithms.
- It handles unbalanced data.
- Specialized dense and sparse matrix algorithms are provided.
- The CONTRAST statement provides a mechanism for obtaining custom linear hypothesis tests that involve the fixed and/or random effects.
- The ESTIMATE statement provides a mechanism for obtaining custom linear estimates of the fixed and/or random effects.
- The FREQ statement enables you to input the frequency of occurrence for each observation as if it appears multiple times in the data set.
- The LSMEANS statement computes least squares means for classification fixed effects.
- The LSMESTIMATE statement computes least squares means estimates for classification fixed effects.
- The MARGINS statement computes predictive margins of fixed effects.
- The OUTPUT statement produces output data tables that contain predicted values, residuals, studentized residuals, confidence limits, and influence statistics.
- The PARMS statement enables you to fit a linear mixed model that has known covariance values or to set boundary values for the parameters.
- The SLICE statement performs a partitioned analysis of the LS-means for an interaction.
- The TEST statement performs Type III tests of the fixed effects in the model.

Because the LMIXED procedure runs on SAS Cloud Analytic Services (CAS; see the section "Using CAS Sessions and CAS Engine Librefs" on page 864), it also does the following:

- enables you to run on a cluster of machines that distribute the data and the computations
- enables you to run in single-machine mode on CAS
- exploits all the available cores and concurrent threads. For information about how the LMIXED procedure uses threads, see the section "Multithreading" on page 102 in Chapter 2, "Shared Concepts."

Notation for the Mixed Model

This section introduces the mathematical notation that is used throughout this chapter to describe the linear mixed model. It assumes familiarity with basic matrix algebra (for an overview, see Searle 1982). A more detailed description of the mixed model is contained in the section "Linear Mixed Models Theory" on page 935.

A statistical model is a mathematical description of how data are generated. The standard linear model, as used by the GLM procedure, is one of the most common statistical models:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

In this expression, $\mathbf{y}$ represents a vector of observed data, $\boldsymbol{\beta}$ is an unknown vector of fixed-effects parameters with a known design matrix $\mathbf{X}$, and $\boldsymbol{\epsilon}$ is an unknown random error vector that models the statistical noise around $\mathbf{X}\boldsymbol{\beta}$. The focus of the standard linear model is to model the mean of $\mathbf{y}$ by using the fixed-effects parameters $\boldsymbol{\beta}$. The residual errors $\boldsymbol{\epsilon}$ are assumed to be independent and identically distributed Gaussian random variables with mean 0 and variance σ^2.

The mixed model generalizes the standard linear model as follows:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\boldsymbol{\gamma} + \boldsymbol{\epsilon}$$

Here, $\boldsymbol{\gamma}$ is an unknown vector of random-effects parameters with a known design matrix $\mathbf{Z}$, and $\boldsymbol{\epsilon}$ is an unknown random-error vector whose elements are no longer required to be independent and homogeneous.

The number of fixed-effects parameters is called the number of levels for fixed effects throughout this chapter. The number of random-effects parameters is called the number of levels for random effects throughout this chapter.

To further develop this notion of variance modeling, assume that $\boldsymbol{\gamma}$ and $\boldsymbol{\epsilon}$ are Gaussian random variables that are uncorrelated, have expectations $\mathbf{0}$, and have variances $\mathbf{G}$ and $\mathbf{R}$, respectively. The variance of $\mathbf{y}$ is thus

$$\mathbf{V} = \mathbf{Z}\mathbf{G}\mathbf{Z}' + \mathbf{R}$$

Note that when $\mathbf{Z} = \mathbf{0}$, the mixed model reduces to the standard linear model.

You can model the variance of the data $\mathbf{y}$ by specifying the structures of $\mathbf{Z}$, $\mathbf{G}$, and $\mathbf{R}$. The model matrix $\mathbf{Z}$ is set up in the same fashion as $\mathbf{X}$, the model matrix for the fixed-effects parameters. For $\mathbf{G}$ and $\mathbf{R}$, you must select some covariance structures. Possible covariance structures include the following:

- variance components
- compound symmetry (common covariance plus diagonal)
- unstructured (general covariance)
- autoregressive

By appropriately defining the model matrices $\mathbf{X}$ and $\mathbf{Z}$ in addition to the covariance structure matrices $\mathbf{G}$ and $\mathbf{R}$, you can perform numerous mixed model analyses.

PROC LMIXED Compared to Other SAS Procedures

The LMIXED procedure provides linear mixed modeling functionality that is most comparable to that of the MIXED, HPMIXED, and HPLMIXED procedures in SAS/STAT software. The GLIMMIX procedure and the NLMIXED procedure can also be used to fit linear mixed models, but these procedures are primarily designed to fit generalized linear mixed models (GLMMs) and nonlinear mixed models, respectively.

The LMIXED procedure, like PROC HPMIXED, can fit linear mixed models by using sparse matrix storage and sparse matrix computations. These sparse matrix techniques are particularly suitable for problems in which the $[\mathbf{XZ}]'[\mathbf{XZ}]$ crossproducts matrix is large and sparse.

The LMIXED procedure, like PROC HPLMIXED, can fit linear mixed models by using specialized efficient algorithms. This is particularly suitable for models that have thousands of levels in a common subject effect.

The LMIXED procedure is designed to fit the following large linear mixed models:

- models that have thousands of levels in the common subject effect. For information about the common subject effect, see "Common Subject Effect" on page 943.
- models that have a large and sparse design matrix.

Using CAS Sessions and CAS Engine Librefs

SAS Cloud Analytic Services (CAS) is the analytic server and associated cloud services in SAS Viya. This section describes how to create a CAS session and set up a CAS engine libref that you can use to connect to the CAS session. It assumes that you have a CAS server already available; contact your system administrator if you need help starting and terminating a server. This CAS server is identified by specifying the host on which it runs and the port on which it listens for communications. To simplify your interactions with this CAS server, the host information and port information for the server are stored as SAS option values that are retrieved automatically whenever this CAS server needs to be accessed. You can examine the host and port values for the server at your site by using the following statements:

```
proc options option=(CASHOST CASPORT);
run;
```

In addition to starting a CAS server, your system administrator might also have created a CAS session and a CAS engine libref for your use. You can define your own sessions and CAS engine librefs that connect to the CAS server as shown in the following statements:

```
cas mysess;
libname mylib cas sessref=mysess;
```

The CAS statement creates the CAS session named mysess, and the LIBNAME statement creates the mylib CAS engine libref that you use to connect to this session. It is not necessary to explicitly name the CASHOST and CASPORT of the CAS server in the CAS statement, because these values are retrieved from the corresponding SAS option values.

If you have created the mysess session, you can terminate it by using the TERMINATE option in the CAS statement as follows:

```
cas mysess terminate;
```

For more information about the CAS and LIBNAME statements, see the section "Introduction to Shared Concepts" on page 6 in Chapter 2, "Shared Concepts."

Getting Started: LMIXED Procedure

(View the complete code for this example at https://github.com/sascommunities/doc-supplement-casstat/tree/main/Examples/lmixedgs.sas.)

This example demonstrates how you can perform a mixed model analysis of covariance when you have many groups. Suppose you are an educational researcher who studies how student scores on math tests change over time. Students are tested four times, and you want to estimate the overall rise or fall, accounting for correlation between test response behaviors of students in the same neighborhood and school. One way to model this correlation is by using a random-effects analysis of covariance, where in addition to having a student-specific quadratic mean response function, the scores for students from the same neighborhood and school are all assumed to share the same quadratic mean test response function, whose parameters are random. The following statements simulate data that have this structure; these statements assume that your libref is named mylib, but you can substitute any appropriately defined libref.

NOTE: Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections "Using CAS Sessions and CAS Engine Librefs" on page 7 and "Loading a SAS Data Set onto a CAS Server" on page 7 in Chapter 2, "Shared Concepts."

```
data mylib.SchoolSample;
   do SchoolID = 1 to 1000;
      do nID = 1 to 50;
         Neighborhood = (SchoolID-1)*15 + nId;
         bInt   = 15*ranuni(1);
         bTime  = 15*ranuni(1);
         bTime2 =   ranuni(1);
         do sID = 1 to 2;
            do Time = 1 to 4;
               Math = bInt + bTime*Time + bTime2*Time*Time + rannor(2);
               output;
               end;
            end;
         end;
      end;
run;
```

These data comes from 1,000 schools and about 15,035 neighborhoods; neighborhoods are associated with more than one school and vice versa. There are 400,000 observations and 50,000 levels of the subject effect.

The following statements use the LMIXED procedure to fit a linear mixed model to these data and produce Figure 17.1 through Figure 17.4:

```
proc lmixed data=mylib.SchoolSample;
   class Neighborhood SchoolID;
   model Math = Time Time*Time / solution;
   random   int Time Time*Time / sub=Neighborhood(SchoolID) type=un;
run;
```

This model fits a quadratic mean response model that has an unstructured covariance matrix to model the covariance between the random parameters of the response model. With 15,035 neighborhood/school combinations, this model can be computationally daunting to fit, but the LMIXED procedure finishes quickly by multithreaded computing.

Figure 17.1 displays the "Model Information," "Optimization Information," "Class Level Information," "Number of Observations Information," and "Dimensions" tables.

Figure 17.1 Mixed Model Analysis of Covariance

The LMIXED Procedure

Model Information	
Data Source	SCHOOLSAMPLE
Response Variable	Math
Estimation Method	Restricted Maximum Likelihood (REML)
Degrees of Freedom Method	Residual
Design Matrix Method	Dense

Optimization Information	
Optimization Technique	Newton-Raphson with Ridging
Hessian in Optimization	Exact
Parameters in Optimization	6
Lower Boundaries	3
Upper Boundaries	0
Residual Variance	Profiled
Starting Values From	Data

Class Level Information		
Class	**Levels**	**Values**
Neighborhood	15035	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 ...
SchoolID	1000	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 ...

Number of Observations Read	400000
Number of Observations Used	400000

Dimensions	
G-side Covariance Parameters	6
R-side Covariance Parameters	1
Columns in X	3
Columns in Z per Subject	3
Subjects (Blocks in V)	50000

Figure 17.2 displays the convergence status and the "Covariance Parameter Estimates" table.

Figure 17.2 Mixed Model Analysis of Covariance

Convergence criterion (ABSGCONV=0.00001) satisfied.

Covariance Parameter Estimates

Cov Parm	Subject	Estimate
UN(1,1)	Neighborhood(SchoolID)	18.7964
UN(2,1)	Neighborhood(SchoolID)	-0.1215
UN(2,2)	Neighborhood(SchoolID)	18.9874
UN(3,1)	Neighborhood(SchoolID)	-0.00044
UN(3,2)	Neighborhood(SchoolID)	-0.00805
UN(3,3)	Neighborhood(SchoolID)	0.08394
Residual		0.9998

Figure 17.3 displays the "Fit Statistics" table.

Figure 17.3 Mixed Model Analysis of Covariance

Fit Statistics	
-2 Res Log Likelihood	1681870
AIC (smaller is better)	1681884
AICC (smaller is better)	1681884
BIC (smaller is better)	1681945
CAIC (smaller is better)	1681952
HQIC (smaller is better)	1681903

Figure 17.4 displays the "Solution for Fixed Effects" table.

Figure 17.4 Mixed Model Analysis of Covariance

Solution for Fixed Effects

Effect	Estimate	Standard Error	DF	t Value	Pr > |t|
Intercept	7.4961	0.02129	4E5	352.04	<.0001
Time	7.4939	0.02108	4E5	355.55	<.0001
Time*Time	0.5009	0.002044	4E5	245.03	<.0001

These estimates are very close to the quadratic coefficients that are used in the DATA step multiplied by 1/2, which is the expected value of a uniform random variable on [0,1].

Syntax: LMIXED Procedure

The following statements are available in the LMIXED procedure:

PROC LMIXED < *options* > ;
BLUP OUT= *libref.data-table* < *options* > ;
BY *variables* ;
CLASS *variable* < **(** *options* **)** >. . . < *variable* < **(** *options* **)** > > < / *global-options* > ;
CONTRAST *'label' contrast-specification* < **,** *contrast-specification* > < **,** . . . > < / *options* > ;
DISPLAY < *table-list* > < / *options* > ;
DISPLAYOUT *table-spec-list* < / *options* > ;
EFFECT *name* **=** *effect-type* **(** *variables* < / *options* > **)** ;
ESTIMATE *'label' contrast-specification* < **(** *divisor* **=** *n* **)** >
< **,** *'label' contrast-specification* < **(** *divisor* **=** *n* **)** > > < **,** . . . > < / *options* > ;
FREQ *variable* ;
LSMEANS *fixed-effects* < / *options* > ;
LSMESTIMATE *fixed-effect* < *'label'* > *values* < *divisor* **=** *n* > < **,** < *'label'* > *values* < *divisor* **=** *n* > > < **,**
. . . > < / *options* > ;
MARGINS *fixed-effects* < / *options* > ;
MODEL *dependent* **=** < *fixed-effects* > < / *options* > ;
OPTIMIZATION < *options* > ;
OUTPUT OUT= *libref.data-table*
< **COPYVARS=** *(variables)* >
< *keyword* < *=name* > >. . . < *keyword* < *=name* > > ;
PARMS < **(** *value-list* **)** . . . > < / *options* > ;
RANDOM *random-effects* < */options* > ;
REPEATED *repeated-effect* < */options* > ;
SLICE *model-effect* < */options* > ;
STORE < **OUT=** > *libref.data-table* ;
TEST *fixed-effects* < */options* > ;
WEIGHT *variable* ;

Items within angle brackets (< >) are optional.

The PROC LMIXED and MODEL statements are required, and the MODEL statement must appear after the CLASS statement if a CLASS statement is included. The RANDOM statement must follow the MODEL statement. The CLASS, CONTRAST, EFFECT, ESTIMATE, LSMEANS, LSMESTIMATE, MARGINS, RANDOM, SLICE, and TEST statements can appear multiple times. Other statements can appear only once.

The BLUP, BY, CONTRAST, ESTIMATE, LSMEANS, LSMESTIMATE, MARGINS, MODEL, OPTIMIZATION, OUTPUT, PARMS, RANDOM, REPEATED, SLICE, TEST, and WEIGHT statements are described in full in alphabetical order after the PROC LMIXED statement. The CLASS and EFFECT statements are described by summarizing their functionality and syntax; for more information about these statements, see Chapter 2, "Shared Concepts."

Table 17.1 summarizes the basic functions and important options and statements that are supported by the LMIXED procedure.

Table 17.1 Summary of PROC LMIXED Statements

Statement	Description	Important Options
PROC LMIXED	Invokes the procedure	The DATA= option specifies the input data table; the METHOD= option specifies the estimation method.
BLUP	Creates a data table that contains the estimation of fixed and random effects	The OUT= option specifies the output data table; the SOLVER= option specifies the mixed model equations solver.
CLASS	Declares qualitative variables that create indicator variables in **X** and **Z** matrices	None
CONTRAST	Constructs custom linear hypothesis tests	The E option displays the **L** matrix coefficients.
ESTIMATE	Constructs custom linear estimates	The CL option produces confidence limits.
LSMEANS	Computes least squares means for classification fixed effects	The DIFF option computes differences of the least squares means; the CL option produces confidence limits; the SLICE= option tests simple effects.
LSMESTIMATE	Computes least squares means estimates for classification fixed effects	The ADJUST= option determines the method of multiple comparison adjustment; the FTEST option produces an *F* test.
MARGINS	Computes predictive margins of fixed effects	The DIFF= option computes differences of predictive margins; the SLICEBY= option computes predictive margins by slices.
MODEL	Specifies the dependent variable and the fixed effects, setting up **X**	The S option requests a solution for fixed-effects parameters.
OPTIMIZATION	Controls aspects of the nonlinear optimizations	The TECHNIQUE= option specifies the optimization technique.
OUTPUT	Creates a data table that contains observationwise statistics	The ALLSTATS option requests that all statistics be computed.
PARMS	Specifies a grid of initial values for the covariance parameters	The HOLD= and NOITER options hold the covariance parameters or their ratios constant; the PARMSDATA= option reads the initial values from a data table.
RANDOM	Specifies random effects, setting up **Z** and **G**	The SUBJECT= option creates block-diagonality; the TYPE= option specifies the covariance structure; the S option requests a solution for the random effects.
REPEATED	Specifies **R**	The SUBJECT= option creates block-diagonality; the TYPE= option specifies covariance structure; the R= option displays estimated blocks of **R**; the GROUP= option enables between-subject heterogeneity.
SLICE	Performs a partitioned analysis of the LS-means for an interaction	The SLICEBY option provides a simple effect to partition the LS-means.

Table 17.1 *continued*

Statement	Description	Important Options
STORE	Saves the model and results of the statistical analysis	The OUT= option specifies the name of the table where the model and results are stored.
TEST	Performs hypothesis tests on the fixed effects	The CHISQ option performs χ^2 tests in addition to any F tests.

PROC LMIXED Statement

PROC LMIXED < *options* > **;**

The PROC LMIXED statement invokes the procedure. Table 17.2 summarizes important options in the PROC LMIXED statement by function. These and other options in the PROC LMIXED statement are then described fully in alphabetical order.

Table 17.2 PROC LMIXED Statement Options

Option	Description
Basic Options	
BYOP=	Specifies the BY-group processing approach
DATA=	Specifies the input data table
METHOD=	Specifies the estimation method
NOBOUND	Removes boundary constraints on covariance parameters
NTHREADS=	Specifies the maximum number of concurrently scheduled threads to use
TIMING	Displays the "Timing" table
Options Related to Output	
ITDETAILS	Displays estimates and gradients added to the "Iteration History" table
MAXCLPRINT=	Specifies the maximum number of levels of CLASS variables to print
MMEQ	Displays the mixed model equations
NOCLPRINT	Suppresses the "Class Level Information" table completely or in part
RANKS	Displays the rank of the design matrix **X**
SEED=	Specifies the seed for pseudorandom number generation
SIMPLE	Displays the "Descriptive Statistics" table
Singularity Tolerances	
SINGCHOL=	Specifies the singularity criterion for Cholesky decomposition
SINGRES=	Specifies the singularity criterion for the residual variance
SINGULAR=	Specifies the general singularity criterion

Table 17.2 *continued*

Option	Description
Design Matrix Method	
DMMETHOD=	Specifies the computing method (sparse or dense)

You can specify the following *options*:

BYOP=PARALLEL | SEQUENTIAL
specifies the computational approach to take in BY-group processing. You can specify the following approaches:

PARALLEL performs the analysis for the BY groups in parallel on each thread. This approach can significantly improve performance when you have a lot of BY groups. But it requires a large amount of memory for a large number of BY groups.

SEQUENTIAL performs the analysis for the BY groups sequentially, one group at a time.

By default, BYOP=SEQUENTIAL.

DATA=*libref.data-table*
names the input data table for PROC LMIXED to use. The default is the most recently created data table. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to the data, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 864.

data-table specifies the name of the input data table.

DMMETHOD=DENSE | SPARSE<(*sparse-options***)>**
DMM=DENSE | SPARSE<(*sparse-options***)>**
specifies the method for computing the likelihood, gradient, and Hessian. For more information, see the section "Common Subject Effect" on page 943. Also see Table 17.13 for information about the covariance types that these two computing methods support.

You can specify the following methods:

DENSE is more suitable for models that have a large number of levels for a common subject effect and a relatively small number of levels for fixed effects.

SPARSE is more memory-efficient and faster for models that have a large and sparse design matrix. However, this method works only when you specify METHOD=REML. In general, the solutions for fixed-effects parameters are not the same as the solutions when you specify DMM=DENSE. But you can specify the DENSESOL option in the MODEL statement to request the solutions for fixed-effects parameters to match the solutions when DMM=DENSE.

For more information about these two methods, see the section "Computing Methodologies for REML in the Mixed Model" on page 940.

If you omit this option, the default is set according to the following rules:

- In fixed-effects models without CLASS variables, DMMETHOD=DENSE by default.
- In fixed-effects models with CLASS variables, DMMETHOD=SPARSE by default.
- In random-effects models without a common subject effect, DMMETHOD=SPARSE by default.
- In random-effects models with a common subject effect, DMMETHOD=DENSE by default.
- In models with a REPEATED statement, DMMETHOD=SPARSE by default.
- In all other situations, DMMETHOD=SPARSE.

You can specify the following *sparse-options* for the DMMETHOD=SPARSE option in parentheses:

METHOD=BASIC | NDORDERING
specifies the algorithm on which sparse matrix computing is based. You can specify the following algorithms:

BASIC is the algorithm with minimum memory consumption.

NDORDERING is the algorithm for permutation and inverse permutation. This algorithm achieves significant speed, but at the cost of memory.

By default, METHOD=BASIC unless the number of rows of the mixed model equations (MMEq) exceeds 2,000, in which case the default is METHOD=NDORDERING.

ITDETAILS
displays the parameter values at each iteration and enables the writing of notes to the SAS log that pertain to infinite likelihood and singularities during optimization iterations.

MAXCLPRINT=*number*
specifies the maximum number of levels of CLASS variables to print in the ODS table ClassLevels. MAXCLPRINT=0 enables you to print all levels of each CLASS variable. However, the NOCLPRINT option takes precedence over the MAXCLPRINT= option. By default, MAXCLPRINT=20.

METHOD=MIVQUE0 | ML | REML
specifies the estimation method for the covariance parameters.

MIVQUE0 performs minimum variance quadratic unbiased estimation of the covariance parameters (Rao 1972; Giesbrecht 1989).

ML performs maximum likelihood estimation.

REML performs residual (restricted) maximum likelihood estimation.

By default, METHOD=REML.

MMEQ
displays the coefficients of the mixed model equations, which are

$$\begin{bmatrix} \mathbf{X}'\widehat{\mathbf{R}}^{-1}\mathbf{X} & \mathbf{X}'\widehat{\mathbf{R}}^{-1}\mathbf{Z} \\ \mathbf{Z}'\widehat{\mathbf{R}}^{-1}\mathbf{X} & \mathbf{Z}'\widehat{\mathbf{R}}^{-1}\mathbf{Z}+\widehat{\mathbf{G}}^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{X}'\widehat{\mathbf{R}}^{-1}\mathbf{y} \\ \mathbf{Z}'\widehat{\mathbf{R}}^{-1}\mathbf{y} \end{bmatrix}$$

where $\widehat{\mathbf{G}}$ and $\widehat{\mathbf{R}}$ are estimates of $\mathbf{G}$ and $\mathbf{R}$.

For more information about these equations, see the section "Linear Mixed Models Theory" on page 935.

NOBOUND
removes boundary constraints on covariance parameters. This option is supported only when METHOD=MIVQUE0.

For example, variance components have a default lower boundary constraint of 0, and the NOBOUND option allows their estimates to be negative.

NOBLUPVAR
suppresses computing the covariances of random-effects estimates, which are used in the computation required by the OUTPUT statement and by the SOLUTION option in the RANDOM statement. Computing the covariances of random-effects estimates usually consumes a large amount of memory resources. You can use this option to reduce the consumption of memory resources when you are not interested in the covariances of random-effects estimates.

This option is supported only when DMMETHOD=DENSE.

NOCLPRINT<=*number*>
suppresses the display of the "Class Level Information" table if you do not specify *number*. If you specify *number*, the values of the classification variables are displayed only for variables whose number of levels is less than *number*. Specifying a *number* helps reduce the size of the "Class Level Information" table if some classification variables have a large number of levels.

NOINFO
suppresses the display of the "Model Information," "Number of Observations," and "Dimensions" tables.

NOITPRINT
suppresses the display of the "Iteration History" table.

NOPRINT
suppresses the normal display of results. This option is useful when you want only to create one or more output data tables by using the OUTPUT statement.

NOPROFILE
includes the residual variance as one of the covariance parameters in the optimization iterations. (By default, this parameter is profiled out of the optimization iterations, except when you have specified the HOLD= option in the PARMS statement.)

The NOPROFILE option is supported only when DMMETHOD=SPARSE.

NTHREADS=*number*

specifies the maximum number of concurrently scheduled threads for analytic computations and overrides the SAS system option THREADS or NOTHREADS. If you do not specify the NTHREADS= option, the number of threads is determined by the number of CPUs on the host on which the analytic computations run. PROC LMIXED creates one thread per CPU for the analytic computations.

By default, PROC LMIXED runs in multiple concurrent threads unless multithreading has been turned off by the NOTHREADS system option or by specifying NTHREADS=1. The largest *number* that you can specify is 1,024; however, the actual number of threads available depends on your system.

For more information about how PROC LMIXED uses threads, see the section "Multithreading" on page 102 in Chapter 2, "Shared Concepts."

NOTE: The SAS system option THREADS or NOTHREADS applies to the client machine on which the procedures that are based on CAS run. It does not apply to the compute nodes in a distributed environment.

RANKS

displays the rank of the design matrix **X**.

This option is supported only when DMMETHOD=SPARSE.

SEED=*number*

specifies an integer to be used to start the pseudorandom number generator. If you do not specify a seed, or if you specify a *number* less than or equal to 0, the seed is generated by reading the time of day from the computer's clock.

SIMPLE

displays the mean, standard deviation, coefficient of variation, minimum, and maximum for each variable that is not a classification variable in the input DATA= table.

SINGCHOL=*number*

tunes the singularity criterion in Cholesky decompositions. The default is 1E4 times the machine epsilon, which is approximately 1E–12 on most computers.

SINGRES=*number*

sets the tolerance for which the residual variance is considered to be zero. The default is 1E4 times the machine epsilon, which is approximately 1E–12 on most computers.

SINGULAR=*number*

tunes the general singularity criterion that the LMIXED procedure applies in sweeps and inversions. The default is 1E4 times the machine epsilon, which is approximately 1E–12 on most computers.

TIMING

displays the amount of time (in seconds) that PROC LMIXED took to perform each different task in the analysis and its percentage of the total time.

BLUP Statement

BLUP OUT=*libref.data-table* < *options* > **;**

The BLUP statement creates a data table that contains the best linear unbiased estimation (BLUE) of fixed effects and the best linear unbiased prediction (BLUP) of random effects. To use the BLUP statement, you must also use the PARMS statement to specify fixed values for all covariance parameters. When you use the BLUP statement, the ODS IterHistory table displays solver iterations that are used for BLUP solutions rather than displaying optimization iterations.

You must specify the following option:

OUT=*libref.data-table*
names the output data table for PROC LMIXED to use. You must specify this option before any other options. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to where the data table is to be stored, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 864.

data-table specifies the name of the output data table.

You can also specify the following *options*:

ITPRINT=*number*
displays that the iteration history after every *number* of iterations. By default, ITPRINT=10, which means the procedure displays the iteration history for every 10 iterations.

MAXITER=*number*
specifies the maximum number of iterations. The default value is the number of parameters in the OUT= data table plus 2.

SOLVER=DIRECT | IOC | IOD
specifies the solver to use for BLUP solutions.

DIRECT requires storing mixed model equations (MMEQ) in memory and computing the Cholesky decomposition of MMEQ. This solver is the most accurate, but it is the most inefficient in terms of speed and memory.

IOC requires storing mixed model equations in memory and iterates on MMEQ to solve for the solutions.This solver is the most efficient in terms of speed.

IOD does not build mixed model equations; instead it iterates on data to solve for the solutions. This solver is most efficient in terms of memory.

By default, SOLVER=IOC.

TOL=*number*
specifies the tolerance value. The default value is the square root of machine precision.

BY Statement

BY *variables* **;**

You can specify a BY statement in PROC LMIXED to obtain separate analyses of observations in groups that are defined by the values of the BY variables. If you specify more than one BY statement, only the last one specified is used. For more information, see the discussion of BY-group processing in the "Grouping Data" section of *SAS Programmers Guide: Essentials*.

CLASS Statement

CLASS *variable* < **(***options***)** >. . . < *variable* < **(***options***)** > > < / *global-options* > **;**

The CLASS statement names the classification variables to be used as explanatory variables in the analysis. Table 17.3 summarizes the values that you can use as either an *option* or a *global-option*. The options are fully documented in the section "CLASS Statement" on page 9 in Chapter 2, "Shared Concepts."

Table 17.3 CLASS Statement Options

Option	Description
DESCENDING	Reverses the sort order
MISSING	Treats missing values as valid levels
ORDER=	Specifies the sort order for the levels
PARAM=	Specifies the parameterization of the variable
REF=	Specifies the reference level of the variable and puts this level at the end of the list of levels. This interpretation of the specified REF= level does not apply when DMMETHOD=SPARSE, because in this case PROC LMIXED does not necessarily solve mixed model equations in the original order.

Note that the term 'INTERCEPT' is reserved as a keyword in several statements in PROC LMIXED. Therefore, you cannot use it it as a classification variable name in the CLASS statement.

CONTRAST Statement

CONTRAST *'label' contrast-specification* < **,** *contrast-specification* > < **,** . . . > < / *options* > **;**

The CONTRAST statement provides a mechanism for obtaining custom hypothesis tests. It is patterned after the CONTRAST statement in PROC MIXED and enables you to select an appropriate inference space (McLean, Sanders, and Stroup 1991).

You can test the hypothesis $\mathbf{L}'\boldsymbol{\phi} = \mathbf{0}$, where $\mathbf{L}' = [\mathbf{K}' \; \mathbf{M}']$ and $\boldsymbol{\phi}' = [\boldsymbol{\beta}' \; \boldsymbol{\gamma}']$, in several inference spaces. The inference space corresponds to the choice of $\mathbf{M}$. When $\mathbf{M} = \mathbf{0}$, your inferences apply to the entire population from which the random effects are sampled; this is known as the *broad* inference space. When all elements of $\mathbf{M}$ are nonzero, your inferences apply only to the observed levels of the random effects. This is known as the *narrow* inference space, and you can also choose it by specifying all the random effects as fixed. The GLM procedure uses the narrow inference space. Finally, by zeroing portions of $\mathbf{M}$ that correspond to selected main effects and interactions, you can choose *intermediate* inference spaces. The broad inference space is usually the most appropriate, and it is used when you do not specify any random effects in the CONTRAST statement.

You can specify the following:

label identifies the contrast in the table. A label is required for every contrast that you specify. Labels can be up to 20 characters long and must be enclosed in single quotes.

contrast-specification identifies the fixed and random effects and their coefficients from which the $\mathbf{L}'$ matrix is formed. Each *contrast-specification* is referred to as a row in the $\mathbf{L}'$ matrix. Specifically, *contrast-specification* takes the form

< *fixed-effect values* . . . > < | *random-effect values* . . . >

fixed-effect identifies an effect that appears in the MODEL statement. The keyword INTERCEPT can be used as an effect when an intercept is fitted in the model. You do not need to include all effects that appear in the MODEL statement.

random-effect identifies an effect that appears in the RANDOM statement. The first random effect must follow a vertical bar (|); however, random effects are not required.

values are constants that are elements of the $\mathbf{L}$ matrix associated with the fixed and random effects.

The *contrast-specification* is specified by a series of nonpositional units. Each unit starts with an effect name, followed by a list of elements in square brackets. In each set of square brackets, you enter the coefficient and level indicators for each variable in an effect. A nonpositional unit has the following form:

```
effect-name [coefficient, level indicators]
            [coefficient, level indicators]
            ...
            [coefficient, level indicators]
```

For example, in the following CONTRAST statement, the *contrast-specification* has two nonpositional units. The first unit is “b [1, 1] [-1, 2]”, and the second unit is “a*b [1, 2 1] [-1, 2 2]”.

```
class a b;
model y = a b a*b;
contrast 'B at A2'
          b   [1, 1]   [-1, 2]
          a*b [1, 2 1] [-1, 2 2];
```

The first nonpositional unit starts with effect 'b', followed by two sets of square brackets. In bracket [1,1], the first element is the coefficient, and the second element is the level indicator 1, which corresponds to the first level of the CLASS variable 'b'.

The second nonpositional unit starts with effect 'a*b', followed by two sets of square brackets, [1, 2 1] and [-1, 2 2]. In bracket [1,2 1], the coefficient is 1, and the level indicators are the second level of variable 'a' and the first level of variable 'b'. In bracket [-1,2 2], the coefficient is -1, and the level indicators are the second level of variable 'a' and the second level of variable 'b'.

The comma that separates the coefficient and level indicators in a bracket is optional.

The rows of the $\mathbf{K}'$ component of $\mathbf{L}'$ are specified on the left side of the vertical bars (|). These rows test the fixed effects and are therefore checked for estimability. The rows of the $\mathbf{M}'$ component of $\mathbf{L}'$ are specified on the right side of the vertical bars. They test the random effects, and no estimability checking is necessary.

If PROC LMIXED finds the fixed-effects portion in at least one specified *contrast-specification* to be nonestimable (see the SINGULAR= option on page 880), then it determines that the whole contrast is nonestimable and displays missing values for the test statistics and a note in the log.

Assume that there are two fixed effects, 'a' and 'a*b'. If you specify the coefficient for effect 'a' but not for effect 'a*b', then the coefficients for effect 'a' are automatically "filled in" over the levels of the higher-order effect 'a*b'. This feature is designed to preserve estimability in cases where there are complex higher-order effects. The coefficients for the higher-order effect are determined by equitably distributing the coefficients of the lower-level effect. In addition, if the intercept is specified, it is distributed over all classification effects that are not contained by any other specified effect. If an effect is not specified for coefficients and does not contain any coefficient-specified effects, then all of its coefficients in $\mathbf{L}$ are set to 0. You can override this default by specifying coefficients for the higher-order effect.

If you specify a SUBJECT effect in the RANDOM statement, then the coefficients that you specify for the effects in the RANDOM statement are equitably distributed across the levels of the SUBJECT effect. You can use the E option to see exactly what $\mathbf{L}$ matrix is used.

The SUBJECT and GROUP options in the CONTRAST statement are useful for the case where a SUBJECT= or GROUP= variable appears in the RANDOM statement, and you want to contrast different subjects or groups. By default, CONTRAST statement coefficients about random effects are distributed equally across subjects and groups.

PROC LMIXED handles missing level combinations of CLASS variables similarly to the way PROC GLM does. Both procedures delete fixed-effects parameters that correspond to missing levels in order to preserve estimability. However, PROC LMIXED does not delete missing level combinations for random-effects parameters because linear combinations of the random-effects parameters are always estimable. These conventions can affect the way that you specify the CONTRAST statement coefficients.

The CONTRAST statement computes the statistic

$$F = \frac{\begin{bmatrix} \widehat{\boldsymbol{\beta}} \\ \widehat{\boldsymbol{\gamma}} \end{bmatrix}' \mathbf{L}(\mathbf{L}'\widehat{\mathbf{C}}\mathbf{L})^{-1}\mathbf{L}' \begin{bmatrix} \widehat{\boldsymbol{\beta}} \\ \widehat{\boldsymbol{\gamma}} \end{bmatrix}}{r}$$

where $r = \mathrm{rank}(\mathbf{L}'\widehat{\mathbf{C}}\mathbf{L})$ and approximates its distribution with an F distribution. In this expression, $\widehat{\mathbf{C}}$ is an estimate of the generalized inverse of the coefficient matrix in the mixed model equations.

The numerator degrees of freedom in the F approximation is $r = \mathrm{rank}(\mathbf{L}'\widehat{\mathbf{C}}\mathbf{L})$, and the denominator degrees of freedom is the residual degrees of freedom that you define by specifying the DDFM=RESIDUAL option. You can change the denominator degrees of freedom by using the DF= option.

Results from all CONTRAST statements are displayed together in the Contrasts ODS table.

You can specify the following *options* in the CONTRAST statement after a slash (/):

CHISQ
performs χ^2 tests in addition to any F tests. A χ^2 statistic equals its corresponding F statistic times the associate numerator degrees of freedom, and the same degrees of freedom is used to compute the p-value for the χ^2 test. This p-value will always be less than that for the F test, because it effectively corresponds to an F test with infinite denominator degrees of freedom.

DF=*number*
specifies the denominator degrees of freedom for the F test. The default is the residual degrees of freedom that you define by specifying the DDFM=RESIDUAL option.

E
displays the **L** matrix coefficients for the contrast. In the ODS output, the name of this "L Matrix Coefficients" table is Coef.

GROUP *coeffs*
sets up random-effect contrasts between different groups when a GROUP= variable appears in the RANDOM statement. By default, CONTRAST statement coefficients about random effects are distributed equally across groups. You can enter a multiple-row contrast, but you can enter only one row for the GROUP coefficients. For example, the following CONTRAST statement has a GROUP option with two sets of brackets; these two sets of brackets apply to each row of contrast.

```
contrast 'Trt @ x=0.4 and 0.5' trt [1 1] [-1 2] | x [0.4 1],
                               trt [1 1] [-1 3] | x [0.4 1],
                               trt [1 1] [-1 2] | x [0.5 1],
                               trt [1 1] [-1 3] | x [0.5 1] /
                               group [1 1] [-1 2];
```

If the GROUP effect has two levels and random effect 'x' has one level, then the specified coefficient 0.4 in the first contrast row is expanded across two levels of the GROUP effect weighted by GROUP coefficients. The resulting coefficients are

```
0.4 -0.4
```

SINGULAR=*number*

tunes the estimability checking. If $\mathbf{U}$ is a matrix, define ABS($\mathbf{U}'$) to be a vector $\mathbf{v}$ whose element v_i is the largest absolute value of the elements of row i of $\mathbf{U}'$. If one of the elements in (ABS($\mathbf{K}' - \mathbf{K}'\mathbf{T}$) - ABS($\mathbf{K}'$)**number*) is greater than zero, then the corresponding row of $\mathbf{K}'$ is declared nonestimable. If any row is nonestimable, then $\mathbf{K}$ is declared nonestimable. Here $\mathbf{T}$ is the Hermite form matrix $(\mathbf{X}'\mathbf{X})^{-}\mathbf{X}'\mathbf{X}$. The value of *number* must be between 0 and 1, inclusive. By default, SINGULAR=1E–4.

SUBJECT *coeffs*

sets up random-effects contrasts between different subjects when a SUBJECT= variable appears in the RANDOM statement. By default, CONTRAST statement coefficients about random effects are distributed equally across subjects. The rules for listing subject coefficients for multiple-row CONTRAST statements are the same as the rules for listing GROUP coefficients.

DISPLAY Statement

DISPLAY < *table-list* > < / *options* > **;**

The DISPLAY statement enables you to specify a list of display tables to display or exclude. This statement is similar to the ODS SELECT, ODS EXCLUDE, and ODS TRACE statements. However, the DISPLAY statement can improve performance when a large number of tables could be generated (such as in BY-group processing). The procedure processes the DISPLAY statement on a CAS server and thus sends only a subset of ODS tables to the SAS client. Because ODS statements are processed on a SAS client, first all the generated display tables are sent to the client, and then the client creates a subset.

If you use both DISPLAY and ODS statements together, the DISPLAY statement takes precedence over the ODS statements. Note that the ODS EXCLUDE statement processes tables that are sent to the client after they have been filtered by the DISPLAY statement. In some cases, it might appear that the ODS EXCLUDE statement is taking precedence because it can further filter the tables. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

You can specify the *table-list* as a list of table names, paths, partial pathnames, and regular expressions.

The table names that you can specify are listed in the section "ODS Table Names" on page 947. A path is a table name that is prefixed with dot-separated grouping information. For example, a SelectionSummary table that a procedure produces during a selection routine might have the path *Bygroup1.Summary.SelectionSummary*. A partial pathname does not include all groups; for example, *SelectionSummary* and *Summary.SelectionSummary* are partial pathnames for *Bygroup1.Summary.SelectionSummary*.

When you specify a table name or partial pathname, all display tables whose paths end in the specified name are selected for display or exclusion. For example, both *SelectionSummary* and *Summary.SelectionSummary* select *Bygroup1.Summary.SelectionSummary*.

A regular expression is enclosed in forward slashes (/). For example, specifying "/tions/" selects all pathnames that contain the substring "tions"; in particular, the *Bygroup1.Summary.SelectionSummary* table is selected. Specifying "!/tions/" selects all pathnames that do not contain the substring "tions"; in particular, the *Bygroup1.Summary.SelectionSummary* table is not selected.

You can specify the following *options* after a slash (/):

CASESENSITIVE
performs a case-sensitive comparison of table names in the *table-list* to display table names when tables are subsetted for display. To preserve case, you must enclose table names in the *table-list* in quotation marks.

EXCLUDE
displays all display tables except those that you specify in the *table-list*.

EXCLUDEALL
suppresses display of all tables. This option takes precedence over the other options.

TRACE
displays the display table names, labels, and paths.

DISPLAYOUT Statement

DISPLAYOUT *table-spec-list* < / *options* > **;**

The DISPLAYOUT statement enables you to create output tables from your displayed output. This statement is similar to the ODS OUTPUT statement. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

The *table-spec-list* specifies a list of output tables to create. Each entry in the list has either a *key*=*value* format or a *key* format:

key=*value*	specifies *key* as the ODS table name, path, or partial pathname, and specifies *value* as the output table name.
key	specifies *key* as the ODS table name and also as the output table name.

The ODS table names that you can specify are listed in the section "ODS Table Names" on page 947. You cannot specify the ODS table named OutputCasTables in the *table-spec-list*.

Table names and partial pathnames are discussed under the DISPLAY statement. The DISPLAYOUT statement does not support regular expressions.

You can specify the following *options* after a slash (/):

INCLUDEALL
creates output tables for all display tables. The name of the created output table is the same as the name of the corresponding display table. If you specify this option, the *table-spec-list* specification is ignored.

NOREPLACE
does not replace any existing output table of the same name.

REPEATED
replicates all output tables on all nodes.

EFFECT Statement

EFFECT *name*=*effect-type* **(***variables* < / *options* >**) ;**

The EFFECT statement enables you to construct special collections of columns for design matrices. These collections are called *constructed effects* in order to distinguish them from the usual model effects that are formed from continuous or classification variables, as discussed in the section "GLM Parameterization of Classification Variables and Effects" on page 67 in Chapter 2, "Shared Concepts."

You can specify the following *effect-types*:

COLLECTION specifies a collection effect that defines one or more variables as a single effect that has multiple degrees of freedom. The variables in a collection are considered as a unit for purposes of estimation and inference.

MULTIMEMBER | MM specifies a multimember classification effect whose levels are determined by one or more variables that appear in a CLASS statement.

POLYNOMIAL | POLY specifies a multivariate polynomial effect in the specified numeric variables.

SPLINE specifies a regression spline effect whose columns are univariate spline expansions of one or more variables. A spline expansion replaces the original variable with an expanded or larger set of new variables.

For more information about the syntax of these *effect-types* and how columns of constructed effects are computed, see the section "EFFECT Statement" on page 18 in Chapter 2, "Shared Concepts."

Table 17.4 summarizes the *options* available in the EFFECT statement.

Table 17.4 EFFECT Statement Options

Option	Description
Collection Effects Options	
DETAILS	Displays the constituents of the collection effect
Multimember Effects Options	
DETAILS	Displays the levels of the multimember effect
NOEFFECT	Requests that observations whose levels are all missing for the multimember variables have 0 values in the corresponding design matrix columns
STDIZE	Standardizes the design matrix entries so that each observation has a sum of 1
WEIGHT=	Specifies the weight variable for the contributions of each classification effect
Polynomial Effects Options	
DEGREE=	Specifies the degree of the polynomial

Table 17.4 *continued*

Option	Description
DETAILS	Displays details of the specified polynomial
MDEGREE=	Specifies the maximum degree of any variable in a term of the polynomial
NOSEPARATE	Treats the polynomial as a single effect with multiple degrees of freedom
STANDARDIZE=	Specifies centering and scaling suboptions for the variables that define the polynomial
Spline Effects Options	
BASIS=	Specifies the type of basis (B-spline basis or truncated power function basis) for the spline effect
DATABOUNDARY	Uses the extremes of the data as boundary knots for a B-spline basis
DEGREE=	Specifies the degree of the spline effect
DETAILS	Displays the knots and locations for each spline basis function
KNOTMAX=	Requests equally spaced right-side boundary knots starting at the variables' maximum and ending at the KNOTMAX= value
KNOTMETHOD=	Specifies how to construct the knots for the spline effect
KNOTMIN=	Requests equally spaced left-side boundary knots starting at the KNOTMIN= value and ending at the variables' minimum value
NATURALCUBIC	Specifies a natural cubic spline basis for the spline effect

ESTIMATE Statement

ESTIMATE '*label*' *contrast-specification* < **(***divisor***=***n***)** >
< , '*label*' *contrast-specification* < **(***divisor***=***n***)** > > < , . . . > < / *options* > **;**

The ESTIMATE statement provides a mechanism for obtaining custom linear estimate of the fixed and/or random effects. As in the CONTRAST statement, the basic element of the ESTIMATE statement is the *contrast-specification*, which consists of MODEL and RANDOM effects and their coefficients. Specifically, a *contrast-specification* takes the form

< *fixed-effect values* . . . > < | *random-effect values* . . . >

Based on the *contrast-specifications* in your ESTIMATE statement, PROC LMIXED constructs the matrix $\mathbf{L}' = [\mathbf{K}' \ \mathbf{M}']$, as in the CONTRAST statement, where $\mathbf{K}$ is associated with the fixed effects and $\mathbf{M}$ is associated with the G-side random effects. Each *contrast-specification* is referred to as a row in the $\mathbf{L}'$ matrix.

PROC LMIXED then produces for each row $\mathbf{l}$ of $\mathbf{L}'$ an approximate t test of the hypothesis $H: \mathbf{l}\boldsymbol{\phi} = 0$, where $\boldsymbol{\phi} = [\boldsymbol{\beta}' \ \boldsymbol{\gamma}']'$.

Note that multiple-row estimates are permitted. Unlike the CONTRAST statement, you need to specify a '*label*' for every row of the multiple-row estimate, because PROC LMIXED produces one test per row.

PROC LMIXED checks the estimability of the fixed-effects portion in each row of the multiple-row estimate.

If one row is found to be nonestimable (see the SINGULAR= option on page 886), then the procedure displays missing values for that row.

PROC LMIXED uses the residual degrees of freedom that you define by specifying the DDFM=RESIDUAL option. You can modify the degrees of freedom by using the DF= option. If you select DDFM=NONE and do not modify the degrees of freedom by using the DF= option, PROC LMIXED uses infinite degrees of freedom, essentially computing approximate z tests.

If PROC LMIXED finds the fixed-effects portion of the specified estimate to be nonestimable, then it displays "Non-est" for the estimate entry.

The construction of the **L** matrix for an ESTIMATE statement follows the same rules that are listed for the CONTRAST statement.

Table 17.5 summarizes the *options* available in the ESTIMATE statement.

Table 17.5 ESTIMATE Statement Options

Option	Description
ADJUST=	Specifies the method to use for multiple comparison adjustment of estimates
ALPHA=	Specifies the confidence level
CL	Constructs t-type confidence limits
DF=	Specifies the degrees of freedom
DIVISOR=	Specifies values to divide the coefficients
E	Displays the matrix coefficients
GROUP	Sets up random-effect contrasts between groups
LOWER	Performs one-sided, lower-tailed inference
SINGULAR=	Tunes the estimability checking
SUBJECT	Sets up random-effect estimates between subjects
UPPER	Performs one-sided, upper-tailed inference

Results from all ESTIMATE statements are displayed together in the Estimates ODS table.

You can specify the following *options* in the ESTIMATE statement after a slash (/):

ADJUST=BON | SCHEFFE | SIDAK | SIMULATE< (*simoptions*) > | T
performs a multiple comparison adjustment of the p-values and confidence limits for the estimates. The adjusted quantities are produced in addition to the unadjusted quantities. Adjusted confidence limits are produced if you specify the CL or ALPHA= option. For a description of the adjustments, see the ADJUST= option in the LSMEANS statement.

ALPHA=*number*
constructs a t-type confidence interval with confidence level $1 - number$. The value of *number* must be between 0 and 1, exclusive; the default is 0.05. If DDFM=NONE and you do not specify degrees of freedom by using the DF= option, PROC LMIXED uses infinite degrees of freedom, essentially computing a z interval.

CL

constructs *t*-type confidence limits. If DDFM=NONE and you do not specify degrees of freedom by using the DF= option, PROC LMIXED uses infinite degrees of freedom, essentially computing a *z* interval. By default, the confidence level is 0.95.

DF=*number*

specifies the degrees of freedom for the *t* test. The default is the residual degrees of freedom that you define by specifying the DDFM=RESIDUAL option.

DIVISOR=*value-list*

specifies a list of values by which to divide the coefficients so that fractional coefficients can be entered as integer numerators. If you do not specify a *value-list*, a default value of 1.0 is assumed. Missing values in the *value-list* are converted to 1.0.

If the number of elements in the *value-list* exceeds the number of rows of the estimate, the extra values are ignored. If the number of elements in the *value-list* is less than the number of rows of the estimate, the last value in the *value-list* is carried forward.

If you specify a row-specific divisor as part of the specification of the estimate row, this value multiplies the corresponding divisor implied by the *value-list*. For example, the following statement divides the coefficients in the first row by 8 and divides the coefficients in the third and fourth rows by 3:

```
estimate 'One vs. two'   A [2 1] [-2 2] (divisor=2),
         'One vs. three' A [1 1] [-1 3]            ,
         'One vs. four'  A [3 1] [-3 4]            ,
         'One vs. five'  A [1 1] [-1 5] / divisor=4,.,3;
```

E

displays the matrix coefficients. In the ODS output, the name of this "L Matrix Coefficients" table is Coef.

GROUP *coeffs*

sets up random-effect contrasts between different groups when a GROUP= variable appears in the RANDOM statement. By default, ESTIMATE statement coefficients about random effects are distributed equally across groups. You can enter a multiple-row estimate, but you can enter only one row for the GROUP coefficients. For example, the following ESTIMATE statement has a GROUP option with two sets of brackets; these two sets of brackets apply to each row of the estimate.

```
estimate 'Trt 1 vs 2 @ x=0.4' trt [1 1] [-1 2] | x [0.4 1],
         'Trt 1 vs 3 @ x=0.4' trt [1 1] [-1 3] | x [0.4 1],
         'Trt 1 vs 2 @ x=0.5' trt [1 1] [-1 2] | x [0.5 1],
         'Trt 1 vs 3 @ x=0.5' trt [1 1] [-1 3] | x [0.5 1] /
                              group [1 1] [-1 2];
```

If the GROUP effect has two levels and random effect 'x' has one level, then the specified coefficient 0.4 in the first contrast row is expanded across two levels of the GROUP effect weighted by the GROUP coefficients. The resulting coefficients are

```
0.4 -0.4
```

LOWER

LOWERTAILED

bases the *p*-value for the *t* test only on values less than the test statistic. A two-tailed test is the default. A lower-tailed confidence limit is also produced if you specify the CL or ALPHA= option.

SINGULAR=*number*

tunes the estimability checking as documented for the SINGULAR= option in the CONTRAST statement.

SUBJECT *coeffs*

sets up random-effect estimates between different subjects when a SUBJECT= variable appears in the RANDOM statement. By default, ESTIMATE statement coefficients about random effects are distributed equally across subjects. The rules for listing subject coefficients for an ESTIMATE statement with multiple rows are the same as the rules for listing GROUP coefficients.

UPPER

UPPERTAILED

bases the *p*-value for the *t* test only on values greater than the test statistic. A two-tailed test is the default. An upper-tailed confidence limit is also produced if you specify the CL or ALPHA= option.

FREQ Statement

FREQ *variable* **;**

The *variable* in the FREQ statement identifies a numeric variable in the data set. PROC LMIXED treats each observation as if it appears f times, where f is the value of the FREQ variable for the observation. If it is not an integer, the frequency value is truncated to an integer. If the frequency value is less than 1 or missing, the observation is not used in the analysis. When the FREQ statement is omitted, each observation is assigned a frequency of 1.

The analysis that you produce by using a FREQ statement reflects the expanded number of observations. For example, PROC LMIXED produces the same results for the following two data sets if variable count is used in the FREQ statement:

```
data one;
input p$ a$ b y count;
datalines;
P1 A1 1 8  2
P1 A1 2 10 1
P2 A2 1 7  3
P2 A1 2 8  2
P3 A1 1 12 2
P3 A2 2 9  1
P1 A2 1 7  3
P1 A2 2 8  2
```

```
P2 A2 1 7  3
P2 A2 2 9  2
;

data one;
input p$ a$ b y;
datalines;
P1 A1 1 8
P1 A1 1 8
P1 A1 2 10
P2 A2 1 7
P2 A2 1 7
P2 A2 1 7
P2 A1 2 8
P2 A1 2 8
P3 A1 1 12
P3 A1 1 12
P3 A2 2 9
P1 A2 1 7
P1 A2 1 7
P1 A2 1 7
P1 A2 2 8
P1 A2 2 8
P2 A2 1 7
P2 A2 1 7
P2 A2 1 7
P2 A2 2 9
P2 A2 2 9
;
```

LSMEANS Statement

LSMEANS *fixed-effects* < / *options* > **;**

The LSMEANS statement computes least squares means (LS-means) of classification fixed effects. As in the GLM procedure, LS-means are *predicted population margins*—that is, they estimate the marginal means over a balanced population. In a sense, LS-means are to unbalanced designs as classification and subclassification arithmetic means are to balanced designs. The **L** matrix constructed to compute them is the same as the **L** matrix formed in PROC GLM; however, the standard errors are adjusted for the covariance parameters in the model.

Each LS-mean is computed as $\mathbf{L}'\widehat{\boldsymbol{\beta}}$, where $\mathbf{L}$ is the coefficient matrix associated with the least squares mean and $\widehat{\boldsymbol{\beta}}$ is the estimate of the fixed-effects parameter vector. The approximate standard errors for the LS-mean is computed as the square root of $\mathbf{L}'(\mathbf{X}'\widehat{\mathbf{V}}^{-1}\mathbf{X})^{-}\mathbf{L}$.

LS-means can be computed for any effect in the MODEL statement that involves CLASS variables. You can specify multiple effects in one LSMEANS statement or in multiple LSMEANS statements, and all LSMEANS statements must appear after the MODEL statement. As in the ESTIMATE statement, the **L** matrix is tested for estimability, and if this test fails, PROC LMIXED displays "Non-est" for the LS-means entries.

Assuming the LS-mean is estimable, PROC LMIXED constructs an approximate t test to test the null hypothesis that the associated population quantity equals zero. By default, the denominator degrees of freedom for this test are the residual degrees of freedom.

Table 17.6 summarizes the *options* available in the LSMEANS statement.

Table 17.6 LSMEANS Statement Options

Option	Description
ADJUST=	Determines the method of multiple comparison adjustment of LS-means differences
ALPHA=	Specifies the confidence level
AT=	Modifies covariate values in computing LS-means
CL	Constructs *t*-type confidence limits
CORR	Displays the estimated correlation matrix
COV	Displays the estimated covariance matrix
DF=	Specifies the degrees of freedom
DIFF	Displays the differences of the LS-means
E	Displays the matrix coefficients for LSMEANS effects
SINGULAR=	Tunes the estimability checking
SLICE=	Partitions interaction LSMEANS effects
STEPDOWN	Adjusts multiple comparison *p*-values further through a step-down adjustment process

You can specify the following *options* in the LSMEANS statement after a slash (/).

ADJUST=*method*
requests a multiple comparison adjustment for the *p*-values and confidence limits for the differences of LS-means. The adjusted quantities are produced along with the unadjusted quantities. The ADJUST= option works together with the DIFF option. If you omit the DIFF option, then the ADJUST= option implies the DIFF option except for ADJUST=T.

You can specify the following *methods* for the ADJUST= option:

BON performs Bonferroni *t* tests of differences between LS-means. The method involves correction factors described in Chapter 53, "The GLM Procedure" (*SAS/STAT User's Guide*), and Chapter 86, "The MULTTEST Procedure" (*SAS/STAT User's Guide*); also see Westfall and Young (1993) and Westfall et al. (1999).

DUNNETT performs Dunnett's *t* test, testing whether any treatments are significantly different from a single control for the effects in the LSMEANS statement. If the LS-means are correlated, PROC LMIXED uses the factor-analytic covariance approximation described in Hsu (1992) and identifies the adjustment as "Dunnett-Hsu" in the results. The approximation derives approximate "effective sample sizes" for which exact critical values are computed.

NELSON performs Nelson's *t* test of each LS-mean against an average of the LS-means (Ott 1967; Nelson 1982, 1991, 1993). When the LS-means are correlated, PROC LMIXED uses the factor-analytic covariance approximation described in Hsu (1992) and identifies the adjustment as "Nelson-Hsu" in the results. The approximation derives approximate "effective sample sizes" for which exact critical values are computed.

SCHEFFE performs Scheffé's multiple comparison procedure.

SIDAK performs pairwise t tests on differences between LS-means with levels adjusted according to Šidák's inequality. The method involves correction factors described in Chapter 53, "The GLM Procedure" (*SAS/STAT User's Guide*), and Chapter 86, "The MULTTEST Procedure" (*SAS/STAT User's Guide*); also see Westfall and Young (1993) and Westfall et al. (1999).

SIMULATE < (*simoptions*) > performs the simulation-based multiple comparison procedure. This method computes adjusted p-values and confidence limits from the simulated distribution of the maximum or maximum absolute value of a multivariate t random vector. All covariance parameters, except the residual scale parameter, are fixed at their estimated values throughout the simulation, potentially resulting in some underdispersion. The simulation estimates q, the true $(1-\alpha)$ quantile, where $1-\alpha$ is the confidence coefficient. The default value of α is 0.05; you can change this value by specifying the ALPHA= option in the LSMEANS statement.

The number of samples is set so that the tail area for the simulated q is within γ of $1-\alpha$ with $100(1-\epsilon)\%$ confidence. In equation form,

$$\Pr(|F(\widehat{q}) - (1-\alpha)| \leq \gamma) = 1-\epsilon$$

where $\hat{q}$ is the simulated q and F is the true distribution function of the maximum; see Edwards and Berry (1987) for details. By default, $\gamma = 0.005$ and $\epsilon = 0.01$, placing the tail area of $\hat{q}$ within 0.005 of 0.95 with 99% confidence.

You can specify the following *simoptions* in parentheses after the ADJUST=SIMULATE option:

ACC=*value* specifies the target accuracy radius γ of a $100(1-\epsilon)\%$ confidence interval for the true probability content of the estimated $(1-\alpha)$ quantile. By default, ACC=0.005. Note that, if you also specify the CVADJUST *simoption*, then the actual accuracy radius will probably be substantially less than this target.

CVADJUST specifies that the quantile be estimated by the control variate adjustment method of Hsu and Nelson (1998) instead of simply as the quantile of the simulated sample. Specifying this option usually has the effect of significantly reducing the accuracy radius γ of a $100 \times (1-\epsilon)\%$ confidence interval for the true probability content of the estimated $(1-\alpha)$ quantile. The control-variate-adjusted quantile estimate takes approximately twice as long to compute, but it is typically much more accurate than the sample quantile.

EPS=*value* specifies the value ϵ for a $100 \times (1-\epsilon)\%$ confidence interval for the true probability content of the estimated $(1-\alpha)$ quantile. The default value of the accuracy confidence is 99%, corresponding to EPS=0.01.

NSAMP=*n* specifies the sample size for the simulation. By default, *n* is set by a formula that is based on the values of the target accuracy radius γ and accuracy confidence $100 \times (1-\epsilon)\%$ for an interval

for the true probability content of the estimated $(1-\alpha)$ quantile. When the default values are used for γ, ϵ, and α (0.005, 0.01, and 0.05, respectively), NSAMP=12605.

REPORT — specifies that a report on the simulation be displayed, including a listing of the parameters, such as γ, ϵ, and α, as well as an analysis of various methods of estimating or approximating the quantile.

SEED=*number* — specifies an integer to be used to start the pseudorandom number generator for the simulation. If you do not specify a seed, or if you specify a *number* less than or equal to 0, the seed is generated by reading the time of day from the computer's clock.

SMM | GT2 performs pairwise comparisons based on the studentized maximum modulus and Šidák's uncorrelated-*t* inequality, yielding Hochberg's GT2 method when sample sizes are unequal.

T | NONE performs no adjustment for multiple comparisons.

TUKEY performs Tukey's studentized range test on LS-means. When your data are unbalanced, PROC LMIXED uses the approximation described in Kramer (1956) and identifies the adjustment as "Tukey-Kramer" in the results.

By default, ADJUST=T | NONE.

If you specify ADJUST=DUNNETT, the procedure analyzes all differences with the first control level unless you have specified different control level. If you specify ADJUST=NELSON, then DIFF=ANOM is assumed.

If you specify ADJUST=TUKEY, then DIFF=ALL is assumed. If you specify other *methods*, then the procedure performs all pairwise differences unless you specify the DIFF option.

Note that computing the exact adjusted *p*-values and critical values for unbalanced designs can be computationally intensive, especially for ADJUST=NELSON.

ALPHA=*number*

requests that a *t*-type confidence interval be constructed for each of the LS-means with confidence level $1-$ *number*. The value of *number* must be between 0 and 1; the default is 0.05.

AT *at-specification*

enables you to modify the values of the covariates that are used in computing LS-means. By default, all covariates are set equal to their mean values for computing standard LS-means. The AT option enables you to assign arbitrary values to the covariates. Additional columns in the output table indicate the values of the covariates.

You can specify the following *at-specifications*:

MEANS sets covariates equal to their mean values (as with standard LS-means) and applies this adjustment to crossproducts of covariates.

variable=*value* sets the covariate *variable* equal to *value*.

(*variable-list*)=(*value-list*) sets covariates in the *variable-list* equal to values in the *value-list*.

As an example, consider the following invocation of PROC LMIXED:

```
proc lmixed;
   class A B;
   model Y = A*B x1 x2 x1*x2;
   LSMEANS A*B;
   LSMEANS A*B / at means;
   LSMEANS A*B / at x1=1.2;
   LSMEANS A*B / at (x1 x2)=(1.2 0.3);
run;
```

For the first two LSMEANS statements, the LS-means coefficient of x1 is $\overline{x}_1$ (the mean of x1) and of x2 is $\overline{x}_2$ (the mean of x2). For the first LSMEANS statement, the coefficient of x1*x2 is $\overline{x_1 x_2}$. However, for the second LSMEANS statement, the coefficient is $\overline{x}_1 \times \overline{x}_2$. The third LSMEANS statement sets the coefficient of x1 equal to 1.2 and leaves it at $\overline{x}_2$ for x2, and the final LSMEANS statement sets these values to 1.2 and 0.3, respectively.

Even if you specify a WEIGHT variable, the unweighted covariate means are used for the covariate coefficients if you omit the AT option. If you specify the AT option, then WEIGHT or FREQ variables are taken into account as follows. The weighted covariate means are used for the covariate coefficients for which no explicit AT option values are given, or if you specify the AT MEANS option. Observations that do not contribute to the analysis because of a missing dependent variable are used in computing the covariate means. You should use the E option in conjunction with the AT option to verify that the modified LS-means coefficients are the ones that you want.

CL

requests that *t*-type confidence limits be constructed for each of the LS-means. If DDFM=NONE, then PROC LMIXED uses infinite degrees of freedom for this test, essentially computing a *z* interval. The confidence level is 0.95 by default; this can be changed with the ALPHA= option. If you specify an ADJUST= option, then the confidence limits are adjusted for multiplicity. If you also specify the STEPDOWN option, then only *p*-values, not the confidence limits, are step-down-adjusted.

CORR

displays the estimated correlation matrix of the least squares means as part of the "Least Squares Means" table.

COV

displays the estimated covariance matrix of the least squares means as part of the "Least Squares Means" table.

DF=*number*

specifies the degrees of freedom for the *t* test and confidence limits. The default is the residual degrees of freedom that you define by specifying the DDFM=RESIDUAL option.

DIFF<=*difftype***>**

PDIFF<=*difftype***>**

requests that differences of the LS-means be displayed. You can specify the following values for the optional *difftype*.

ALL requests all pairwise differences.

ANOM requests differences between each LS-mean and the average LS-mean, as in the analysis of means (Ott 1967). The average is computed as a weighted mean of the LS-means, with the weights being inversely proportional to the diagonal entries of the $\mathbf{L}\left(\mathbf{X}'\mathbf{X}\right)^{-}\mathbf{L}'$ matrix. When a WEIGHT statement is specified, then the preceding matrix is replaced with $\mathbf{L}\left(\mathbf{X}'\mathbf{W}\mathbf{X}\right)^{-}\mathbf{L}'$ where $\mathbf{W}$ is the diagonal matrix that contains the weights. If LS-means are nonestimable, this design-based weighted mean is replaced with an equally weighted mean. Note that the ANOM procedure in SAS/QC software implements both tables and graphics for the analysis of means with a variety of response types. For one-way designs and normally distributed data, the DIFF=ANOM computations are equivalent to the results of PROC ANOM.

CONTROL requests differences with a control; by default, the control is the first level of each of the specified LSMEANS effects. To specify which levels of the effects are the controls, list the quoted formatted values in parentheses after the CONTROL keyword. For example, if the effects A, B, and C are classification variables, each having two levels, 1 and 2, the following LSMEANS statement specifies the (1,2) level of A*B and the (2,1) level of B*C as controls:

```
lsmeans A*B B*C / diff=control('1' '2', '2' '1');
```

For multiple effects, the results depend upon the order of the list, and so you should check the output to make sure that the controls are correct.

CONTROL produces two-tailed tests and confidence limits.

CONTROLL requests one-tailed results and tests whether the noncontrol levels are significantly smaller than the control. The upper confidence limits for the control minus the noncontrol levels are considered to be infinity and are displayed as missing.

CONTROLU requests one-tailed results and tests whether the noncontrol levels are significantly larger than the control. The upper confidence limits for the noncontrol levels minus the control are considered to be infinity and are displayed as missing.

The differences of the LS-means are displayed in a table titled "Differences of Least Squares Means." The table name is "Diffs."

E

requests that the matrix coefficients for all LSMEANS effects be displayed. The name of this "Matrix Coefficients" table is "Coef."

PDIFF

is the same as the DIFF option. See the description of the DIFF option on page 891.

SINGULAR=*number*

tunes the estimability checking as documented for the SINGULAR= in the CONTRAST statement.

SLICE=*fixed-effect* | **(***fixed-effects***)**

SLICEBY=*fixed-effect* | **(***fixed-effects***)**

specifies effects by which to partition interaction LSMEANS effects. This can produce what are known as tests of simple effects (Winer 1971). For example, suppose that A*B is significant, and you want to test the effect of A for each level of B. The appropriate LSMEANS statement is

```
lsmeans A*B / slice=B;
```

This statement tests for the simple main effects of A for B, which are calculated by extracting the appropriate rows from the coefficient matrix for the A*B LS-means and by using them to form an *F* test.

The SLICE= option produces *F* tests that test the simultaneous equality of cell means at a fixed level of the slice effect (Schabenberger, Gregoire, and Kong 2000).

The SLICE= option produces a table titled "Tests of Effect Slices." The table name is "Slices."

STEPDOWN< (*step-down-options*) >
performs a further step-down adjustment to the multiple-comparison-adjusted *p*-values for differences of LS-means. Specifically, the ADJUST= option in the LSMEANS statement performs a multiple comparison adjustment for the *p*-values and confidence limits that are associated with differences of LS-means. When you specify the STEPDOWN option, PROC LMIXED applies an additional adjustment to these multiple-comparison-adjusted *p*-values. Step-down methods increase the power of multiple comparisons by taking advantage of the fact that a *p*-value will never be declared significant unless all smaller *p*-values are also declared significant.

You must specify the ADJUST= option when you specify the STEPDOWN option in the LSMEANS statement.

The STEPDOWN option affects only *p*-values, not confidence limits.

You can specify the following *step-down-options* in parentheses:

MAXTIME=*n*
specifies an upper limit of *n* seconds to compute the maximal logically consistent sequential subsets of equality hypotheses when you specify the TYPE=LOGICAL option. The default is 60 seconds. Step-down adjustments require substantial computation time when you have a large number of LS-means comparisons, such as all pairwise comparisons when the number of groups exceeds 10. In such cases, it is recommended that you use the TYPE=FREE or TYPE=LOGICAL(*n*) option with a small value of *n*.

REPORT
displays the report of the step-down adjustment, including a list of the sequential subsets (Westfall 1997). When you specify the STEPDOWN option along with the ADJUST=SIMULATE option, the report also includes the step-down simulation results.

TYPE=LOGICAL< (*n*) > | FREE
specifies the type of step-down adjustment method. By default, TYPE=FREE. You can specify the following method types:

TYPE=FREE performs step-down adjustments in which sequential subsets are computed without considering the logical constraints.

TYPE=LOGICAL(*n*) performs step-down adjustments on the basis of maximal logically consistent sequential subsets of equality hypotheses (Shaffer 1986; Westfall 1997). Because considering all logical constraints can be computationally intensive, you can limit the depth of the search tree by specifying a nonnegative integer *n*. If you omit *n* or if $n = -1$, the full search tree is used.

The TYPE=FREE option results are more conservative than the TYPE=LOGICAL option results, but they are generally more efficient in computing step-down adjustments when you have a large number of comparisons. For example, it is not feasible to take logical constraints between all pairwise comparisons when the number of groups exceeds 10. As with TYPE=FREE, results for TYPE=LOGICAL(*n*) are more conservative than the true TYPE=LOGICAL results, but even for TYPE=LOGICAL(0) they can be appreciably less conservative than the TYPE=FREE results, and they are computationally feasible for much larger numbers of comparisons.

When you specify the STEPDOWN option along with the ADJUST=BON option, the method of Holm (1979) is used if TYPE=FREE, and "Method 2" of Shaffer (1986) is used if TYPE=LOGICAL.

When you specify the STEPDOWN option along with the ADJUST=SIMULATE option, the method of Westfall (1997) is used, where the generalized least squares hybrid approach is used to increase Monte Carlo accuracy.

LSMESTIMATE Statement

LSMESTIMATE *fixed-effect* < '*label*' > *values* < *divisor*=*n* >
< , < '*label*' > *values* < *divisor*=*n* > > < , . . . >
< / *options* > ;

The LSMESTIMATE statement provides a mechanism for performing custom hypothesis tests among the least squares means (LS-means). In contrast to the hypotheses that are tested using the ESTIMATE or CONTRAST statement, the LSMESTIMATE statement enables you to form linear combinations of the LS-means rather than linear combination of fixed-effects parameter estimates and/or random-effects solutions. Multiple-row sets of coefficients are permitted.

Computing an LS-means estimate involves two coefficient matrices. Suppose that the *fixed-effect* has n_l levels. Then the LS-means are formed as $\mathbf{L}_1\widehat{\boldsymbol{\beta}}$, where $\mathbf{L}_1$ is an $(n_l \times p)$ coefficient matrix. The $(k \times n_l)$ coefficient matrix $\mathbf{K}$ is formed from the *values* that you supply in the k rows of the LSMESTIMATE statement. The LS-means estimates then represent the $(k \times 1)$ vector

$$\mathbf{K}\mathbf{L}_1\boldsymbol{\beta} = \mathbf{L}\boldsymbol{\beta}$$

You can use the ELSM option to display the coefficient matrix $\mathbf{K}$ and use the E option to display the coefficient matrix $\mathbf{L}$.

The LSMESTIMATE statement produces a t test for each row of coefficients that you specify. You can adjust p-values and confidence intervals for multiplicity by specifying the ADJUST= option. You can obtain an F test of single-row or multiple-row LS-means estimates by specifying the FTEST option.

Note that you specify only a single fixed effect in the LSMESTIMATE statement. The row labels are optional and follow the effects specification. For example, the following statements fit a split-plot design and compare the average of the third and fourth LS-mean of the whole-plot factor A to the first LS-mean of the factor:

```
proc lmixed;
   class a b block;
   model y = a b a*b /  s;
   random int a / sub=block;
   lsmestimate A 'a1 vs avg(a3,a4)' [2, 1] [-1, 3] [-1, 4] divisor=2;
```

```
run;
```

The order in which coefficients are assigned to the LS-means corresponds to the order in which they are displayed in the "Least Squares Means" table. You can use the ELSM option to see how coefficients are matched to levels of the *fixed-effect*.

The *DIVISOR=n* specification enables you to assign a separate divisor to each row of the LS-means estimate. You can also use the DIVISOR= option in this statement after a slash (/).

Table 17.7 summarizes the *options* available in the LSMESTIMATE statement.

Table 17.7 LSMESTIMATE Statement Options

Option	Description
Construction and Computation of LS-Means	
AT	Modifies covariate values in computing LS-means
DIVISOR=	Specifies a list of values to divide the coefficients
SINGULAR=	Tunes the singular criterion for estimability checking
Tests, Adjustment, and *p*-Values	
ADJUST=	Determines the method of multiple comparison adjustment of LS-means estimate differences
ALPHA=α	Determines the confidence level $(1-\alpha)$
CHISQ	Requests a chi-square test in addition to the F test
DF=	Assigns a specific value to degrees of freedom for tests and confidence limits
FTEST	Produces an F test
LOWER	Performs one-sided, lower-tailed inference
UPPER	Performs one-sided, upper-tailed inference
Statistical Output	
CL	Constructs confidence limits for means and mean differences
CORR	Displays the correlation matrix of LS-means estimates
COV	Displays the covariance matrix of LS-means estimates
E	Displays the **L** matrix
ELSM	Displays the **K** matrix

You can specify the following *options* in the LSMESTIMATE statement after a slash (/):

ADJUST=*method*

performs a multiple comparison adjustment for the p-values and confidence limits for the differences of LS-means estimates. The adjusted quantities are produced in addition to the unadjusted p-values and confidence limits. Adjusted confidence limits are produced if you specify the CL or ALPHA= option. For a description of the adjustments, see the ADJUST= option in the LSMEANS statement.

You can specify the following *methods*:

BON

performs Bonferroni t tests of differences between LS-means. The method involves correction factors described in Chapter 53, "The GLM Procedure" (*SAS/STAT User's Guide*), and Chapter 86, "The

MULTTEST Procedure" (*SAS/STAT User's Guide*); also see Westfall and Young (1993) and Westfall et al. (1999).

SCHEFFE performs Scheffé's multiple comparison procedure.

SIDAK performs pairwise t tests on differences between LS-means with levels adjusted according to Šidák's inequality. The method involves correction factors described in Chapter 53, "The GLM Procedure" (*SAS/STAT User's Guide*), and Chapter 86, "The MULTTEST Procedure" (*SAS/STAT User's Guide*); also see Westfall and Young (1993) and Westfall et al. (1999).

SIMULATE<(*simoptions***)>** performs the simulation-based multiple comparison procedure. This method computes adjusted p-values and confidence limits from the simulated distribution of the maximum or maximum absolute value of a multivariate t random vector. All covariance parameters, except the residual scale parameter, are fixed at their estimated values throughout the simulation, potentially resulting in some underdispersion. The simulation estimates q, the true $(1-\alpha)$ quantile, where $1-\alpha$ is the confidence coefficient. For a description of the *simoptions* (except for the CVADJUST option, which you cannot use in the MARGINS statement), see the ADJUST= option in the LSMEANS statement.

ALPHA=*number*

constructs a t-type confidence interval for each of the LS-means with confidence level 1 – *number*. The value of *number* must be between 0 and 1; the default is 0.05.

AT *at-specification*

enables you to modify the values of the covariates that are used in computing LS-means. By default, all covariate effects are set equal to their mean values for computing standard LS-means. The AT option enables you to assign arbitrary values to the covariates. Additional columns in the output table indicate the values of the covariates.

You can specify the following *at-specifications*:

MEANS sets covariates equal to their mean values (as with standard LS-means) and applies this adjustment to crossproducts of covariates.

*variable***=***value* sets the covariate *variable* equal to *value*.

(*variable-list***)=(***value-list***)** sets covariates in the *variable-list* equal to values in the *value-list*.

As an example, consider the following invocation of PROC LMIXED:

```
proc lmixed;
   class A;
   model Y = A x1 x2 x1*x2;
   lsmestimate A 'a1 vs a2' [1, 1] [-1, 2];
   lsmestimate A 'a1 vs a2' [1, 1] [-1, 2] / at means;
   lsmestimate A 'a1 vs a2' [1, 1] [-1, 2] / at x1=1.2;
   lsmestimate A 'a1 vs a2' [1, 1] [-1, 2] / at (x1 x2)=(1.2 0.3);
run;
```

For the first two LSMESTIMATE statements, the LS-means coefficient of x1 is $\overline{x}_1$ (the mean of x1) and of x2 is $\overline{x}_2$ (the mean of x2). For the first LSMESTIMATE statement, the coefficient of x1*x2 is $\overline{x_1 x_2}$. However, for the second LSMESTIMATE statement, the coefficient is $\overline{x}_1 \times \overline{x}_2$. The third LSMESTIMATE statement sets the coefficient of x1 equal to 1.2 and leaves it at $\overline{x}_2$ for x2, and the final LSMESTIMATE statement sets these values to 1.2 and 0.3, respectively.

Even if you specify a WEIGHT variable, the unweighted covariate means are used for the covariate coefficients if you omit the AT option. If you specify the AT option, WEIGHT or FREQ variables are taken into account as follows. The weighted covariate means are used for the covariate coefficients for which no explicit AT option values are given, or if you specify the AT MEANS option. Observations that do not contribute to the analysis because of a missing dependent variable are used in computing the covariate means. You should use the E option in conjunction with the AT option to check that the modified LS-means coefficients are the ones that you want.

CHISQ
performs chi-square tests in addition to *F* tests, when you specify the FTEST option.

CL
constructs the *t*-type confidence limits for each of the LS-means. If DDFM=NONE, then PROC LMIXED uses infinite degrees of freedom for this test, essentially computing a *z* interval. The confidence level is 95% by default, corresponding to the default ALPHA=0.05 option. You can change this by specifying the ALPHA= option.

CORR
displays the estimated correlation matrix of the linear combination of the LS-means.

COV
displays the estimated covariance matrix of the linear combination of the LS-means.

DF=*number*
specifies the degrees of freedom for the *t* test and confidence limits. The default is the residual degrees of freedom that you define by specifying the DDFM=RESIDUAL option.

DIVISOR=*value-list*
specifies a list of values by which to divide the coefficients so that fractional coefficients can be entered as integer numerators. If you do not specify a *value-list*, a default value of 1.0 is assumed. Missing values in the *value-list* are converted to 1.0.

If the number of elements in the *value-list* exceeds the number of rows of the estimate, the extra values are ignored. If the number of elements in the *value-list* is less than the number of rows of the estimate, the last value in the *value-list* is carried forward.

If you specify a row-specific divisor as part of the specification of the estimate row, this value multiplies the corresponding value in the *value-list*. For example, the following statement divides the coefficients in the first row by 8 and the coefficients in the third and fourth rows by 3:

```
lsmestimate A 'One vs. two'   [8, 1] [-8, 2] divisor=2,
              'One vs. three' [1, 1] [-1, 3]          ,
              'One vs. four'  [3, 1] [-3, 4]          ,
              'One vs. five'  [3, 1] [-3, 5]           / divisor=4,.,3;
```

Coefficients in the second row are not altered.

E

displays the **L** coefficients of the estimable function. These are the coefficients that apply to the fixed-effects parameter estimates. The E option displays the coefficients that you would need to enter in an equivalent ESTIMATE statement.

ELSM

displays the **K** matrix coefficients. These are the coefficients that apply to the LS-means. This option is useful to ensure that you assigned the coefficients correctly to the LS-means.

FTEST

JOINT

produces an *F* test that jointly tests the rows of the LSMESTIMATE against zero. If you specify the LOWER or UPPER options, the LMIXED procedure computes a simulation-based *p*-value for the constrained joint test.

LOWER

LOWERTAILED

requests that the *p*-value for the *t* test be based only on values that are less than the test statistic. A two-tailed test is the default. A lower-tailed confidence limit is also produced if you specify the CL or ALPHA= option.

Note that for ADJUST=SCHEFFE the one-sided adjusted confidence intervals and one-sided adjusted *p*-values are the same as the corresponding two-sided statistics, because this adjustment is based on only the right tail of the *F* distribution.

If you specify the FTEST option, then a one-sided left-tailed order restriction is applied to all estimable functions.

SINGULAR=*number*

tunes the singular criterion for estimability checking.

UPPER

UPPERTAILED

requests that the *p*-value for the *t* test be based only on values that are greater than the test statistic. A two-tailed test is the default. An upper-tailed confidence limit is also produced if you specify the CL or ALPHA= option.

Note that for ADJUST=SCHEFFE the one-sided adjusted confidence intervals and one-sided adjusted *p*-values are the same as the corresponding two-sided statistics, because this adjustment is based on only the right tail of the *F* distribution.

If you specify the FTEST option, then a one-sided right-tailed order restriction is applied to all estimable functions.

MARGINS Statement

MARGINS *fixed-effects* < / *options* > **;**

The MARGINS statement computes predictive margins of fixed effects. You can compute predictive margins for any effect in the MODEL statement that involves only classification variables.

The predictive margin for a specific level (group) of a fixed effect represents the average predicted response if all the observations in the data set were in that group (Lane and Nelder 1982; Chang, Gelman, and Pagano 1982). You compute the predictive margin by fixing this effect at the specified level for all observations and averaging the predicted responses. The standard error of the predictive margin is computed by using the delta method. The weighted average predicted response is the weighted predictive margins.

For example, consider the simple regression $y_{ij} = \alpha_i + \beta x_{ij}$, where α_i is the effect of the ith level of the classification effect α, $i = 1, \ldots, R$, and x_{ij} is the covariate for the jth observation in the ith level of the effect α, $j = 1, \ldots, n_i$. Then the predictive margin for level r of effect α is

$$\frac{1}{N}\left(\sum_{i=1}^{R}\sum_{j=1}^{n_i}\hat{\alpha}_r + \hat{\beta}x_{ij}\right)$$

where $N = \sum_{i=1}^{R} n_i$.

For a more complex example, consider the following linear model with an interaction effect:

```
data mylib.bal;
input sex$ group rep y yname$;
cards;
boy    2    1   12  y1
boy    2    2    8  y2
boy    3    1   14  y3
boy    3    2    9  y4
girl   1    1   10  y5
girl   1    2   23  y6
girl   2    1   14  y7
girl   2    2   13  y8
girl   3    1   13  y9
girl   3    2   12  y10
;

proc lmixed data=mylib.bal;
   class sex group;
   model y = sex group sex*group;
   margins sex;
run;
```

The model fits the data that have ten observations: four of them are for the variable boy, and six of them are for the variable girl. Notice that group 1 has no data for boy. The predictive margin for boy is the average of the predicted values of these ten response variables, assuming that all responses are observed from boys. However, you cannot compute the predicted values for y5 and y6 for boy because y5 and y6 are in group 1, which is a group level that is absent from the data for boys. Thus, the predictive margin for boy is not predictable. This nonpredictability is caused by the lack of estimated parameters for boy interacting with group 1, which is due to the missing information in the interaction effect.

You can specify multiple effects in one or multiple MARGINS statements. Predictive margin computations are not supported for constructed effects.

PROC LMIXED constructs an approximate t test to determine whether the null hypothesis that the associated population quantity equals zero is valid. By default, the denominator degrees of freedom for this test are the same as those displayed for the effect in the "Type III Tests of Fixed Effects" table.

Table 17.8 summarizes the options available in the MARGINS statement. All these options are discussed in alphabetical order after the table.

Table 17.8 MARGINS Statement Options

Option	Description
Construction and Computation of Predictive Margins	
AT	Modifies the covariate value in computing predictive margins
DIFF	Computes differences of predictive margins
SLICEBY=	Partitions F tests
SLICEDIFF	Computes differences of sliced predictive margins and determines the types of differences
WEIGHT	Computes the weighted predictive margins
Degrees of Freedom and *p*-Values	
ADJUST=	Specifies the method of multiple comparison adjustment of predictive margin differences
ALPHA=α	Specifies the confidence level $(1 - \alpha)$
DF=	Assigns specific value to degrees of freedom for tests and confidence limits
Statistical Output	
CL	Constructs confidence limits for predictive margins and/or predictive margin differences

You can specify the following *options* in the MARGINS statement after a slash (/).

ADJUST=*method*

performs a multiple comparison adjustment of the p-values and confidence limits for the differences of predictive margins. The adjusted quantities are produced in addition to the unadjusted quantities. By default, PROC LMIXED performs all pairwise differences. If you specify ADJUST=DUNNETT, the procedure analyzes all differences with a control level. The ADJUST= option does not imply the DIFF option.

Adjusted confidence limits are produced if you specify the CL or ALPHA=α option. For a description of the adjustments, see the ADJUST= option in the LSMEANS statement.

You can specify the following *methods*:

BON performs Bonferroni t tests of differences between margins. This method involves correction factors described in Chapter 53, "The GLM Procedure" (*SAS/STAT User's Guide*), and Chapter 86, "The MULTTEST Procedure" (*SAS/STAT User's Guide*); also see Westfall and Young (1993) and Westfall et al. (1999).

DUNNETT performs Dunnett's *t* test, which determines whether any treatments are significantly different from a single control for the effects in the MARGINS statement.

SCHEFFE performs Scheffé's multiple comparison procedure.

SIDAK performs pairwise *t* tests on differences between LS-means with levels adjusted according to Šidák's inequality. This method involves correction factors described in Chapter 53, "The GLM Procedure" (*SAS/STAT User's Guide*), and Chapter 86, "The MULTTEST Procedure" (*SAS/STAT User's Guide*); also see Westfall and Young (1993) and Westfall et al. (1999).

SIMULATE<(*simoptions***)>** performs the simulation-based multiple comparison procedure. This method computes adjusted *p*-values and confidence limits from the simulated distribution of the maximum or maximum absolute value of a multivariate *t* random vector. All covariance parameters, except the residual scale parameter, are fixed at their estimated values throughout the simulation, potentially resulting in some underdispersion. The simulation estimates q, the true $(1-\alpha)$ quantile, where $1-\alpha$ is the confidence coefficient. For a description of the *simoptions* (except for the CVADJUST option, which you cannot use in the MARGINS statement), see the ADJUST= option in the LSMEANS statement.

TUKEY performs Tukey's studentized range test on the predictive margins.

ALPHA=*number*

constructs a *t*-type confidence interval for each of the predictive margins with confidence level is $1-$ *number*. The value of *number* must be between 0 and 1, exclusive; by default, ALPHA=0.05.

AT *at-specification*

enables you to modify the values of the covariates that are used in computing predictive margins. By default, all covariate effects are set equal to their mean values for computing standard LS-means. The AT option enables you to assign arbitrary values to the covariates. Additional columns in the output table indicate the values of the covariates.

You can specify the following *at-specifications*:

MEANS sets covariates equal to their mean values (as with standard LS-means) and applies this adjustment to crossproducts of covariates.

*variable***=***value* sets the covariate *variable* equal to *value*.

(*variable-list***)=(***value-list***)** sets covariates in the *variable-list* equal to values in the *value-list*.

As an example, consider the following invocation of PROC LMIXED:

```
proc lmixed;
   class A;
   model Y = A x1 x2 x1*x2;
   margins A;
   margins A / at means;
   margins A / at x1=1.2;
```

```
   margins A / at (x1 x2)=(1.2 0.3);
run;
```

The first MARGINS statement sets x1 and x2 to their observed values for each observation. For the second MARGINS statement, x1 is set to $\overline{x}_1$ (the mean of x1) and x2 is set to $\overline{x}_2$ (the mean of x2). Their interaction is set to $\overline{x}_1 \times \overline{x}_2$. The third MARGINS statement sets x1 to 1.2, and the final MARGINS statement sets these variables to 1.2 and 0.3, respectively.

Even if you specify a weight variable in the WEIGHT statement, the unweighted covariate means are used for the covariate coefficients if you omit the AT option. If you specify the AT option, the weight variable in the WEIGHT statement or the frequency variable in the FREQ statement is taken into account as follows. The weighted covariate means are used for the covariate coefficients for which no explicit AT option values are given, or if you specify the AT MEANS option. Observations that do not contribute to the analysis because of a missing dependent variable are used in computing the covariate means.

CL

constructs *t*-type confidence limits for each of the predictive margins. If DDFM=NONE, then PROC LMIXED uses infinite degrees of freedom for this test, essentially computing a z interval. The confidence level is 0.95 by default; you can change this by specifying the ALPHA=α option. If you specify an ADJUST= option, then the confidence limits are adjusted for multiplicity.

DF=*number*

specifies the degrees of freedom for the *t* test and confidence limits. The default is the denominator degrees of freedom from the "Type III Tests of Fixed Effects" table corresponding to the margins effect.

DIFF<=*difftype***>**

PDIFF<=*difftype***>**

displays differences of the predictive margins. The optional *difftype* specifies which differences to produce; the *difftype* values are ALL, CONTROL, CONTROLL, and CONTROLU. The ALL *difftype* displays all pairwise differences; it is the default. The CONTROL *difftype* displays differences between each margin and a control, which, by default, is the first level of each effect that you specify in the MARGINS statement.

To specify which levels of the effects are the controls, list the quoted formatted values in parentheses after the CONTROL keyword. For example, for the classification variables A, B, and C, each having two levels, 1 and 2, the following MARGINS statement specifies the (1,2) level of A*B and the (2,1) level of B*C as controls:

```
margins A*B B*C / diff=control('1' '2', '2' '1');
```

For an effect that has multiple classification variables, the quoted formatted values are assigned to these variables in the order in which they appear in the CLASS statement.

Two-tailed tests and confidence limits are associated with the CONTROL *difftype*. For one-tailed results, use either the CONTROLL or CONTROLU *difftype*. The CONTROLL *difftype* tests whether the noncontrol levels are significantly smaller than the control level; the upper confidence limits for the control level minus the noncontrol levels are considered to be infinity and are displayed as missing. Conversely, the CONTROLU *difftype* tests whether the noncontrol levels are significantly larger than

the control level; the upper confidence limits for the noncontrol levels minus the control level are considered to be infinity and are displayed as missing.

If you want to perform multiple comparison adjustments on the differences of predictive margins, you must specify the ADJUST= option.

SLICEBY=*fixed-effect* | **(***fixed-effects***)**

SLICE=*fixed-effect* | **(***fixed-effects***)**

specifies effects by which to partition interaction effects that you specify in the MARGINS statement. This option produces tests of margins for each level of the specified slice effect.

For example, suppose that the effect specified in the MARGINS statement is A*B. To test the margins of A*B for each level of A, you can specify the following statement:

```
margins A*B / sliceby=A;
```

The SLICEBY= option produces an F test that performs a test of the simultaneous equality of the margins at a fixed level of the slice effect A. You can compute differences of the margins while holding one or more factors at a fixed level by specifying the SLICEDIFF option.

SLICEDIFF< =*difftype* **>**

constructs the specified type of differences for the sliced margins and tests them against zero. The *difftype* values are ALL, CONTROL, CONTROLL, and CONTROLU. The ALL *difftype* computes all pairwise differences; it is the default. The CONTROL, CONTROLL, and CONTROLU *difftypes* compute the differences between each margin and a control. Whereas the SLICEBY= option tests the simultaneous equality of the margins at a fixed level of the slice effect, the SLICEDIFF option tests pairwise differences of these margins. This enables you to perform multiple comparisons among the levels of one factor at a fixed level of the other factor.

For example, assume that, in a certain design, factors A and B have $a = 4$ and $b = 3$ levels, respectively. Consider the following statements:

```
proc lmixed;
   class A B;
   model y = A B A*B;
   margins A*B / sliceby=A;
   margins A*B / sliceby=A slicediff=all;
run;
```

The first MARGINS statement produces four F tests, one per level of A. Denote the three margins that correspond to the first level of A as $\mathbf{m}_{a1}^{(1)}$, $\mathbf{m}_{a1}^{(2)}$, and $\mathbf{m}_{a1}^{(3)}$. Then the first F test determines whether the two-degrees-of-freedom hypothesis is valid:

$$H: \begin{cases} \mathbf{m}_{a1}^{(1)} - \mathbf{m}_{a1}^{(2)} = 0 \\ \mathbf{m}_{a1}^{(1)} - \mathbf{m}_{a1}^{(3)} = 0 \end{cases}$$

The SLICEDIFF option performs tests of the differences between all pairs of these three margins. In the example, this corresponds to tests of the following form:

$$H: \mathbf{m}_{a1}^{(1)} - \mathbf{m}_{a1}^{(2)} = 0$$
$$H: \mathbf{m}_{a1}^{(1)} - \mathbf{m}_{a1}^{(3)} = 0$$
$$H: \mathbf{m}_{a1}^{(2)} - \mathbf{m}_{a1}^{(3)} = 0$$

In the example, where $a = 4$ and $b = 3$, the second MARGINS statement produces four sets of predictive margin differences. Within each set, factor A is held fixed at a particular level, and each set consists of three comparisons.

For differences between each margin and a control, the default control is the first level of each effect that you specify in the MARGINS statement. To specify which levels of the effects are the controls, list the quoted formatted values in parentheses after the keyword CONTROL.

For example, if the effects A, B, and C are classification variables, each having three levels (1, 2, and 3), the following MARGINS statement specifies the (1,3) level of A*B as the control:

```
margins A*B / sliceby=(A B)
              slicediff=control('1' '3');
```

This MARGINS statement first produces predictive margin differences while holding the levels of A fixed, and then it produces predictive margin differences while holding the levels of B fixed. In the former case, level '3' of B serves as the control level. In the latter case, level '1' of A serves as the control.

For an effect that has multiple classification variables, the quoted formatted values are assigned to these variables in the order in which they appear in the CLASS statement.

Two-tailed tests and confidence limits are associated with the CONTROL *difftype*. For one-tailed results, use either the CONTROLL or CONTROLU *difftype*. The CONTROLL *difftype* tests whether the noncontrol levels are significantly smaller than the control level; the upper confidence limits for the control level minus the noncontrol levels are considered to be infinity and are displayed as missing. Conversely, the CONTROLU *difftype* tests whether the noncontrol levels are significantly larger than the control level; the upper confidence limits for the noncontrol levels minus the control level are considered to be infinity and are displayed as missing.

When you specify the ADJUST= option, the LMIXED procedure also adjusts the tests for multiplicity. The adjustment is based on the number of comparisons within each level of the SLICEBY= effect.

WEIGHT

computes the weighted predictive margins. The weight values are provided by the WEIGHT statement. By default, the unweighted predictive margins are computed even if you specify the WEIGHT statement.

MODEL Statement

MODEL *dependent* **=** < *fixed-effects* >< / *options* > **;**

The MODEL statement names a single dependent variable and the fixed effects, which determine the **X** matrix of the mixed model. The MODEL statement is required.

An intercept is included in the fixed-effects model by default. If no fixed effects are specified, only this intercept term is fit. You can remove the intercept by using the NOINT option. You cannot explicitly specify an intercept by using the keyword INTERCEPT as an effect in MODEL statement.

Table 17.9 summarizes options in the MODEL statement. These are subsequently discussed in detail in alphabetical order.

Table 17.9 Summary of Important MODEL Statement Options

Option	Description
Model Building	
NOINT	Excludes the fixed-effect intercept from the model
Statistical Computations	
ALPHA=α	Determines the confidence level $(1-\alpha)$ for fixed effects
DDF=	Specifies the denominator degrees of freedom
DDFM=	Specifies the method for computing the denominator degrees of freedom
Statistical Output	
CL	Displays confidence limits for fixed-effects parameter estimates
DENSESOL	Displays dense-type fixed-effects parameter estimates
SOLUTION	Displays fixed-effects parameter estimates

You can specify the following *options* in the MODEL statement after a slash (/):

ALPHA=α

sets the confidence level to be $1-\alpha$ for each confidence interval of the fixed-effects parameters, where α must be a number between 0 and 1. By default, ALPHA=0.05.

CL

constructs that t-type confidence limits for each of the fixed-effects parameter estimates. The confidence level is 0.95 by default; this can be changed by specifying the ALPHA= option.

DDF=*value-list*

enables you to specify your own denominator degrees of freedom for the fixed effects. The *value-list* is a list of numbers or missing values (.) separated by commas. The degrees of freedom should be listed in the order in which the effects appear in the MODEL statement. If you want to retain the default degrees of freedom for a particular effect, use a missing value for its location in the list. For example, the following statement assigns 3 denominator degrees of freedom to A and 4.7 to A*B, while those for B remain the same:

```
model Y = A B A*B / ddf=3,.,4.7;
```

DDFM=CONTAIN | NONE | RESIDUAL

specifies the method of computing the denominator degrees of freedom for the tests of fixed effects. You can specify the following methods:

CONTAIN invokes the *containment method* to compute denominator degrees of freedom (DDF). The containment method is carried out as follows: Denote the fixed effect in question A, and search the effect list that you specify in the RANDOM statement for the effects that *syntactically* contain A. For example, the random effect B(A) contains A, but the random effect C does not, even if it has the same levels as B(A).

Among the random effects that contain A, compute their rank contribution to the $(\mathbf{X}\ \mathbf{Z})$ matrix. The DDF that is assigned to A is the smallest of these rank contributions. If no effects are found, the DDF for A is set equal to the residual degrees of freedom, $N - \mathrm{rank}(\mathbf{X}\ \mathbf{Z})$. This choice of DDF matches the tests that are performed for balanced split-plot designs and should be adequate for moderately unbalanced designs.

The DDFM=CONTAIN option is supported only when DMMETHOD=DENSE.

CAUTION: If you have a $\mathbf{Z}$ matrix that contains a large number of columns, the overall memory requirements and the computing time after convergence can be substantial for the containment method. If they are too great, you might want to use the DDFM=RESIDUAL option instead.

NONE does not apply any denominator degrees of freedom. PROC LMIXED then essentially assumes that infinite degrees of freedom are available in the calculation of *p*-values. The *p*-values for *t* tests are then identical to *p*-values that are derived from the standard normal distribution.

RESIDUAL performs all tests by using the residual degrees of freedom, $n - \mathrm{rank}(\mathbf{X})$, where n is the number of observations that are used.

By default, DDFM=RESIDUAL.

DENSESOL

produces the solutions for the fixed-effects parameters that match the solutions when you specify DMMETHOD=DENSE in the PROC LMIXED statement. The DENSESOL option changes only the ODS table ParameterEstimates.

Note that this option consumes extra computer memory. The DENSESOL option is valid only when DMMETHOD=SPARSE. It is designed specifically for users who need a solution for the fixed-effects parameters to match the solutions when DMMETHOD=DENSE in the PROC LMIXED statement.

NOINT

requests that no intercept be included in the model. (An intercept is included by default.)

SOLUTION

S

produces a solution for the fixed-effects parameters. Using notation from the section "Linear Mixed Models Theory" on page 935, the fixed-effects parameter estimates are $\widehat{\boldsymbol{\beta}}$ and their approximate standard errors are the square roots of the diagonal elements of $(\mathbf{X}'\widehat{\mathbf{V}}^{-1}\mathbf{X})^{-}$.

Along with the estimates and their approximate standard errors, a t statistic is computed as the estimate divided by its standard error. The Pr > |t| column contains the two-tailed p-value that corresponds to the t statistic and associated degrees of freedom. You can use the CL option to request confidence intervals for all of the parameters; they are constructed around the estimate by using a radius that is the product of the standard error times a percentage point from the t distribution.

OPTIMIZATION Statement

OPTIMIZATION < *options* > **;**

The OPTIMIZATION statement specifies the technique and relevant specifications that are used in nonlinear optimization of REML and ML functions.

You can specify the following *options*. The effect of some *options* depends on the technique that is specified in the TECHNIQUE= option.

ABSCONV=*r*

specifies an absolute function convergence criterion. For minimization, termination requires $f(\boldsymbol{\psi}^{(k)}) \leq r$, where $\boldsymbol{\psi}$ is the vector of parameters in the optimization and $f(\cdot)$ is the objective function. The default value of r is the negative square root of the largest double-precision value, which serves only as a protection against overflows.

ABSFCONV=*r*

specifies an absolute function difference convergence criterion. For all techniques except Nelder-Mead simplex (TECHNIQUE=NMSIMP), termination requires a small change of the function value in successive iterations:

$$|f(\boldsymbol{\psi}^{(k-1)}) - f(\boldsymbol{\psi}^{(k)})| \leq r$$

Here, $\boldsymbol{\psi}$ denotes the vector of parameters that participate in the optimization and $f(\cdot)$ is the objective function. The same formula is used for the Nelder-Mead simplex technique, but $\boldsymbol{\psi}^{(k)}$ is defined as the vertex that has the lowest function value and $\boldsymbol{\psi}^{(k-1)}$ is defined as the vertex that has the highest function value in the simplex. By default, ABSFCONV=0.

ABSGCONV=*r*

specifies an absolute gradient convergence criterion. Termination requires the maximum absolute gradient element to be small:

$$\max_j |g_j(\boldsymbol{\psi}^{(k)})| \leq r$$

Here, $\boldsymbol{\psi}$ denotes the vector of parameters that participate in the optimization and $g_j(\cdot)$ is the gradient of the objective function with respect to the j parameter. This criterion is not used by the Nelder-Mead simplex technique. By default, ABSGCONV=1E–5.

FCONV=*r*

specifies a relative function convergence criterion. For all techniques except the Nelder-Mead simplex technique, termination requires a small relative change of the function value in successive iterations:

$$\frac{|f(\boldsymbol{\psi}^{(k)}) - f(\boldsymbol{\psi}^{(k-1)})|}{|f(\boldsymbol{\psi}^{(k-1)})|} \leq r$$

Here, $\boldsymbol{\psi}$ denotes the vector of parameters that participate in the optimization and $f(\cdot)$ is the objective function. The same formula is used for the Nelder-Mead simplex technique, but $\boldsymbol{\psi}^{(k)}$ is defined as the vertex that has the lowest function value and $\boldsymbol{\psi}^{(k-1)}$ is defined as the vertex that has the highest function value in the simplex.

The default is $r = 10^{-\mathrm{FDIGITS}}$, where FDIGITS is $-\log_{10}(\epsilon)$ and ϵ is the machine precision.

FCONV2=*r*

specifies a second function convergence criterion. For all techniques except the Nelder-Mead simplex technique, termination requires a small predicted reduction of the objective function:

$$df^{(k)} \approx f(\boldsymbol{\beta}^{(k)}) - f(\boldsymbol{\beta}^{(k)} + \mathbf{s}^{(k)})$$

The predicted reduction

$$\begin{aligned} df^{(k)} &= -\mathbf{g}^{(k)\prime}\mathbf{s}^{(k)} - \frac{1}{2}\mathbf{s}^{(k)\prime}\mathbf{H}^{(k)}\mathbf{s}^{(k)} \\ &= -\frac{1}{2}\mathbf{s}^{(k)\prime}\mathbf{g}^{(k)} \leq r \end{aligned}$$

is computed by approximating the objective function f by the first two terms of the Taylor series and substituting the Newton step,

$$\mathbf{s}^{(k)} = -[\mathbf{H}^{(k)}]^{-1}\mathbf{g}^{(k)}$$

For the Nelder-Mead simplex technique, termination requires a small standard deviation of the function values of the $p + 1$ simplex vertices $\boldsymbol{\beta}_l^{(k)}$, $l = 0, \ldots, p$,

$$\sqrt{\frac{1}{n+1}\sum_l \left[f(\boldsymbol{\beta}_l^{(k)}) - \overline{f}(\boldsymbol{\beta}^{(k)})\right]^2} \leq r$$

where $\overline{f}(\boldsymbol{\beta}^{(k)}) = \frac{1}{p+1}\sum_l f(\boldsymbol{\beta}_l^{(k)})$. If there are p_{act} boundary constraints active at $\boldsymbol{\beta}^{(k)}$, the mean and standard deviation are computed only for the $n + 1 - p_{act}$ unconstrained vertices.

The default value is *r* = 1E–6 for the Nelder-Mead simplex technique and *r*= 0 otherwise.

GCONV=*r*

specifies a relative gradient convergence criterion. For all techniques except the conjugate-gradient technique and the Nelder-Mead simplex technique, termination requires that the normalized predicted function reduction be small,

$$\frac{\mathbf{g}(\boldsymbol{\psi}^{(k)})'[\mathbf{H}^{(k)}]^{-1}\mathbf{g}(\boldsymbol{\psi}^{(k)})}{|f(\boldsymbol{\psi}^{(k)})|} \leq r$$

Here, $\boldsymbol{\psi}$ denotes the vector of parameters that participate in the optimization, $f(\cdot)$ is the objective function, and $\mathbf{g}(\cdot)$ is the gradient. For the conjugate-gradient technique (where a reliable Hessian estimate $\mathbf{H}$ is not available), the following criterion is used:

$$\frac{\| \mathrm{g}(\boldsymbol{\psi}^{(k)}) \|_2^2 \quad \| \mathrm{s}(\boldsymbol{\psi}^{(k)}) \|_2}{\| \mathrm{g}(\boldsymbol{\psi}^{(k)}) - \mathrm{g}(\boldsymbol{\psi}^{(k-1)}) \|_2 \, |f(\boldsymbol{\psi}^{(k)})|} \leq r$$

This criterion is not used by the Nelder-Mead simplex technique. The default value is *r*=1E–8.

GCONV2=*r*

specifies another relative gradient convergence criterion. For Newton-Raphson with ridging and Newton-Raphson techniques, the following criterion of Browne (1982) is used:

$$\max_j \frac{|\mathrm{g}_j(\boldsymbol{\beta}^{(k)})|}{\sqrt{f(\boldsymbol{\beta}^{(k)})\mathbf{H}^{(k)}_{j,j}}} \leq r$$

This criterion is not used by the other techniques. By default, GCONV2=0.

HESSTYPE=BFGS | FDFULL | FULL

specifies the type of Hessian to use in the optimization. You can specify the following values:

BFGS uses the original Broyden, Fletcher, Goldfarb, and Shanno (BFGS) approximation of the inverse Hessian matrix.

FDFULL uses the finite-difference Hessian.

FULL uses the exact Hessian or average information Hessian, assuming that it is available.

The default value is HESSTYPE=BFGS when there is no Hessian function available in the optimization.

This option is supported only when TECHNIQUE=ACTIVESET or IPDIRECT.

MAXFUNC=*n*

specifies the maximum number *n* of function calls in the optimization process. The optimization can terminate only after completing a full iteration. Therefore, the number of function calls that are performed can exceed *n*. The default values are as follows, depending on the optimization technique:

- When TECHNIQUE=TRUREG, NRRIDG, or NEWRAP, MAXFUNC=123 by default.
- When TECHNIQUE=QUANEW or DBLDOG, MAXFUNC=500 by default.
- When TECHNIQUE=CONGRA, MAXFUNC=1,000 by default.
- When TECHNIQUE=LBFGS, MAXFUNC=2,000 by default.
- When TECHNIQUE=NMSIMP, MAXFUNC=3,000 by default.

MAXITER=*n*

specifies the maximum number *n* of iterations in the optimization process. These default values also apply when *n* is specified as a missing value. The default values are as follows, depending on the optimization technique:

- When TECHNIQUE=TRUREG, NRRIDG, or NEWRAP, MAXITER=50 by default.
- When TECHNIQUE=QUANEW or DBLDOG, MAXITER=200 by default.
- When TECHNIQUE=CONGRA, MAXITER=400 by default.
- When TECHNIQUE=LBFGS or NMSIMP, MAXITER=1,000 by default.

MAXTIME=*r*

specifies an upper limit of *r* seconds of CPU time for the optimization process. The default value is the largest floating-point double representation of your computer. The time specified by the MAXTIME= option is checked only once at the end of each iteration. Therefore, the actual run time can be longer than *r*.

MINITER=*n*

specifies the minimum number of iterations. The default value is 0. If you request more iterations than are needed for convergence to a stationary point, the optimization algorithms can behave strangely. For example, the effect of rounding errors can prevent the algorithm from continuing for the required number of iterations.

OPTTOL=*r*

specifies the criterion value by which the current iterate is determined to be an acceptable approximation of a local optimum, where *r* is a positive real number. Specifically, the optimizer determines that the current iterate is a local optimum when the norm of the scaled vector of the optimality conditions is less than *r*. By default, OPTTOL=1E–5.

This option is relevant only when you specify TECHNIQUE=ACTIVESET or IPDIRECT.

TECHNIQUE=*keyword*

TECH=*keyword*

specifies the optimization technique for obtaining restricted maximum likelihood estimation (REML) or maximum likelihood estimation (ML). There is no algorithm for optimizing general nonlinear functions that always finds the global optimum for a general nonlinear optimization problem in a reasonable amount of time. Because no single optimization technique is always superior to others, PROC LMIXED provides a variety of optimization techniques that work well in various circumstances.

For more information about traditional techniques such as Newton-Raphson optimization, see the section "Optimization Options" on page 57 in the chapter "Shared Concepts" of *SAS Visual Statistics: Procedures*.

For more information about modern techniques such as active-set and interior-point direct optimization, see the chapter "The Nonlinear Programming Solver" in *SAS/OR User's Guide: Mathematical Programming*.

You can specify the following *keywords*:

ACTIVESET | AS performs an active-set optimization.

CONGRA performs a conjugate-gradient optimization.

DBLDOG performs a version of double-dogleg optimization.

IPDIRECT | IP performs an interior-point direct optimization.

LBFGS performs a limited-memory BFGS optimization.

NEWRAP performs a Newton-Raphson optimization that combines a line-search algorithm with ridging.

NMSIMP performs a Nelder-Mead simplex optimization.

NONE performs no optimization.

NRRIDG performs a Newton-Raphson optimization with ridging.

QUANEW performs a dual quasi-Newton optimization.

TRUREG performs a trust-region optimization.

By default, TECHNIQUE=NRRIDG when DMMETHOD=DENSE. Otherwise, TECHNIQUE=QUANEW by default.

XCONV=*r*

specifies the relative parameter convergence criterion, which depends on the technique that is specified in the TECHNIQUE= option as follows:

- For all techniques except the Nelder-Mead simplex technique, termination requires a small relative parameter change in subsequent iterations:

$$\frac{\max_j |\psi_j^{(k)} - \psi_j^{(k-1)}|}{\max(|\psi_j^{(k)}|, |\psi_j^{(k-1)}|)} \leq r$$

- For the Nelder-Mead simplex technique, the same formula is used, but $\psi_j^{(k)}$ is defined as the vertex that has the lowest function value and $\psi_j^{(k-1)}$ is defined as the vertex that has the highest function value in the simplex.

The default value is r = 1E–8 for the NMSIMP technique and r = 0 otherwise.

OUTPUT Statement

OUTPUT OUT=*libref.data-table*
< **COPYVARS=***(variables)* >
< *keyword* < *=name* > >. . . < *keyword* < *=name* > > **;**

The OUTPUT statement creates a data table that contains observationwise statistics, which are computed after the model is fitted. The variables in the input data table are *not* included in the output data table in order to avoid data duplication for large data tables; however, variables that you specify in the COPYVARS= option are included.

The output statistics are computed on the basis of the parameter estimates. By default, the predicted values are computed for observations that have missing response values. You can use the OUTPUT statement to obtain the predicted values for new observations.

You must specify the following option:

OUT=*libref.data-table*

names the output data table for PROC LMIXED to use. You must specify this option before any other options. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to where the data table is to be stored, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 864.

data-table specifies the name of the output data table.

You can also specify the following syntax elements:

ALLSTATS
ALLSTAT
adds all available statistics to the output data table.

ALPHA=*number*
specifies the significance level for the construction of confidence intervals in the output data table. The confidence level is $1 - $ *number*, where *number* must be between 0 and 1. By default, *number* is equal to the value of the ALPHA= option in the MODEL statement, or 0.05 if that option is not specified.

COPYVAR=*variable*
COPYVARS=(*variables***)**
copies one or more *variables* from the input data table to the output data table.

keyword **<=***name***>**
specifies the statistics to include in the output data table and optionally names the new variables that contain the statistics. Specify a keyword for each desired statistic (see the following list of keywords), followed optionally by an equal sign and a variable to contain the statistic.

If you specify *keyword=name*, the new variable that contains the requested statistic has the specified *name*. If you omit the optional *=name* after a *keyword*, then a default name is used.

You can specify the following values for *keyword* to request statistics:

Table 17.10 Keywords for Output Statistics

Keyword	Description	Expression
PRED	Linear predictor	$\widehat{\eta} = \mathbf{x}'\widehat{\boldsymbol{\beta}} + \mathbf{z}'\widehat{\boldsymbol{\gamma}}$
PREDPA	Marginal linear predictor	$\widehat{\eta}_m = \mathbf{x}'\widehat{\boldsymbol{\beta}}$
STDERR	Standard error of linear predictor	$\sqrt{\mathrm{Var}[\widehat{\eta} - \mathbf{z}'\boldsymbol{\gamma}]}$
STDERRPA	Standard error of marginal linear predictor	$\sqrt{\mathrm{Var}[\widehat{\eta}_m]}$
RESIDUAL	Residual	$r = y - \widehat{\eta}$
RESIDUALPA	Marginal residual	$r_m = y - \widehat{\eta}_m$
PEARSON	Pearson-type residual	$r/\sqrt{\widehat{\mathrm{Var}}[y\|\boldsymbol{\gamma}]}$
PEARSONPA	Marginal Pearson-type residual	$r_m/\sqrt{\widehat{\mathrm{Var}}[y]}$
STUDENT	Studentized residual	$r/\sqrt{\widehat{\mathrm{Var}}[r]}$
STUDENTPA	Studentized marginal residual	$r_m/\sqrt{\widehat{\mathrm{Var}}[r_m]}$
LCL	Lower prediction limit for linear predictor	
LCLPA	Lower confidence limit for marginal linear predictor	
UCL	Upper prediction limit for linear predictor	
UCLPA	Upper confidence limit for marginal linear predictor	
VARIANCE	Conditional variance of response variable	$\widehat{\mathrm{Var}}[y\|\boldsymbol{\gamma}]$
VARIANCEPA	Marginal variance of response variable	$\widehat{\mathrm{Var}}[y]$

PARMS Statement

PARMS < *(value-list)...* > < / *options* > **;**

The PARMS statement specifies initial values for the covariance parameters, or it requests a grid search over several values of these parameters. You must specify the values in the order in which they appear in the "Covariance Parameter Estimates" table.

When there is a BLUP statement, the PARMS statement is required and the specified values are not the initial values. They are the fixed covariance parameter values that are used in computing the best linear unbiased prediction (BLUP).

The *value-list* specification can take any of several forms:

m	a single value
$m_1, m_2, \ldots, m_n$	several values
m to n	a sequence in which m equals the starting value, n equals the ending value, and the increment equals 1
m to n by i	a sequence in which m equals the starting value, n equals the ending value, and the increment equals i
m_1, m_2 to m_3	mixed values and sequences

Suppose that a model has three covariance parameters that have known values of 2, 1, and 3. You can fix the variance components at these values by using the following statement:

```
parms (2) (1) (3) /hold;
```

The NOPROFILE option in the PROC LMIXED statement suppresses profiling of the residual variance parameter during its calculations, thereby enabling its value to be held at 3 as specified in the PARMS statement.

You can use the PARMS statement to input known parameters.

If you specify more than one set of initial values, the LMIXED procedure performs a grid search of the likelihood surface and uses the best point on the grid for subsequent analysis. Specifying a large number of grid points can result in long computing times.

The results from the PARMS statement are the values of the parameters on the specified grid (denoted by CovP1 through CovP*n*), the residual variance (possibly estimated) for models that have a residual variance parameter, and various functions of the likelihood.

If you specifies multiple PARMS statements, the LMIXED procedure uses the first one and ignores the rest.

You can specify the following *options* in the PARMS statement after a slash (/):

HOLD< =*order-list* **>**

EQCONS< =*order-list* **>**

HOLD< =ALL >

EQCONS< =ALL >

holds the values of the covariance parameters whose order is specified in the *order-list* to the initial values that are specified in the *value-list* in the RANDOM statement.

For example, the following statement constrains the first and third covariance parameters to equal 5 and 2, respectively:

```
parms (5) (3) (2) (3) / hold=1,3;
```

If you specify ALL or you do not specify *order-list*, then all covariance parameters are held to their initial values. When not all parameters are held to their initial values, this is referred to as partial holding.

Specifying the HOLD= option implies the specification of NOPROFILE option in the PROC LMIXED statement.

The partial holding cannot retain the values exactly for the TYPE=UN covariance structure. This is because the underlining structure is TYPE=CHOL in the optimization for covariance parameter estimation.

LOWERB=*value-list*

specifies the lower boundary constraints on the covariance parameters, where *value-list* is a list of numbers or missing values (.) separated by commas. You must list the numbers in the order that the LMIXED procedure uses for the covariance parameters, and each number corresponds to a lower boundary constraint. A missing value instructs the LMIXED procedure to use its default constraint. If you do not specify numbers for all the covariance parameters, the LMIXED procedure assumes that the remaining ones are missing.

This option is useful when you want to constrain the $\mathbf{G}$ matrix to be positive definite in order to avoid the more computationally intensive algorithms that would be required when $\mathbf{G}$ becomes singular. The corresponding statements for a random coefficients model are as follows:

```
proc lmixed;
   class person;
   model y = time;
   random int time / type=fa0(2) sub=person;
   parms / lowerb=1e-4,.,1e-4;
run;
```

The TYPE=FA0(2) structure specifies a Cholesky root parameterization for the 2×2 unstructured blocks in $\mathbf{G}$. This parameterization ensures that the $\mathbf{G}$ matrix is nonnegative definite, and the PARMS statement then ensures that it is positive definite by constraining the two diagonal terms to be greater than or equal to 1E–4.

NOITER

requests that no optimization iterations be performed and that the LMIXED procedure use the best value from the grid search to perform inferences.

PARMSDATA=*libref.data-table*

PDATA=*libref.data-table*

reads in covariance parameter values from a data table. *libref.data-table* is a two-level name, where *libref* refers to the library, and *data-table* specifies the name of the input data table. For more information about this two-level name, see the DATA= option and the section "Using CAS Sessions and CAS Engine Librefs" on page 864.

The data table should contain either the numerical variables Estimate and RowId or the numerical variables Covp1–Covp*q*, where *q* denotes the number of covariance parameters.

If the data table contains the numerical variables Covp1–Covp*q* and it contains multiple sets of values for covariance parameters, then the LMIXED procedure evaluates the initial objective function for each set of values and commences the optimization step by using the set that has the lowest functional value as the starting values. For example, the following statements request that the objective function be evaluated for three sets of initial values:

```
data mylib.data_covp;
   input covp1-covp4;
   datalines;
180 200 170 1000
170 190 160  900
160 180 150  800
;
proc lmixed;
   class A B C rep;
   model yield = A;
   random rep B C;
   parms / pdata=mylib.data_covp;
run;
```

Another way of specifying the initial values for the covariance parameters is to use the numerical variables Estimate and RowId in the data table. The values in RowId variable indicate the order of the covariance parameters in the "Covariance Parameter Estimates" table.

```
data mylib.covparm;
   input rowid estimate;
   datalines;
1  180
2  200
3  170
4  1000
;
```

A BY variable is not supported in the PARMSDATA= data table. The data table is processed in its entirety for every BY group and a message is written to the log. The same set of starting values is used by all BY groups.

```
data mylib.data_covp;
   input covp1-covp4;
   datalines;
180 200 170 1000
;
proc lmixed;
   class A B C rep;
   model yield = A;
   random rep B C;
   parms / pdata=mylib.data_covp;
```

```
   by year;
run;
```

UPPERB=*value-list*

specifies upper boundary constraints on the covariance parameters, where the *value-list* specification is a list of numbers or missing values (.) separated by commas. You must list the numbers in the order that the LMIXED procedure uses for the covariance parameters, and each number corresponds to the upper boundary constraint. A missing value instructs the LMIXED procedure to use its default constraint. If you do not specify numbers for all of the covariance parameters, the LMIXED procedure assumes that the remaining ones are missing.

RANDOM Statement

RANDOM *random-effects* < / *options* > **;**

The RANDOM statement defines the random effects that constitute the $\boldsymbol{\gamma}$ vector in the mixed model. You can use this statement to specify traditional variance component models and to specify random coefficients. The random effects can be classification or continuous, and multiple RANDOM statements are allowed.

Using notation from the section "Linear Mixed Models Theory" on page 935, the purpose of the RANDOM statement is to define the $\mathbf{Z}$ matrix of the mixed model, the random effects in the $\boldsymbol{\gamma}$ vector, and the structure of $\mathbf{G}$. The $\mathbf{Z}$ matrix is constructed exactly as the $\mathbf{X}$ matrix for the fixed effects is constructed, and the $\mathbf{G}$ matrix is constructed to correspond with the effects that constitute $\mathbf{Z}$. The structure of $\mathbf{G}$ is defined by using the TYPE= option.

You can specify INTERCEPT (or INT) as a random effect to indicate the intercept. The LMIXED procedure does not include the intercept in the RANDOM statement by default as it does in the MODEL statement.

Table 17.11 summarizes the important options in the RANDOM statement. All options are subsequently discussed in alphabetical order.

Table 17.11 Summary of Important RANDOM Statement Options

Option	Description
Construction of Covariance Structure	
GROUP=	Defines an effect that specifies the covariance structure of groups
SUBJECT=	Identifies the subjects in the model
TYPE=	Specifies the covariance structure
Statistical Output	
ALPHA=α	Determines the confidence level $(1-\alpha)$
CL	Requests confidence limits for predictors of random effects
SOLUTION	Displays solutions $\widehat{\boldsymbol{\gamma}}$ of the random effects

You can specify the following *options* in the RANDOM statement after a slash (/):

ALPHA=*α*

sets the confidence level to be $1-\alpha$ for each confidence interval of the random-effects estimates, where α must be a number between 0 and 1. By default, ALPHA=0.05.

CL

constructs *t*-type confidence limits for each of the random-effect estimates. The confidence level is 0.95 by default; this can be changed with the ALPHA= option.

GROUP=*effect*

defines an effect that specifies heterogeneity in the covariance structure of $\mathbf{G}$. All observations that have the same level of the group effect have the same covariance parameters. Each new level of the group effect produces a new set of covariance parameters that has the same structure as the original group. You should exercise caution in defining the group effect, because strange covariance patterns can result from its misuse. Also, the group effect can greatly increase the number of estimated covariance parameters, which can impede the optimization process.

The group *effect* must consist of CLASS variables only.

SOLUTION

S

produces the solution for the random-effects parameters. Using notation from the section "Linear Mixed Models Theory" on page 935, these estimates are the empirical best linear unbiased predictors (EBLUPs), $\widehat{\boldsymbol{\gamma}} = \widehat{\mathbf{G}}\mathbf{Z}'\widehat{\mathbf{V}}^{-1}(\mathbf{y} - \mathbf{X}\widehat{\boldsymbol{\beta}})$. They can be useful for comparing the random effects from different experimental units and can also be treated as residuals in performing diagnostics for your mixed model.

The numbers displayed in the SE Pred column of the "Solution for Random Effects" table are not the standard errors of the $\widehat{\boldsymbol{\gamma}}$ that are displayed in the Estimate column; rather, they are the standard errors of predictions $\widehat{\boldsymbol{\gamma}}_i - \boldsymbol{\gamma}_i$, where $\widehat{\boldsymbol{\gamma}}_i$ is the *i*th EBLUP and $\boldsymbol{\gamma}_i$ is the *i*th random-effect parameter.

SUBJECT=*effect*

SUB=*effect*

identifies the subjects in your mixed model. Complete independence is assumed across subjects; thus, the SUBJECT= option produces a block-diagonal structure in $\mathbf{G}$ that has identical blocks. In fact, specifying an *effect* is equivalent to nesting all other effects in the RANDOM statement within the *effect*.

TYPE=*covariance-structure*

specifies the covariance structure of $\mathbf{G}$. Valid values for *covariance-structure* and their descriptions are listed in Table 17.12. Although a variety of structures are available, most applications call for either TYPE=VC or TYPE=UN.

By default, TYPE=VC (a structure that contains variance components), which models a different variance component for each random effect. The TYPE=UN (unstructured) option is useful for correlated random coefficient models. For example, the following statement specifies a random intercept-slope model that has different variances for the intercept and slope and a covariance between them:

```
random intercept age / type=un subject=student;
```

You can also use TYPE=FA0(2) to request a **G** estimate that is constrained to be nonnegative definite.

If you are constructing your own columns of **Z** by using continuous variables, you can use the TYPE=TOEP(1) structure to group them together to have a common variance component. If you want to have different covariance structures in different parts of **G**, you must use multiple RANDOM statements with different TYPE= options.

In Table 17.12, the Parameters column represents the number of covariance parameters in the structure, t is the overall dimension of the covariance matrix, and $1(A)$ equals 1 when A is true and 0 otherwise. For example, $1(i = j)$ equals 1 when $i = j$ and 0 otherwise, and $1(|i - j| < q)$ equals 1 when $|i - j| < q$ and 0 otherwise. For the TYPE=TOEPH structures, $\rho_0 = 1$; for the TYPE=UNR structures, $\rho_{ii} = 1$ for all i.

Table 17.12 Covariance Structures

Structure	Description	Parameters	(i, j) element
ANTE(1)	Antedependence	$2t - 1$	$\sigma_i \sigma_j \prod_{k=i}^{j-1} \rho_k$
AR(1)	Autoregressive(1)	2	$\sigma^2 \rho^{\lvert i-j \rvert}$
ARH(1)	Heterogeneous AR(1)	$t + 1$	$\sigma_i \sigma_j \rho^{\lvert i-j \rvert}$
ARMA(1,1)	Autoregressive moving average(1,1)	3	$\sigma^2[\gamma \rho^{\lvert i-j \rvert - 1} 1(i \neq j) + 1(i = j)]$
CHOL	Cholesky root	$t(t + 1)/2$	l_{ij}
CS	Compound symmetry	2	$\sigma_1 + \sigma^2 1(i = j)$
CSH	Heterogeneous compound symmetry	$t + 1$	$\sigma_i \sigma_j [\rho 1(i \neq j) + 1(i = j)]$
FA(q)	Factor analytic (FA)	$\frac{q}{2}(2t - q + 1) + t$	$\Sigma_{k=1}^{\min(i,j,q)} \lambda_{ik} \lambda_{jk} + \sigma_i^2 1(i = j)$
FA0(q)	No diagonal FA	$\frac{q}{2}(2t - q + 1)$	$\Sigma_{k=1}^{\min(i,j,q)} \lambda_{ik} \lambda_{jk}$
FA1(q)	Equal diagonal FA	$\frac{q}{2}(2t - q + 1) + 1$	$\Sigma_{k=1}^{\min(i,j,q)} \lambda_{ik} \lambda_{jk} + \sigma^2 1(i = j)$
HF	Huynh-Feldt	$t + 1$	$(\sigma_i^2 + \sigma_j^2)/2 + \lambda 1(i \neq j)$
SIMPLE	An alias for VC	q	$\sigma_k^2 1(i = j)$ for the kth effect
TOEP	Toeplitz	t	$\sigma_{\lvert i-j \rvert + 1}$
TOEP(q)	Banded Toeplitz (TOEP)	q	$\sigma_{\lvert i-j \rvert + 1} 1(\lvert i - j \rvert < q)$
TOEPH	Heterogeneous TOEP	$2t - 1$	$\sigma_i \sigma_j \rho_{\lvert i-j \rvert}$
TOEPH(q)	Banded heterogeneous TOEP	$t + q - 1$	$\sigma_i \sigma_j \rho_{\lvert i-j \rvert} 1(\lvert i - j \rvert < q)$
UC	Uniform correlation (UC)	2	$\sigma^2[\rho 1(i \neq j) + 1(i = j)]$
UCH	Heterogeneous UC	$t + 1$	$\sigma_i \sigma_j [\rho 1(i \neq j) + 1(i = j)]$
UN	Unstructured	$t(t + 1)/2$	σ_{ij}
UN(q)	Banded	$\frac{q}{2}(2t - q + 1)$	$\sigma_{ij} 1(\lvert i - j \rvert < q)$
UNR	Unstructured correlation	$t(t + 1)/2$	$\sigma_i \sigma_j \rho_{\max(i,j) \min(i,j)}$
VC	Variance components	q	$\sigma_k^2 1(i = j)$ for the kth effect

Table 17.13 lists the covariance types and applicable computing methods (sparse or dense) for each covariance type.

Table 17.13 Covariance Structures

Structure	Description	DMMETHOD=
ANTE(1)	Antedependence	DENSE
AR(1)	Autoregressive(1)	DENSE or SPARSE
ARH(1)	Heterogeneous AR(1)	DENSE
ARMA(1,1)	Autoregressive moving average(1,1)	DENSE
CHOL	Cholesky root	SPARSE
CS	Compound symmetry	DENSE or SPARSE
CSH	Heterogeneous compound symmetry	DENSE or SPARSE
FA(q)	Factor analytic (FA)	DENSE
FA0(q)	No diagonal FA	DENSE
FA1(q)	Equal diagonal FA	DENSE
HF	Huynh-Feldt	DENSE
SIMPLE	An alias for VC	DENSE or SPARSE
TOEP(1)	Toeplitz	DENSE or SPARSE
TOEP(q)	Banded Toeplitz (TOEP)	DENSE
TOEPH	Heterogeneous TOEP	DENSE
TOEPH(q)	Banded heterogeneous TOEP	DENSE
UC	Uniform correlation (UC)	SPARSE
UCH	Heterogeneous UC	SPARSE
UN	Unstructured	DENSE or SPARSE
UN(q)	Banded	DENSE
UNR	Unstructured correlation	DENSE
VC	Variance components	DENSE or SPARSE

Table 17.14 lists some examples of the structures in Table 17.12.

Table 17.14 Covariance Structure Examples

Description	Structure	Example
First-order antedependence	ANTE(1)	$\begin{bmatrix} \sigma_1^2 & \sigma_1\sigma_2\rho_1 & \sigma_1\sigma_3\rho_1\rho_2 \\ \sigma_2\sigma_1\rho_1 & \sigma_2^2 & \sigma_2\sigma_3\rho_2 \\ \sigma_3\sigma_1\rho_2\rho_1 & \sigma_3\sigma_2\rho_2 & \sigma_3^2 \end{bmatrix}$

Table 17.14 *continued*

Description	Structure	Example
First-order autoregressive	AR(1)	$\sigma^2 \begin{bmatrix} 1 & \rho & \rho^2 & \rho^3 \\ \rho & 1 & \rho & \rho^2 \\ \rho^2 & \rho & 1 & \rho \\ \rho^3 & \rho^2 & \rho & 1 \end{bmatrix}$
Heterogeneous autoregressive(1)	ARH(1)	$\begin{bmatrix} \sigma_1^2 & \sigma_1\sigma_2\rho & \sigma_1\sigma_3\rho^2 & \sigma_1\sigma_4\rho^3 \\ \sigma_2\sigma_1\rho & \sigma_2^2 & \sigma_2\sigma_3\rho & \sigma_2\sigma_4\rho^2 \\ \sigma_3\sigma_1\rho^2 & \sigma_3\sigma_2\rho & \sigma_3^2 & \sigma_3\sigma_4\rho \\ \sigma_4\sigma_1\rho^3 & \sigma_4\sigma_2\rho^2 & \sigma_4\sigma_3\rho & \sigma_4^2 \end{bmatrix}$
First-order autoregressive moving average	ARMA(1,1)	$\sigma^2 \begin{bmatrix} 1 & \gamma & \gamma\rho & \gamma\rho^2 \\ \gamma & 1 & \gamma & \gamma\rho \\ \gamma\rho & \gamma & 1 & \gamma \\ \gamma\rho^2 & \gamma\rho & \gamma & 1 \end{bmatrix}$
Cholesky root	CHOL	$\begin{bmatrix} l_{11} & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 \\ l_{41} & l_{42} & l_{43} & l_{44} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} & l_{31} & l_{41} \\ 0 & l_{22} & l_{32} & l_{42} \\ 0 & 0 & l_{33} & l_{43} \\ 0 & 0 & 0 & l_{44} \end{bmatrix}$
Compound symmetry	CS	$\begin{bmatrix} \sigma^2+\sigma_1 & \sigma_1 & \sigma_1 & \sigma_1 \\ \sigma_1 & \sigma^2+\sigma_1 & \sigma_1 & \sigma_1 \\ \sigma_1 & \sigma_1 & \sigma^2+\sigma_1 & \sigma_1 \\ \sigma_1 & \sigma_1 & \sigma_1 & \sigma^2+\sigma_1 \end{bmatrix}$
Heterogeneous compound symmetry	CSH	$\begin{bmatrix} \sigma_1^2 & \sigma_1\sigma_2\rho & \sigma_1\sigma_3\rho & \sigma_1\sigma_4\rho \\ \sigma_2\sigma_1\rho & \sigma_2^2 & \sigma_2\sigma_3\rho & \sigma_2\sigma_4\rho \\ \sigma_3\sigma_1\rho & \sigma_3\sigma_2\rho & \sigma_3^2 & \sigma_3\sigma_4\rho \\ \sigma_4\sigma_1\rho & \sigma_4\sigma_2\rho & \sigma_4\sigma_3\rho & \sigma_4^2 \end{bmatrix}$
First-order factor analytic	FA(1)	$\begin{bmatrix} \lambda_1^2+d_1 & \lambda_1\lambda_2 & \lambda_1\lambda_3 & \lambda_1\lambda_4 \\ \lambda_2\lambda_1 & \lambda_2^2+d_2 & \lambda_2\lambda_3 & \lambda_2\lambda_4 \\ \lambda_3\lambda_1 & \lambda_3\lambda_2 & \lambda_3^2+d_3 & \lambda_3\lambda_4 \\ \lambda_4\lambda_1 & \lambda_4\lambda_2 & \lambda_4\lambda_3 & \lambda_4^2+d_4 \end{bmatrix}$
Huynh-Feldt	HF	$\begin{bmatrix} \sigma_1^2 & \frac{\sigma_1^2+\sigma_2^2}{2}-\lambda & \frac{\sigma_1^2+\sigma_3^2}{2}-\lambda \\ \frac{\sigma_2^2+\sigma_1^2}{2}-\lambda & \sigma_2^2 & \frac{\sigma_2^2+\sigma_3^2}{2}-\lambda \\ \frac{\sigma_3^2+\sigma_1^2}{2}-\lambda & \frac{\sigma_3^2+\sigma_2^2}{2}-\lambda & \sigma_3^2 \end{bmatrix}$
Toeplitz	TOEP	$\begin{bmatrix} \sigma^2 & \sigma_1 & \sigma_2 & \sigma_3 \\ \sigma_1 & \sigma^2 & \sigma_1 & \sigma_2 \\ \sigma_2 & \sigma_1 & \sigma^2 & \sigma_1 \\ \sigma_3 & \sigma_2 & \sigma_1 & \sigma^2 \end{bmatrix}$
Toeplitz with two bands	TOEP(2)	$\begin{bmatrix} \sigma^2 & \sigma_1 & 0 & 0 \\ \sigma_1 & \sigma^2 & \sigma_1 & 0 \\ 0 & \sigma_1 & \sigma^2 & \sigma_1 \\ 0 & 0 & \sigma_1 & \sigma^2 \end{bmatrix}$

Table 17.14 *continued*

Description	Structure	Example
Heterogeneous Toeplitz	TOEPH	$\begin{bmatrix} \sigma_1^2 & \sigma_1\sigma_2\rho_1 & \sigma_1\sigma_3\rho_2 & \sigma_1\sigma_4\rho_3 \\ \sigma_2\sigma_1\rho_1 & \sigma_2^2 & \sigma_2\sigma_3\rho_1 & \sigma_2\sigma_4\rho_2 \\ \sigma_3\sigma_1\rho_2 & \sigma_3\sigma_2\rho_1 & \sigma_3^2 & \sigma_3\sigma_4\rho_1 \\ \sigma_4\sigma_1\rho_3 & \sigma_4\sigma_2\rho_2 & \sigma_4\sigma_3\rho_1 & \sigma_4^2 \end{bmatrix}$
Uniform correlation	UC	$\sigma^2 \begin{bmatrix} 1 & \rho & \rho & \rho \\ \rho & 1 & \rho & \rho \\ \rho & \rho & 1 & \rho \\ \rho & \rho & \rho & 1 \end{bmatrix}$
Heterogeneous UC	UCH	$\begin{bmatrix} \sigma_1^2 & \sigma_1\sigma_2\rho & \sigma_1\sigma_3\rho & \sigma_1\sigma_4\rho \\ \sigma_2\sigma_1\rho & \sigma_2^2 & \sigma_2\sigma_3\rho & \sigma_2\sigma_4\rho \\ \sigma_3\sigma_1\rho & \sigma_3\sigma_2\rho & \sigma_3^2 & \sigma_3\sigma_4\rho \\ \sigma_4\sigma_1\rho & \sigma_4\sigma_2\rho^2 & \sigma_4\sigma_3\rho & \sigma_4^2 \end{bmatrix}$
Unstructured	UN	$\begin{bmatrix} \sigma_1^2 & \sigma_{21} & \sigma_{31} & \sigma_{41} \\ \sigma_{21} & \sigma_2^2 & \sigma_{32} & \sigma_{42} \\ \sigma_{31} & \sigma_{32} & \sigma_3^2 & \sigma_{43} \\ \sigma_{41} & \sigma_{42} & \sigma_{43} & \sigma_4^2 \end{bmatrix}$
Banded main diagonal	UN(1)	$\begin{bmatrix} \sigma_1^2 & 0 & 0 & 0 \\ 0 & \sigma_2^2 & 0 & 0 \\ 0 & 0 & \sigma_3^2 & 0 \\ 0 & 0 & 0 & \sigma_4^2 \end{bmatrix}$
Unstructured correlations	UNR	$\begin{bmatrix} \sigma_1^2 & \sigma_1\sigma_2\rho_{21} & \sigma_1\sigma_3\rho_{31} & \sigma_1\sigma_4\rho_{41} \\ \sigma_2\sigma_1\rho_{21} & \sigma_2^2 & \sigma_2\sigma_3\rho_{32} & \sigma_2\sigma_4\rho_{42} \\ \sigma_3\sigma_1\rho_{31} & \sigma_3\sigma_2\rho_{32} & \sigma_3^2 & \sigma_3\sigma_4\rho_{43} \\ \sigma_4\sigma_1\rho_{41} & \sigma_4\sigma_2\rho_{42} & \sigma_4\sigma_3\rho_{43} & \sigma_4^2 \end{bmatrix}$
Variance components	VC (default)	$\begin{bmatrix} \sigma_B^2 & 0 & 0 & 0 \\ 0 & \sigma_B^2 & 0 & 0 \\ 0 & 0 & \sigma_{AB}^2 & 0 \\ 0 & 0 & 0 & \sigma_{AB}^2 \end{bmatrix}$

The following list provides some further information about these covariance structures:

TYPE=ANTE(1) specifies the first-order antedependence structure (Kenward 1987; Patel 1991; Macchiavelli and Arnold 1994). In Table 17.12, σ_i^2 is the i variance parameter, and ρ_k is the kth autocorrelation parameter that satisfies $|\rho_k| < 1$.

This covariance type is available only when DMMETHOD=DENSE.

TYPE=AR(1) specifies a first-order autoregressive structure. The LMIXED procedure imposes the constraint $|\rho| < 1$ for stationarity.

TYPE=ARH(1) specifies a heterogeneous first-order autoregressive structure. As with TYPE=AR(1), the LMIXED procedure imposes the constraint $|\rho| < 1$ for stationarity.

This covariance type is available only when DMMETHOD=DENSE.

TYPE=ARMA(1,1) specifies the first-order autoregressive moving average structure. In Table 17.12, ρ is the autoregressive parameter, γ models a moving average component, and σ^2 is the residual variance. In the notation of Fuller (1976, p. 68), $\rho = \theta_1$ and

$$\gamma = \frac{(1 + b_1\theta_1)(\theta_1 + b_1)}{1 + b_1^2 + 2b_1\theta_1}$$

The example in Table 17.14 and $|b_1| < 1$ imply that

$$b_1 = \frac{\beta - \sqrt{\beta^2 - 4\alpha^2}}{2\alpha}$$

where $\alpha = \gamma - \rho$ and $\beta = 1 + \rho^2 - 2\gamma\rho$. The LMIXED procedure imposes the constraints $|\rho| < 1$ and $|\gamma| < 1$ for stationarity, although the resulting covariance matrix is not positive definite for some values of ρ and γ in this region. When the estimated value of ρ becomes negative, the computed covariance is multiplied by $\cos(\pi d_{ij})$ to account for the negativity.

This covariance type is available only when DMMETHOD=DENSE.

TYPE=CHOL specifies an unstructured variance-covariance matrix that is parameterized through its Cholesky root. All diagonal values are constrained to be positive. This parameterization guarantees a positive definite covariance matrix. For example, a 2×2 unstructured covariance matrix can be written as

$$\mathrm{Var}[\boldsymbol{\xi}] = \begin{bmatrix} \sigma_1^2 & \sigma_{21} \\ \sigma_{21} & \sigma_2^2 \end{bmatrix}$$

Without imposing constraints on the three parameters, there is no guarantee that the estimated variance matrix is positive definite. Even if σ_1^2 and σ_2^2 are nonzero, a large value for σ_{21} can lead to a negative eigenvalue of $\mathrm{Var}[\boldsymbol{\xi}]$. The Cholesky root of a positive definite matrix $\mathbf{A}$ is a lower triangular matrix $\mathbf{L}$ such that $\mathbf{L}\mathbf{L}' = \mathbf{A}$. The Cholesky root of the preceding 2×2 matrix can be written as

$$\mathbf{L} = \begin{bmatrix} l_{11} & 0 \\ l_{21} & l_{22} \end{bmatrix}$$

The elements of the unstructured variance matrix are then simply $\sigma_1^2 = l_{11}^2$, $\sigma_{21} = l_{21}l_{11}$, and $\sigma_2^2 = l_{21}^2 + l_{22}^2$. Similar operations yield the generalization to covariance matrices of higher orders.

For example, the following statements model the covariance matrix of each subject as an unstructured matrix:

```
proc lmixed;
   class sub;
   model y = x;
   random  time / sub=patient type=chol;
run;
```

The LMIXED procedure constrains the diagonal elements of the Cholesky root to be positive. This guarantees that the structure is positive definite.

This covariance type is available only when DMMETHOD=SPARSE.

TYPE=CS specifies the compound-symmetry structure, which has constant variance and constant covariance.

TYPE=CSH specifies the heterogeneous compound-symmetry structure. This structure has a different variance parameter for each diagonal element, and it uses the square roots of these parameters in the off-diagonal entries. In Table 17.12, σ_i^2 is the i variance parameter, and ρ is the correlation parameter that satisfies $|\rho| < 1$.

TYPE=FA(q) specifies the factor-analytic structure with q factors (Jennrich and Schluchter 1986). This structure is of the form $\mathbf{\Lambda}\mathbf{\Lambda}' + \mathbf{D}$, where $\mathbf{\Lambda}$ is a $t \times q$ rectangular matrix and $\mathbf{D}$ is a $t \times t$ diagonal matrix with t different parameters. When $q > 1$, the elements of $\mathbf{\Lambda}$ in its upper right corner (that is, the elements in the i row and j column for $j > i$) are set to 0 to fix the rotation of the structure.

This covariance type is available only when DMMETHOD=DENSE.

TYPE=FA0(q) is similar to the FA(q) structure except that no diagonal matrix $\mathbf{D}$ is included. When $q < t$ (that is, when the number of factors is less than the dimension of the matrix), this structure is nonnegative definite but not of full rank. In this situation, you can use this structure for approximating an unstructured $\mathbf{G}$ matrix in the RANDOM statement. When $q = t$, you can use this structure to constrain $\mathbf{G}$ to be nonnegative definite in the RANDOM statement.

This covariance type is available only when DMMETHOD=DENSE.

TYPE=FA1(q) is similar to the TYPE=FA(q) structure except that all the elements in $\mathbf{D}$ are constrained to be equal. This offers a useful and more parsimonious alternative to the full factor-analytic structure.

This covariance type is available only when DMMETHOD=DENSE.

TYPE=HF specifies the Huynh-Feldt covariance structure (Huynh and Feldt 1970). This structure is similar to the TYPE=CSH structure in that it has the same number of parameters and heterogeneity along the main diagonal. However, it constructs the off-diagonal elements by taking arithmetic means rather than geometric means.

You can perform a likelihood ratio test of the Huynh-Feldt conditions by running the LMIXED procedure twice, once with TYPE=HF and once with TYPE=UN, and then subtracting their respective values of -2 times the maximized likelihood.

If the LMIXED procedure does not converge under your Huynh-Feldt model, you can specify your own starting values in the PARMS statement. The default MIVQUE0 starting values can sometimes be poor for this structure. A good choice for starting values is often the parameter estimates that correspond to an initial fit that uses TYPE=CS.

This covariance type is available only when DMMETHOD=DENSE.

TYPE=SIMPLE is an alias for TYPE=VC.

TYPE=TOEP<(q)> specifies a banded Toeplitz structure. This structure can be viewed as a moving average structure with order equal to $q - 1$. The TYPE=TOEP option is a full Toeplitz matrix, which can be viewed as an autoregressive structure with order equal to the dimension of the matrix. The specification TYPE=TOEP(1) is the same as $\sigma^2 I$, where I is an identity matrix, and it can be useful for specifying the same variance component for several effects.

TYPE=TOEPH<(*q*)> specifies a heterogeneous banded Toeplitz structure. In Table 17.12, σ_i^2 is the i variance parameter and ρ_j is the j correlation parameter that satisfies $|\rho_j| < 1$. If you specify the order parameter q, then the LMIXED procedure estimates only the first q bands of the matrix, setting all higher bands equal to 0. TYPE=TOEPH(1) is equivalent to both TYPE=UN(1).

This covariance type is available only when DMMETHOD=DENSE.

TYPE=UC specifies the uniform correlation structure, which has constant variance and constant correlation,

$$\mathrm{Cov}\left[\xi_i, \xi_j\right] = \begin{cases} \sigma^2 & i = j \\ \sigma^2\rho & i \neq j \end{cases}$$

Under uniform correlation, the $\mathbf{G}$ matrix has the form $\sigma^2[(1-\rho)\mathbf{I} + \rho\mathbf{J}]$. The variance σ^2 is constrained to be positive, and the correlation ρ is constrained to be greater than $-1/(t-1)$, where t is the dimension of the structure. These constraints guarantee that the structure is positive definite. This structure is equivalent to the compound-symmetry structure with a better numerical property in terms of optimization.

The uniform correlation structure arises frequently in agriculture and animal sciences.

This covariance type is available only when DMMETHOD=SPARSE.

TYPE=UCH specifies the heterogeneous uniform correlation structure. This structure has a different variance parameter for each diagonal element, and it uses the square roots of these parameters in the off-diagonal entries. In Table 17.12, σ_i^2 is the ith variance parameter that satisfies $\sigma_i^2 > 0$, and ρ is the correlation parameter that satisfies $\rho > -1/(t-1)$, where t is the dimension of the structure. This guarantees that the structure is positive definite.

This covariance type is available only when DMMETHOD=SPARSE.

TYPE=UN specifies a completely general (unstructured) covariance matrix that is parameterized directly in terms of variances and covariances. The variances are constrained to be nonnegative, and the covariances are unconstrained. When DMMETHOD=DENSE, this structure is not constrained to be nonnegative definite in order to avoid nonlinear constraints; however, you can use the TYPE=FA0 structure if you want this constraint to be imposed by a Cholesky factorization. When DMMETHOD=SPARSE, this structure is constrained to be nonnegative definite by a Cholesky factorization.

TYPE=UN(*q*) specifies a banded unstructured covariance matrix that is parameterized directly in terms of variances and covariances. The LMIXED procedure estimates only the first q bands of the matrix, setting all higher bands equal to 0.

This covariance type is available only when DMMETHOD=DENSE.

TYPE=UNR specifies a completely general (unstructured) covariance matrix that is parameterized in terms of variances and correlations. This structure fits the same model as the TYPE=UN option but with a different parameterization. The i variance parameter is σ_i^2. The parameter ρ_{jk} is the correlation between the j and k measurements; it satisfies $|\rho_{jk}| < 1$.

This covariance type is available only when DMMETHOD=DENSE.

TYPE=VC specifies standard variance components. This is the default structure for the RANDOM statements. In the RANDOM statement, a distinct variance component is assigned to each effect.

Jennrich and Schluchter (1986) provide general information about the use of covariance structures, and Wolfinger (1996) presents details about many of the heterogeneous structures.

REPEATED Statement

REPEATED *repeated-effect* < / *options* > **;**

The REPEATED statement defines the repeated effect and the residual covariance structure in the mixed model. The residual variance-covariance matrix is denoted as **R**. The *repeated-effect* is required and consists entirely of classification variables. The levels of the *repeated-effect* must be different for each observation within a subject in order to avoid the singular **R** matrix.

Table 17.15 summarizes the *options* available in the REPEATED statement.

Table 17.15 Summary of REPEATED Statement Options

Option	Description
Construction of Covariance Structure	
GROUP=	Defines an effect that specifies heterogeneity in the residual covariance structure
NOCONVERT	Prevents the covariance structure from being internally converted
SUBJECT=	Identifies the subjects in the residual covariance structure
TYPE=	Specifies the residual covariance structure (the default is VC)
Statistical Output	
R=	Displays blocks of the estimated **R** matrix
RC=	Display the Cholesky root (lower) of blocks of the estimated **R** matrix
RCI=	Displays the inverse Cholesky root (lower) of blocks of the estimated **R** matrix
RCORR=	Displays the correlation matrix that corresponds to blocks of the estimated **R** matrix
RI=	Displays the inverse of blocks of the estimated **R** matrix

In the current release, the REPEATED statement is supported only when DMMETHOD=SPARSE.

You must specify one or both of the following *options* after a slash (/):

GROUP=*effect*

GRP=*effect*

defines an effect that specifies heterogeneity in the residual covariance structure. All observations that have the same level of the *effect* have the same covariance parameters. Each new level of the *effect* produces a new set of covariance parameters that have the same structure as the original group. You

should exercise caution in defining the *effect*, because strange covariance patterns can result when it is misused. Also, the *effect* can greatly increase the number of estimated covariance parameters, which can adversely affect the optimization process.

NOCONVERT
prevents the covariance structure from being internally converted to an equivalent one that leads to efficient computation.

SUBJECT=*effect*
SUB=*effect*
identifies the subjects in your mixed model. Complete independence is assumed across subjects; therefore, this option produces a block-diagonal structure in **R** that has identical blocks. The data set is not required to be grouped by the *effect*. When the *effect* consists entirely of classification variables, the blocks of **R** correspond to observations that share the same level of that *effect*. These blocks are sorted according to this *effect* as well.

You can also specify the following *options* after a slash (/):

R<=*value-list***>**
displays blocks of the estimated **R** matrix. The first block that is determined by the SUBJECT= *effect* is the default displayed block.

The *value-list* indicates the subjects for which blocks of **R** are displayed. For example, the following statement displays block matrices for the first, third, and fifth persons:

```
repeated time / type=un subject=person r=1,3,5;
```

See the PARMS statement for the possible forms of *value-list*.

RC<=*value-list***>**
displays the Cholesky root of blocks of the estimated **R** matrix. The *value-list* specification is the same as for the R= option.

RCI<=*value-list***>**
displays the inverse Cholesky root of blocks of the estimated **R** matrix. The *value-list* specification is the same as for the R= option.

RCORR<=*value-list***>**
displays the correlation matrix that corresponds to blocks of the estimated **R** matrix. The *value-list* specification is the same as for the R= option.

RI<=*value-list***>**
produces the inverse of blocks of the estimated **R** matrix. The *value-list* specification is the same as for the R= option.

TYPE=*covariance-structure*
specifies the structure of the residual variance-covariance matrix **R**. The SUBJECT= option defines the blocks of **R**, and the TYPE= option specifies the structure of these blocks. In the current release, the valid values of *covariance-structure* are AR(1), CHOL, CS, CSH, UN, and VC.

For more information about these covariance structure types, see the description in the section "RANDOM Statement" on page 916.

SLICE Statement

SLICE *model-effect* < / *options* > **;**

The SLICE statement is similar to the LSMEANS statement. It provides a general mechanism for performing a partitioned analysis of the LS-means for an interaction. This analysis is also known as an analysis of simple effects (Winer 1971). For the SLICE statement, these effects must be higher-order effects of at least two classification variables. Each effect is then sliced, or partitioned, into subsets that correspond to variables used in forming the effect.

By default, the interaction effect is partitioned by all main effects. For example, the following statements produce simple-effect differences among the A levels for each level of B and simple-effect differences among the B levels for each level of A:

```
class a b;
model y = a b a*b;
slice a*b / diff;
```

For example, if the *model-effect* is a three-way interaction effect, the default output includes comparisons of the two-way interaction means.

Suppose, for example, that the interaction effect A*B is significant in your analysis and that you want to test the effect of A for each level of B. The appropriate statement is

```
slice A*B / sliceBy = B;
```

This statement produces an F test for each level of B that compares the equality of the levels of A.

For example, assume that in a balanced design, factors A and B have $a = 4$ and $b = 3$ levels, respectively. Consider the following statements:

```
class a b;
model y = a b a*b;
slice a*b / sliceby=a diff;
```

The SLICE statement produces four F tests, one for each level of A. The first test is constructed by extracting the three rows that correspond to the first level of A from the coefficient matrix for the A*B interaction. Call this matrix $\mathbf{L}_{a1}$ and its rows $\mathrm{l}_{a1}^{(1)}$, $\mathrm{l}_{a1}^{(2)}$, and $\mathrm{l}_{a1}^{(3)}$. The SLICE statement tests the two-degrees-of-freedom hypothesis:

$$H: \begin{cases} \left(\mathrm{l}_{a1}^{(1)} - \mathrm{l}_{a1}^{(2)}\right)\boldsymbol{\beta} = 0 \\ \left(\mathrm{l}_{a1}^{(1)} - \mathrm{l}_{a1}^{(3)}\right)\boldsymbol{\beta} = 0 \end{cases}$$

In a balanced design, where μ_{ij} denotes the mean response if A is at level i and B is at level j, this hypothesis is equivalent to $H: \mu_{11} = \mu_{12} = \mu_{13}$. The DIFF option considers the three rows of $\mathbf{L}_{a1}$ in turn and performs tests of the difference between pairs of rows. By default, all pairwise differences within the subset of $\mathbf{L}$ are considered; in the example, this corresponds to tests of the form

$$H: \left(\mathrm{l}_{a1}^{(1)} - \mathrm{l}_{a1}^{(2)}\right)\boldsymbol{\beta} = 0$$
$$H: \left(\mathrm{l}_{a1}^{(1)} - \mathrm{l}_{a1}^{(3)}\right)\boldsymbol{\beta} = 0$$
$$H: \left(\mathrm{l}_{a1}^{(2)} - \mathrm{l}_{a1}^{(3)}\right)\boldsymbol{\beta} = 0$$

In the example, where $a = 4$ and $b = 3$, this produces four sets of least squares means differences. Within each set, factor A is held fixed at a particular level and each set consists of three comparisons.

Table 17.16 summarizes the *options* available in the SLICE statement.

Table 17.16 SLICE Statement Options

Option	Description
ADJUST=	Determines the method of multiple comparison adjustment of LS-means differences
ALPHA=	Specifies the confidence level
AT	Modifies covariate values in computing LS-means
CL	Constructs *t*-type confidence limits
CORR	Displays the estimated correlation matrix
COV	Displays the estimated covariance matrix
DF=	Specifies the degrees of freedom
DIFF	Displays the differences of the LS-means
E	Displays the matrix coefficients
MEANS	Displays the LS-means slice
NOF	Suppresses the *F* test
SINGULAR=	Tunes the estimability checking
SLICEBY	Partitions the interaction effects

You can specify the following *options* in the SLICE statement after a slash (/):

ADJUST=*method*
performs a multiple comparison adjustment of the *p*-values and confidence limits for the differences of LS-means. The adjusted quantities are produced along with the unadjusted quantities. This option works together with the DIFF option. If you omit the DIFF option, then the ADJUST= option implies the DIFF option except for ADJUST=T.

You can specify the following *methods*:

BON
performs Bonferroni *t* tests of differences between LS-means. This method involves correction factors described in Chapter 53, "The GLM Procedure" (*SAS/STAT User's Guide*), and Chapter 86, "The MULTTEST Procedure" (*SAS/STAT User's Guide*); also see Westfall and Young (1993) and Westfall et al. (1999).

DUNNETT
performs Dunnett's *t* test, which determines whether any treatments are significantly different from a single control for the effects in the SLICES statement. If the LS-means are correlated, PROC LMIXED uses the factor-analytic covariance approximation described in Hsu (1992) and identifies the adjustment as "Dunnett-Hsu" in the results. The approximation derives approximate "effective sample sizes" for which exact critical values are computed.

NELSON
performs Nelson's *t* test of each LS-mean against an average of the LS-means (Ott 1967; Nelson 1982, 1991, 1993). When the LS-means

are correlated, PROC LMIXED uses the factor-analytic covariance approximation described in Hsu (1992) and identifies the adjustment as "Nelson-Hsu" in the results. The approximation derives approximate "effective sample sizes" for which exact critical values are computed.

SCHEFFE performs Scheffé's multiple comparison procedure.

SIDAK performs pairwise *t* tests on differences between LS-means with levels adjusted according to Šidák's inequality. This method involves correction factors described in Chapter 53, "The GLM Procedure" (*SAS/STAT User's Guide*), and Chapter 86, "The MULTTEST Procedure" (*SAS/STAT User's Guide*); also see Westfall and Young (1993) and Westfall et al. (1999).

SIMULATE < (*simoptions***) >** performs the simulation-based multiple comparison procedure. This method computes adjusted *p*-values and confidence limits from the simulated distribution of the maximum or maximum absolute value of a multivariate *t* random vector. All covariance parameters, except the residual scale parameter, are fixed at their estimated values throughout the simulation, potentially resulting in some underdispersion. The simulation estimates q, the true $(1-\alpha)$ quantile, where $1-\alpha$ is the confidence coefficient. For a description of the *simoptions* (except for the CVADJUST option, which you cannot use in the SLICE statement), see the ADJUST= option in the LSMEANS statement.

SMM | GT2 performs pairwise comparisons on the basis of the studentized maximum modulus and Šidák's uncorrelated-*t* inequality, yielding Hochberg's GT2 method when sample sizes are unequal.

T | NONE performs no adjustment for multiple comparisons.

TUKEY performs Tukey's studentized range test on LS-means. When your data are unbalanced, PROC LMIXED uses the approximation described in Kramer (1956) and identifies the adjustment as "Tukey-Kramer" in the results.

By default, ADJUST=T.

If you specify ADJUST=DUNNETT, the procedure analyzes differences between each level and the first control level unless you specify a different control level. If you specify ADJUST=NELSON, then DIFF=ANOM is assumed.

If you specify ADJUST=TUKEY, then DIFF=ALL is assumed. If you specify other *methods*, then the procedure performs all pairwise differences unless you specify the DIFF option.

Note that computing the exact adjusted *p*-values and critical values for unbalanced designs can be computationally intensive, especially when ADJUST=NELSON.

ALPHA=*number*

constructs a *t*-type confidence interval for each LS-means whose confidence level is $1-$ *number*. The value of *number* must be between 0 and 1, exclusive; by default, ALPHA=0.05.

AT *at-specification*

enables you to modify the values of the covariates that are used in computing LS-means. By default, all covariate effects are set equal to their mean values for computation of standard LS-means. The AT option enables you to assign arbitrary values to the covariates. Additional columns in the output table indicate the values of the covariates.

You can specify the following *at-specifications*:

MEANS	sets covariates equal to their mean values (as with standard LS-means) and applies this adjustment to crossproducts of covariates.
variable=*value*	sets the covariate *variable* equal to *value*.
(*variable-list*)=(*value-list*)	sets covariates in the *variable-list* equal to values in the *value-list*.

As an example, consider the following invocation of PROC LMIXED:

```
proc lmixed;
   class A B;
   model Y = A*B x1 x2 x1*x2;
   SLICE A*B;
   SLICE A*B / at means;
   SLICE A*B / at x1=1.2;
   SLICE A*B / at (x1 x2)=(1.2 0.3);
run;
```

For the first two SLICE statements, the LS-means coefficient of x1 is $\overline{x}_1$ (the mean of x1) and of x2 is $\overline{x}_2$ (the mean of x2). For the first SLICE statement, the coefficient of x1*x2 is $\overline{x_1 x_2}$. However, for the second SLICE statement, the coefficient is $\overline{x}_1 \times \overline{x}_2$. The third SLICE statement sets the coefficient of x1 equal to 1.2 and leaves it at $\overline{x}_2$ for x2, and the final SLICE statement sets these values to 1.2 and 0.3, respectively.

Even if you specify a WEIGHT variable, the unweighted covariate means are used for the covariate coefficients if you omit the AT option. If you specify the AT option, then WEIGHT or FREQ variables are taken into account as follows. The weighted covariate means are used for the covariate coefficients for which no explicit AT option values are given, or if you specify the AT MEANS option. Observations that do not contribute to the analysis because of a missing dependent variable are used in computing the covariate means. You should use the E option in conjunction with the AT option to verify that the modified LS-means coefficients are the ones that you want.

CL

constructs *t*-type confidence limits for each of the LS-means. If DDFM=NONE, then PROC LMIXED uses infinite degrees of freedom for this test, essentially computing a z interval. By default, the confidence level is 0.95; you can change this by using the ALPHA= option.

CORR

displays the estimated correlation matrix of the LS-means as part of the "Least Squares Means" table.

COV
displays the estimated covariance matrix of the LS-means as part of the "Least Squares Means" table.

DF=*number*
specifies the degrees of freedom for the *t* test and confidence limits. The default is the residual degrees of freedom that you specify in the DDFM=RESIDUAL option.

DIFF<=*difftype***>**
PDIFF<=*difftype***>**
displays differences of the LS-means. You can specify the following values for the optional *difftype*:

ALL displays all pairwise differences; this is the default.

ANOM displays differences between each LS-mean and the average LS-mean, as in the analysis of means (Ott 1967). The average is computed as a weighted mean of the LS-means, where the weights are inversely proportional to the diagonal entries of the $\mathbf{L}\left(\mathbf{X}'\mathbf{X}\right)^{-}\mathbf{L}'$ matrix. When you specify a WEIGHT statement, this matrix is replaced by $\mathbf{L}\left(\mathbf{X}'\mathbf{W}\mathbf{X}\right)^{-}\mathbf{L}'$, where $\mathbf{W}$ is the diagonal matrix that contains the weights. If the LS-means are nonestimable, this design-based weighted mean is replaced by an equally weighted mean. Note that the ANOM procedure in SAS/QC software implements both tables and graphics for the analysis of means with a variety of response types. For one-way designs and normally distributed data, the DIFF=ANOM computations are equivalent to the results of PROC ANOM.

CONTROL displays differences with a control; by default, the control is the first level of each specified SLICE statement effect. To specify which levels of the effects are the controls, list the quoted formatted values in parentheses after the CONTROL keyword. For example, if the effects A, B, and C are classification variables, and each has two levels (1 and 2), then the following SLICE statement specifies the (1,2) level of A*B and the (2,1) level of B*C as controls:

```
lsmeans A*B B*C / diff=control('1' '2', '2' '1');
```

For multiple effects, the results depend on the order of the list, so you should check the output to make sure that the controls are correct.

The CONTROL keyword produces two-tailed tests and confidence limits.

CONTROLL displays one-tailed test results and performs tests to determine whether the noncontrol levels are significantly smaller than the control. The upper confidence limits for the control minus the noncontrol levels are considered to be infinity and are displayed as missing.

CONTROLU displays one-tailed test results and performs tests to determine whether the noncontrol levels are significantly larger than the control. The upper confidence limits for the noncontrol levels minus the control are considered to be infinity and are displayed as missing.

The differences of the LS-means are displayed in a table titled "Differences of Least Squares Means." The ODS table name is SliceDiffs.

E

displays the matrix coefficients for all specified SLICE statement effects. The name of this table is "Matrix Coefficients." The ODS table name is Coef.

MEANS

displays the LS-means.

NOF

suppresses the *F* test for testing the mutual equality of the estimable functions in the partition.

SINGULAR=*number*

tunes the estimability checking as documented for the SINGULAR= option in the CONTRAST statement.

SLICEBY *sliceby-spec-list*

SLICE *sliceby-spec-list*

determines how to construct the partition of the LS-means for the *model-effect*. The *sliceby-spec-list* is a list of *sliceby-specifications*. A *sliceby-specification* consists of an effect name followed by an optional list of formatted values. The *sliceby-spec-list* takes either of the following two forms:

< **=** > *sliceby-specification*

(*sliceby-specification* < *sliceby-specification* < . . . >>)

If you have only one *sliceby-specification*, use the first form. If you have one or more *sliceby-specifications*, you can use the second form.

The following SLICE statement contains one *sliceby-specification*. It creates partitions of the A*B interaction effect for all levels of variable A.

```
class a b;
model y = a b a*b;
slice a*b / sliceby=a;
```

The following SLICE statement contains one *sliceby-specification*. It creates one partition of the A*B interaction effect for one level, associated with the formatted value 'A1', of variable A.

```
class a b;
model y = a b a*b;
slice a*b / sliceby a='A1';
```

The following SLICE statement contains two *sliceby-specifications*. The first *sliceby-specification* creates two partitions of the A*B interaction effect for two levels, associated with the formatted values 'B2' and 'B3', of variable B. The second *sliceby-specification* creates one partition for the level, associated with the formatted values 'A1', of variable A.

```
class a b;
model y = a b a*b;
slice a*b / sliceby(b=('B2','B3') a='A1') diff;
```

The following SLICE statement contains one *sliceby-specification*. It creates two partitions of the A*B*C interaction effect for two levels of the B*C interaction effect. These two levels are associated with the two combinations of the formatted values: the combination of 'B1' and 'C1' and the combination of 'B1' and 'C2'.

```
class a b c;
model y = a b a*b*c;
slice a*b*c / sliceby b*c=('B1' 'C1', 'B1' 'C2');
```

The following SLICE statement contains two *sliceby-specifications*. The first *sliceby-specification* creates two partitions of the A*B*C interaction effect for two levels that are associated with the two combinations of the formatted values: the combination of 'B1' and 'C1' and the combination of 'B1' and 'C2'. The third *sliceby-specification* creates one partition for one level, associated with the formatted value 'A1', of variable A.

```
class a b c;
model y = a b a*b*c;
slice a*b*c / sliceby(b*c=('B1' 'C1', 'B1' 'C2') a='A1');
```

Notice that an interaction effect of B*C has the internal order of B and C or C and B. The order is determined by the order of these two variables in the CLASS statement. Thus, the order of formatted values in the *sliceby-specification* must follow the order of variables in the CLASS statement. For example, b*c=('B1' 'C1') is correct if B precedes C in the CLASS statement; otherwise, you have to use b*c=('C1' 'B1').

STORE Statement

STORE < **OUT=** >*libref.data-table* **;**

The STORE statement saves the model and results of the statistical analysis. The resulting analytic store has a binary file format that cannot be modified. At a later time you can use the saved results of that analysis to perform specific statistical analysis tasks, without having to fit the model again. The contents of the analytic store can be processed by the ASTORE procedure.

For more information about the ASTORE procedure, see the description in Chapter 4, "ASTORE Procedure" (*SAS Viya: Machine Learning Procedures*).

libref.data-table is a two-level name, where *libref* refers to the library, and *data-table* specifies the name of the output data table. For more information about this two-level name, see the section "Using CAS Sessions and CAS Engine Librefs" on page 864.

TEST Statement

TEST *fixed-effects* < / *options* > **;**

The TEST statement performs hypothesis tests on the fixed effects. You can specify multiple effects in one TEST statement or in multiple TEST statements.

You can specify the following *options* in the TEST statement after a slash (/):

CHISQ
performs χ^2 tests in addition to any F tests. A χ^2 statistic equals its corresponding F statistic times the associated numerator degrees of freedom, and PROC LMIXED uses the same degrees of freedom to compute the p-value for the χ^2 test. This p-value is always less than that for the F test, because a χ^2 test effectively corresponds to an F test that has infinite denominator degrees of freedom.

DF=*number*
specifies the denominator degrees of freedom for the F test. The default is the residual degrees of freedom that you define by specifying the DDFM=RESIDUAL option.

E
displays matrix coefficients that are associated with test types for specified effects.

E3

EIII
displays Type III matrix coefficients if a Type III test is performed.

WEIGHT Statement

WEIGHT *variable* **;**

The *variable* in the WEIGHT statement is used as a weight to perform a weighted analysis of the data. Observations that have nonpositive or missing weights are not included in the analysis. The values of the WEIGHT variable can be nonintegral and are not truncated. When you omit the WEIGHT statement, all observations that are used in the analysis are assigned a weight of 1.

The *variable* in the WEIGHT statement might also be used as a weight to compute the weighted predictive margins. For more information, see the description in the MARGINS statement.

Details: LMIXED Procedure

Linear Mixed Models Theory

This section provides an overview of a likelihood-based approach to linear mixed models. This approach simplifies and unifies many common statistical analyses, including those that involve repeated measures, random effects, and random coefficients. The basic assumption is that the data are linearly related to unobserved multivariate normal random variables. For extensions to nonlinear and nonnormal situations, see the documentation of the GLIMMIX and NLMIXED procedures in the *SAS/STAT User's Guide*. Additional theory and examples are provided in Littell et al. (2006); Verbeke and Molenberghs (1997, 2000); and Burdick and Graybill (1992).

Matrix Notation for General Linear Model

Suppose that you observe n data points $y_1, \ldots, y_n$ and you want to explain them by using n values for each of p explanatory variables $x_{11}, \ldots, x_{1p}, x_{21}, \ldots, x_{2p}, \ldots, x_{n1}, \ldots, x_{np}$. The x_{ij} values can be either regression-type continuous variables or dummy variables that indicate class membership. The general linear model for this setup is

$$y_i = \sum_{j=1}^{p} x_{ij}\beta_j + \epsilon_i \quad i = 1, \ldots, n$$

where $\beta_1, \ldots, \beta_p$ are unknown fixed-effects parameters to be estimated and $\epsilon_1, \ldots, \epsilon_n$ are unknown independent and identically distributed normal (Gaussian) random variables with mean 0 and variance σ^2.

The preceding equations can be written simultaneously by using vectors and a matrix, as follows:

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \ldots & x_{1p} \\ x_{21} & x_{22} & \ldots & x_{2p} \\ \vdots & \vdots & & \vdots \\ x_{n1} & x_{n2} & \ldots & x_{np} \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}$$

For convenience, simplicity, and extendability, this entire system is written as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

where $\mathbf{y}$ denotes the vector of observed y_i's, $\mathbf{X}$ is the known matrix of x_{ij}'s, $\boldsymbol{\beta}$ is the unknown fixed-effects parameter vector, and $\boldsymbol{\epsilon}$ is the unobserved vector of independent and identically distributed Gaussian random errors.

In addition to denoting data, random variables, and explanatory variables in the preceding fashion, the subsequent sections make use of basic matrix operators such as transpose ($'$), inverse ($^{-1}$), generalized inverse ($^{-}$), determinant ($|\cdot|$), and matrix multiplication. See Searle (1982) for details about these and other matrix techniques.

Formulation of the Mixed Model

The previous general linear model is certainly a useful one (Searle 1971), and it is the one fitted by the GLM procedure. However, the distributional assumption about $\boldsymbol{\epsilon}$ is often too restrictive. The mixed model extends the general linear model by allowing a more flexible specification of the covariance matrix of $\boldsymbol{\epsilon}$. In other words, it allows for both correlation and heterogeneous variances, although normality is still assumed.

The mixed model is written as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\boldsymbol{\gamma} + \boldsymbol{\epsilon}$$

where all components are the same as in the general linear model except for the addition of the known design matrix, $\mathbf{Z}$, and the vector of unknown random-effects parameters, $\boldsymbol{\gamma}$. The matrix $\mathbf{Z}$ can contain either continuous or dummy variables, just like $\mathbf{X}$. The name *mixed model* comes from the fact that the model contains both fixed-effects parameters, $\boldsymbol{\beta}$, and random-effects parameters, $\boldsymbol{\gamma}$. See Henderson (1990) and Searle, Casella, and McCulloch (1992) for history of the development of the mixed model.

A key assumption in the preceding analysis is that $\boldsymbol{\gamma}$ and $\boldsymbol{\epsilon}$ are normally distributed with

$$\mathrm{E}\begin{bmatrix} \boldsymbol{\gamma} \\ \boldsymbol{\epsilon} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}$$

$$\mathrm{Var}\begin{bmatrix} \boldsymbol{\gamma} \\ \boldsymbol{\epsilon} \end{bmatrix} = \begin{bmatrix} \mathbf{G} & \mathbf{0} \\ \mathbf{0} & \mathbf{R} \end{bmatrix}$$

Therefore, the variance of $\mathbf{y}$ is $\mathbf{V} = \mathbf{Z}\mathbf{G}\mathbf{Z}' + \mathbf{R}$. You can model $\mathbf{V}$ by setting up the random-effects design matrix $\mathbf{Z}$ and specifying covariance structures for $\mathbf{G}$ and $\mathbf{R}$.

Note that this is a general specification of the mixed model, in contrast to many texts and articles that discuss only simple random effects. Simple random effects are a special case of the general specification in which $\mathbf{Z}$ contains dummy variables, $\mathbf{G}$ contains variance components in a diagonal structure, and $\mathbf{I}$ denotes the identity matrix. The general linear model is a further special case with $\mathbf{Z} = \mathbf{0}$.

The following example illustrates the most common formulation of the general linear mixed model.

Example: Split-Plot Design

The split-plot design involves two experimental treatment factors, A and B, and two different sizes of experimental units to which they are applied (Winer 1971; Snedecor and Cochran 1980; Milliken and Johnson 1992; Steel, Torrie, and Dickey 1997). The levels of A are randomly assigned to the larger-sized experimental units, called *whole plots*, whereas the levels of B are assigned to the smaller-sized experimental units, the *subplots*. The subplots are assumed to be nested within the whole plots, so that a whole plot consists of a cluster of subplots and a level of A is applied to the entire cluster.

Such an arrangement is often necessary by nature of the experiment; the classic example is the application of fertilizer to large plots of land and different crop varieties that are planted in subdivisions of the large plots. For this example, fertilizer is the whole-plot factor A and variety is the subplot factor B.

This example is a split-plot design for which the whole plots are arranged in a randomized block design. The appropriate PROC LMIXED statements are as follows:

```
proc lmixed;
   class a b block;
   model y = a b a*b;
   random block a*block;
run;
```

Here

$$\mathbf{R} = \sigma^2 \mathbf{I}_{24}$$

and $\mathbf{X}$, $\mathbf{Z}$, and $\mathbf{G}$ have the following forms:

$$\mathbf{X} = \begin{bmatrix}
1 & 1 & & & 1 & & 1 & & & & & \\
1 & 1 & & & & 1 & & 1 & & & & \\
1 & & 1 & & 1 & & & & 1 & & & \\
1 & & 1 & & & 1 & & & & 1 & & \\
1 & & & 1 & 1 & & & & & & 1 & \\
1 & & & 1 & & 1 & & & & & & 1 \\
\vdots & & \vdots & & \vdots & & & & \vdots & & & \\
1 & 1 & & & 1 & & 1 & & & & & \\
1 & 1 & & & & 1 & & 1 & & & & \\
1 & & 1 & & 1 & & & & 1 & & & \\
1 & & 1 & & & 1 & & & & 1 & & \\
1 & & & 1 & 1 & & & & & & 1 & \\
1 & & & 1 & & 1 & & & & & & 1
\end{bmatrix}$$

$$\mathbf{Z} = \begin{bmatrix}
1 & & 1 & & & & & & & \\
1 & & 1 & & & & & & & \\
1 & & & 1 & & & & & & \\
1 & & & 1 & & & & & & \\
1 & & & & 1 & & & & & \\
1 & & & & 1 & & & & & \\
 & 1 & & & & 1 & & & & \\
 & 1 & & & & 1 & & & & \\
 & 1 & & & & & 1 & & & \\
 & 1 & & & & & 1 & & & \\
 & 1 & & & & & & 1 & & \\
 & 1 & & & & & & 1 & & \\
 & & 1 & & & & & & 1 & \\
 & & 1 & & & & & & 1 & \\
 & & 1 & & & & & & & 1 \\
 & & 1 & & & & & & & 1
\end{bmatrix}$$

$$\mathbf{G} = \begin{bmatrix}
\sigma^2_B & & & & & & & \\
 & \sigma^2_B & & & & & & \\
 & & \sigma^2_B & & & & & \\
 & & & \sigma^2_B & & & & \\
 & & & & \sigma^2_{AB} & & & \\
 & & & & & \sigma^2_{AB} & & \\
 & & & & & & \ddots & \\
 & & & & & & & \sigma^2_{AB}
\end{bmatrix}$$

where σ^2_B is the variance component for Block and σ^2_{AB} is the variance component for A*Block. Changing the RANDOM statement as follows fits the same model, but with **Z** and **G** sorted differently:

```
random int a / subject=block;
```

$$\mathbf{Z} = \begin{bmatrix}
1 & 1 & & & & & & & & & & & & & & & & & & \\
1 & 1 & & & & & & & & & & & & & & & & & & \\
1 & & 1 & & & & & & & & & & & & & & & & & \\
1 & & 1 & & & & & & & & & & & & & & & & & \\
1 & & & 1 & & & & & & & & & & & & & & & & \\
1 & & & 1 & & & & & & & & & & & & & & & & \\
 & & & & 1 & 1 & & & & & & & & & & & & & & \\
 & & & & 1 & 1 & & & & & & & & & & & & & & \\
 & & & & 1 & & 1 & & & & & & & & & & & & & \\
 & & & & 1 & & 1 & & & & & & & & & & & & & \\
 & & & & 1 & & & 1 & & & & & & & & & & & & \\
 & & & & 1 & & & 1 & & & & & & & & & & & & \\
 & & & & & & & & 1 & 1 & & & & & & & & & & \\
 & & & & & & & & 1 & 1 & & & & & & & & & & \\
 & & & & & & & & 1 & & 1 & & & & & & & & & \\
 & & & & & & & & 1 & & 1 & & & & & & & & & \\
 & & & & & & & & 1 & & & 1 & & & & & & & & \\
 & & & & & & & & 1 & & & 1 & & & & & & & & \\
 & & & & & & & & & & & & 1 & 1 & & & & & & \\
 & & & & & & & & & & & & 1 & 1 & & & & & & \\
 & & & & & & & & & & & & 1 & & 1 & & & & & \\
 & & & & & & & & & & & & 1 & & 1 & & & & & \\
 & & & & & & & & & & & & 1 & & & 1 & & & & \\
 & & & & & & & & & & & & 1 & & & 1 & & & &
\end{bmatrix}$$

$$\mathbf{G} = \begin{bmatrix}
\sigma^2_B & & & & & & & & & & & & & & & & & & & \\
 & \sigma^2_{AB} & & & & & & & & & & & & & & & & & & \\
 & & \sigma^2_{AB} & & & & & & & & & & & & & & & & & \\
 & & & \sigma^2_{AB} & & & & & & & & & & & & & & & & \\
 & & & & \ddots & & & & & & & & & & & & & & & \\
 & & & & & \sigma^2_B & & & & & & & & & & & & & & \\
 & & & & & & \sigma^2_{AB} & & & & & & & & & & & & & \\
 & & & & & & & \sigma^2_{AB} & & & & & & & & & & & & \\
 & & & & & & & & \sigma^2_{AB} & & & & & & & & & & \\
 & & & & & & & & & & & & & & & & & & & \\
 & & & & & & & & & & & & & & & & & & & \\
 & & & & & & & & & & & & & & & & & & & \\
 & & & & & & & & & & & & & & & & & & & \\
 & & & & & & & & & & & & & & & & & & & \\
 & & & & & & & & & & & & & & & & & & & \\
 & & & & & & & & & & & & & & & & & & & \\
 & & & & & & & & & & & & & & & & & & & \\
 & & & & & & & & & & & & & & & & & & &
\end{bmatrix}$$

Estimating Covariance Parameters in the Mixed Model

Estimation is more difficult in the mixed model than in the general linear model. Not only do you have $\boldsymbol{\beta}$ as in the general linear model, but you also have unknown parameters in $\boldsymbol{\gamma}$, $\mathbf{G}$, and $\mathbf{R}$. Least squares is no longer the best method. Generalized least squares (GLS) is more appropriate; it minimizes

$$(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})'\mathbf{V}^{-1}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$$

However, GLS requires knowledge of $\mathbf{V}$ and therefore knowledge of $\mathbf{G}$ and $\mathbf{R}$. When such information is lacking, one approach is to use an *estimated* GLS, in which you insert some reasonable estimate for $\mathbf{V}$ into the minimization problem. The goal thus becomes to find a reasonable estimate of $\mathbf{G}$ and $\mathbf{R}$.

In many situations, the best approach is to use likelihood-based methods, which exploit the assumption that $\boldsymbol{\gamma}$ and $\boldsymbol{\epsilon}$ are normally distributed (Hartley and Rao 1967; Patterson and Thompson 1971; Harville 1977; Laird and Ware 1982; Jennrich and Schluchter 1986). The LMIXED procedure implements two likelihood-based methods: maximum likelihood (ML) and restricted (residual) maximum likelihood (REML). A favorable theoretical property of ML and REML is that they accommodate data that are missing at random (Rubin 1976; Little 1995).

The LMIXED procedure constructs an objective function that is associated with ML or REML and maximizes it over all unknown parameters. Using calculus, it is possible to reduce this maximization problem to one over only the parameters in $\mathbf{G}$ and $\mathbf{R}$. The corresponding log-likelihood functions are as follows:

$$\text{ML}: \quad l(\mathbf{G},\mathbf{R}) = -\frac{1}{2}\log|\mathbf{V}| - \frac{1}{2}\mathbf{r}'\mathbf{V}^{-1}\mathbf{r} - \frac{n}{2}\log(2\pi)$$

$$\text{REML}: \quad l_R(\mathbf{G},\mathbf{R}) = -\frac{1}{2}\log|\mathbf{V}| - \frac{1}{2}\log|\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}| - \frac{1}{2}\mathbf{r}'\mathbf{V}^{-1}\mathbf{r} - \frac{n-p}{2}\log(2\pi)$$

where $\mathbf{r} = \mathbf{y} - \mathbf{X}(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X})^{-}\mathbf{X}'\mathbf{V}^{-1}\mathbf{y}$, p is the rank of $\mathbf{X}$, and n is the total number of observations. By default, the LMIXED procedure minimizes a normalized form of -2 times these functions by using a ridge-stabilized Newton-Raphson algorithm. Lindstrom and Bates (1988) provide reasons for preferring Newton-Raphson to the expectation-maximum (EM) algorithm that is described in Dempster, Laird, and Rubin (1977) and Laird, Lange, and Stram (1987); in addition, they provide analytical details for implementing a QR-decomposition approach to the problem. Wolfinger, Tobias, and Sall (1994) present the sweep-based algorithms that are implemented in the LMIXED procedure. You can change the optimization technique by specifying the TECHNIQUE= option in the OPTIMIZATION statement.

One advantage of using the Newton-Raphson algorithm is that the second derivative matrix of the objective function that is evaluated at the optima is available upon completion. Denote this matrix as $\mathbf{H}$; then the asymptotic theory of maximum likelihood (Serfling 1980) shows that $2\mathbf{H}^{-1}$ is an asymptotic variance-covariance matrix of the estimated parameters of $\mathbf{G}$ and $\mathbf{R}$.

The residual variance σ^2 is *profiled* out of the likelihood by default. This means solving analytically for the optimal σ^2 and plugging this expression back into the likelihood formula (Wolfinger, Tobias, and Sall 1994). This reduces the number of optimization parameters by 1 and can improve convergence properties.

Instead of ML or REML, you can use the noniterative MIVQUE0 method to estimate $\mathbf{G}$ and $\mathbf{R}$ (Rao 1972; LaMotte 1973; Wolfinger, Tobias, and Sall 1994). In fact, by default PROC LMIXED uses MIVQUE0 estimates as starting values for the ML and REML methods when DMMETHOD=DENSE. However, Swallow and Monahan (1984) present simulation evidence that favors REML and ML over MIVQUE0.

Computing Methodologies for REML in the Mixed Model

As a high-performance procedure, PROC LMIXED can use the distributed and multithreading capabilities of your system (for more information, see the section "Computational Method" on page 944) to speed up the fitting process in some linear mixed models that use the restricted maximum likelihood (REML) and the maximum likelihood (ML) approaches. This section illustrates how to use multithreaded computing or the sparse matrix technique to compute the objective function in REML.

The restricted log likelihood is defined as

$$\text{REML}: \quad l_R(\mathbf{G},\mathbf{R}) = -\frac{1}{2}\log|\mathbf{V}| - \frac{1}{2}\log|\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}| - \frac{1}{2}\mathbf{r}'\mathbf{V}^{-1}\mathbf{r} - \frac{n-p}{2}\log(2\pi)$$

where $\mathbf{r} = \mathbf{y} - \mathbf{X}(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X})^{-}\mathbf{X}'\mathbf{V}^{-1}\mathbf{y}$, p is the rank of $\boldsymbol{X}$, and n is the total number of observations.

The computational complexity resides mostly in calculating these components: $\log|\mathbf{V}|$, $\log|\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}|$, and $\mathbf{r}'\mathbf{V}^{-1}\mathbf{r}$. In directly handling a matrix $\mathbf{V}$ of dimension $n \times n$, a large value of n poses difficulties in both storage and computation.

In some situations, the computation of these $\mathbf{V}$-related quantities can be greatly simplified, thus enabling parallel computing. When you have a common subject effect, the data can be subsetted and worked with in a smaller size. This leads to simplification of the REML function components,

$$\log|\mathbf{V}| = \sum_{i=1}^{s} \log|V_i|$$

$$\log|\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}| = \log|\sum_{i=1}^{s}(X_i'V_i^{-1}X_i)|$$

where s is the number of levels in the common subject effect and V_i is the covariance matrix of response variables that belong to subject level i. Because summations enable parallel accumulation of the individual pieces, they are ideal for multithreaded and distributed computing. The $\mathbf{r}'\mathbf{V}^{-1}\mathbf{r}$ term is calculated as a by-product of the determinant of $\sum_{i=1}^{s}(X_i'V_i^{-1}X_i)$. This method is called the dense method and can be specified in the DMMETHOD=DENSE option in the PROC LMIXED statement.

If the model does not have a common subject effect, the restricted log likelihood is computed using the mixed model equations (Henderson 1984). The three components can be rewritten as:

$$\log|\mathbf{V}| + \log|\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}| = \log|\mathbf{M}| + \log|\mathbf{G}| + \log|\mathbf{R}|$$

$$\mathbf{r}'\mathbf{V}^{-1}\mathbf{r} = \mathbf{y}'\mathbf{R}^{-1}\mathbf{y} - \widehat{\boldsymbol{\beta}}'\mathbf{X}'\mathbf{R}^{-1}\mathbf{y} - \widehat{\boldsymbol{\gamma}}'\mathbf{Z}'\mathbf{R}^{-1}\mathbf{y}$$

where $\mathbf{M}$ is

$$\begin{bmatrix} \mathbf{X}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{X}'\mathbf{R}^{-1}\mathbf{Z} \\ \mathbf{Z}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{Z}'\mathbf{R}^{-1}\mathbf{Z} + \mathbf{G}^{-1} \end{bmatrix}$$

Generally speaking, matrix $\mathbf{M}$ is large and computing it requires substantial memory. Fortunately, $\mathbf{M}$ is often very sparse in many real-world problems if you work with categorical variables that have many levels (sparsity in $\mathbf{X}$ and $\mathbf{Z}$). The LMIXED procedure uses sparse matrix techniques to compute the objective function; these techniques can be substantially more efficient than the dense approach. This method is called the sparse method and can be specified in the DMMETHOD=SPARSE option in the PROC LMIXED statement.

Estimating Fixed and Random Effects in the Mixed Model

The maximum likelihood (ML) and the restricted maximum likelihood (REML) methods provide estimates of $\mathbf{G}$ and $\mathbf{R}$, which are denoted $\widehat{\mathbf{G}}$ and $\widehat{\mathbf{R}}$, respectively. To obtain estimates of $\boldsymbol{\beta}$ and predicted values of $\boldsymbol{\gamma}$, the standard method is to solve the mixed model equations (Henderson 1984):

$$\begin{bmatrix} \mathbf{X}'\widehat{\mathbf{R}}^{-1}\mathbf{X} & \mathbf{X}'\widehat{\mathbf{R}}^{-1}\mathbf{Z} \\ \mathbf{Z}'\widehat{\mathbf{R}}^{-1}\mathbf{X} & \mathbf{Z}'\widehat{\mathbf{R}}^{-1}\mathbf{Z} + \widehat{\mathbf{G}}^{-1} \end{bmatrix} \begin{bmatrix} \widehat{\boldsymbol{\beta}} \\ \widehat{\boldsymbol{\gamma}} \end{bmatrix} = \begin{bmatrix} \mathbf{X}'\widehat{\mathbf{R}}^{-1}\mathbf{y} \\ \mathbf{Z}'\widehat{\mathbf{R}}^{-1}\mathbf{y} \end{bmatrix}$$

Assume that the covariance matrix of response variables is $\widehat{\mathbf{V}}$. Then the solutions can be written as

$$\widehat{\boldsymbol{\beta}} = (\mathbf{X}'\widehat{\mathbf{V}}^{-1}\mathbf{X})^{-}\mathbf{X}'\widehat{\mathbf{V}}^{-1}\mathbf{y}$$

$$\widehat{\boldsymbol{\gamma}} = \widehat{\mathbf{G}}\mathbf{Z}'\widehat{\mathbf{V}}^{-1}(\mathbf{y} - \mathbf{X}\widehat{\boldsymbol{\beta}})$$

and have connections with empirical Bayes estimators (Laird and Ware 1982; Carlin and Louis 1996). Note that the $\boldsymbol{\gamma}$ are random variables and not parameters (unknown constants) in the model. Technically, determining values for $\boldsymbol{\gamma}$ from the data is thus a prediction task, whereas determining values for $\boldsymbol{\beta}$ is an estimation task.

Statistical Properties

If $\mathbf{G}$ and $\mathbf{R}$ are known, $\widehat{\boldsymbol{\beta}}$ is the best linear unbiased estimator (BLUE) of $\boldsymbol{\beta}$, and $\widehat{\boldsymbol{\gamma}}$ is the best linear unbiased predictor (BLUP) of $\boldsymbol{\gamma}$ (Searle 1971; Harville 1988, 1990; Robinson 1991; McLean, Sanders, and Stroup 1991). Here, "best" means it has minimum mean square error. The covariance matrix of $(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}, \widehat{\boldsymbol{\gamma}} - \boldsymbol{\gamma})$ is

$$\mathbf{C} = \begin{bmatrix} \mathbf{X}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{X}'\mathbf{R}^{-1}\mathbf{Z} \\ \mathbf{Z}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{Z}'\mathbf{R}^{-1}\mathbf{Z} + \mathbf{G}^{-1} \end{bmatrix}^{-}$$

where $^{-}$ denotes a generalized inverse (Searle 1971).

However, $\mathbf{G}$ and $\mathbf{R}$ are usually unknown and are estimated by using one of the aforementioned methods. These estimates, $\widehat{\mathbf{G}}$ and $\widehat{\mathbf{R}}$, are therefore simply substituted into the preceding expression to obtain

$$\widehat{\mathbf{C}} = \begin{bmatrix} \mathbf{X}'\widehat{\mathbf{R}}^{-1}\mathbf{X} & \mathbf{X}'\widehat{\mathbf{R}}^{-1}\mathbf{Z} \\ \mathbf{Z}'\widehat{\mathbf{R}}^{-1}\mathbf{X} & \mathbf{Z}'\widehat{\mathbf{R}}^{-1}\mathbf{Z} + \widehat{\mathbf{G}}^{-1} \end{bmatrix}^{-}$$

as the approximate variance-covariance matrix of $(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}, \widehat{\boldsymbol{\gamma}} - \boldsymbol{\gamma})$. In this case, the BLUE and BLUP acronyms no longer apply, but the word *empirical* is often added to indicate such an approximation. The appropriate acronyms thus become EBLUE and EBLUP.

McLean and Sanders (1988) show that $\widehat{\mathbf{C}}$ can also be written as

$$\widehat{\mathbf{C}} = \begin{bmatrix} \widehat{\mathbf{C}}_{11} & \widehat{\mathbf{C}}_{21}' \\ \widehat{\mathbf{C}}_{21} & \widehat{\mathbf{C}}_{22} \end{bmatrix}$$

where

$$\widehat{\mathbf{C}}_{11} = (\mathbf{X}'\widehat{\mathbf{V}}^{-1}\mathbf{X})^{-}$$
$$\widehat{\mathbf{C}}_{21} = -\widehat{\mathbf{G}}\mathbf{Z}'\widehat{\mathbf{V}}^{-1}\mathbf{X}\widehat{\mathbf{C}}_{11}$$
$$\widehat{\mathbf{C}}_{22} = (\mathbf{Z}'\widehat{\mathbf{R}}^{-1}\mathbf{Z} + \widehat{\mathbf{G}}^{-1})^{-1} - \widehat{\mathbf{C}}_{21}\mathbf{X}'\widehat{\mathbf{V}}^{-1}\mathbf{Z}\widehat{\mathbf{G}}$$

Note that $\widehat{\mathbf{C}}_{11}$ is the familiar estimated generalized least squares formula for the variance-covariance matrix of $\widehat{\boldsymbol{\beta}}$.

Common Subject Effect

PROC LMIXED is designed to fit a subclass of large (in number of observations and number of levels in a subject) linear mixed models. The procedure takes advantage of distributed and multithreading capabilities of your system by working on subsets of the data in parallel, and it aggregates results in the estimation process, which can lead to substantial speed improvements. This subclass of models requires that the $\mathbf{G}$ matrix have a block-diagonal structure, meaning that the data can be grouped according to a subject effect. This subject effect is referred to as a "common subject effect."

The common subject effect should be the only subject effect that is used in a program, and it must be specified as the same SUBJECT= effect in all RANDOM statements.

For example, a model that has a single RANDOM statement with a SUBJECT= specification has a common subject effect by definition:

```
random intercept age / subject=student;
```

You can have multiple RANDOM statements. The same SUBJECT= effect is the common subject:

```
random time          / subject=student*school;
random intercept age / subject=student*school;
```

In contrast, a single RANDOM statement without SUBJECT= specification, as shown in the following statements, implies that the model does not have a common subject effect:

```
random time;
```

More than one subject effect appearing in a program, as shown in the following statements, implies that the model does not have a common subject effect:

```
random time          / subject=student*school;
random intercept age / subject=student;
```

If you have a variance component model (TYPE=VC), in some situations, you can use a random effect as a subject effect, which leads to the same model specification but with a common subject effect. For example, the following two RANDOM statements are equivalent (the second specification has a common subject effect in time):

```
random time;

random int / subject=time;
```

Another equivalency is that

```
random time          / subject=student*school;
random intercept age / subject=student;
```

can be rewritten as

```
random time*school     / subject=student;
random intercept   age / subject=student;
```

where the student becomes the common subject effect.

Computational Method

Distributed Computing

Distributed computing refers to the use of multiple autonomous computers that communicate through a secure network. Distributed computing solves computational problems by dividing them into many tasks, each of which is solved by one or more computers. Each computer in this distributed environment is called a node.

Multithreading

The LMIXED procedure allocates data to different threads and calculates design matrices by accumulating the contributions from all threads. The LMIXED procedure also computes the log likelihood by accumulating the contributions from all threads. In addition, operations on matrices such as sweeps might be multithreaded if the matrices are of sufficient size to realize performance benefits from managing multiple threads for the particular matrix operation.

For more information about how the LMIXED procedure uses threads, see the section "Multithreading" on page 102 in Chapter 2, "Shared Concepts."

Displayed Output

The following sections describe the output that the LMIXED procedure produces. The output is organized into various tables, which are discussed in the order of their appearance.

Model Information

The "Model Information" table describes the model, some of the variables it involves, and the method used in fitting it. The "Model Information" table also has a row labeled Fixed Effects SE Method. This row describes the method that was used to compute the approximate standard errors for the fixed-effects parameter estimates and related functions of them.

Class Level Information

The "Class Level Information" table lists the levels of every variable that is specified in the CLASS statement.

Dimensions

The "Dimensions" table lists the sizes of relevant matrices. This table can be useful in determining the requirements for CPU time and memory.

Number of Observations

The "Number of Observations" table shows the number of observations that were read from the data table and the number of observations that were used in fitting the model.

Optimization Information

The "Optimization Information" table displays important details about the optimization process.

The number of parameters that are updated in the optimization equals the number of parameters in this table minus the number of equality constraints. The number of constraints is displayed if you fix covariance parameters by specifying the HOLD= option in the PARMS statement. The LMIXED procedure also lists the number of upper and lower boundary constraints. The LMIXED procedure might impose boundary constraints for certain parameters, such as variance components and correlation parameters. If you specify the HOLD= option in the PARMS statement, covariance parameters have an upper and lower boundary equal to the parameter value.

Descriptive Statistics

The "Descriptive Statistics" table lists simple statistics such as means and standard deviations for the dependent variable, for each covariate in the MODEL statement, and for the weight variable in the WEIGHT statement.

Iteration History

The "Iteration History" table describes the optimization of the restricted log likelihood or log likelihood. The function to be minimized (the *objective function*) is $-2l$ for ML and $-2l_R$ for REML; the column name of the objective function in the "Iteration History" table is "-2 Log Like" for ML and "-2 Res Log Like" for REML. The minimization is performed by using a ridge-stabilized Newton-Raphson algorithm, and the rows of this table describe the iterations that this algorithm takes in order to minimize the objective function.

The Evaluations column of the "Iteration History" table tells how many times the objective function is evaluated during each iteration.

The Criterion column of the "Iteration History" table is, by default, a relative Hessian convergence quantity,

$$\frac{\mathbf{g}_k' \mathbf{H}_k^{-1} \mathbf{g}_k}{|f_k|}$$

where f_k is the value of the objective function at iteration k, $\mathbf{g}_k$ is the gradient (first derivative) of f_k, and $\mathbf{H}_k$ is the Hessian (second derivative) of f_k. If $\mathbf{H}_k$ is singular, then the LMIXED procedure uses the following relative quantity:

$$\frac{\mathbf{g}_k' \mathbf{g}_k}{|f_k|}$$

To prevent division by $|f_k|$, specify the ABSGCONV= option in the OPTIMIZATION statement. To use a relative function or gradient criterion, specify the FCONV= or GCONV= option, respectively.

The Hessian criterion is considered superior to function and gradient criteria because it measures orthogonality rather than lack of progress (Bates et al. 1987). Provided that the initial estimate is feasible and the maximum number of iterations is not exceeded, the Newton-Raphson algorithm is considered to have converged when

the criterion is less than the tolerance specified in the FCONV= or GCONV= option in the OPTIMIZATION statement (or less than the default tolerance value of 1E–8). If convergence is not achieved, the LMIXED procedure displays the estimates of the parameters at the last iteration.

A convergence criterion that is missing indicates that a boundary constraint has been dropped; it is usually not a cause for concern.

Convergence Status

The "Convergence Status" table displays the status of the iterative estimation process at the end of the optimization. The status appears as a message in the listing, and this message is repeated in the log. The ODS object "ConvergenceStatus" also contains several nonprinting columns that can be helpful in checking the success of the iterative process, in particular during batch processing. The Status variable takes on the value 0 for a successful convergence (even if the Hessian matrix is not positive definite). The values 1 and 2 of the Status variable indicate lack of convergence and infeasible initial parameter values, respectively. The variable pdG can be used to check whether the $\mathbf{G}$ matrix is positive definite.

The "Convergence Status" table is not produced for models that are not fit iteratively, such as for models that have no random effects or when the NOITER option is in effect.

Covariance Parameter Estimates

The "Covariance Parameter Estimates" table contains the estimates of the parameters in $\mathbf{G}$ and $\mathbf{R}$. (See the section "Estimating Covariance Parameters in the Mixed Model" on page 939.) Their values are labeled in the table along with Subject information if applicable. The estimates are displayed in the Estimate column and are the results of one of the following estimation methods: REML, ML, and MIVQUE0.

Fit Statistics

The "Fit Statistics" table provides some statistics about the estimated mixed model. Expressions for -2 times the log likelihood are provided in the section "Estimating Covariance Parameters in the Mixed Model" on page 939. If the log likelihood is an extremely large number, then the LMIXED procedure has deemed the estimated $\mathbf{V}$ matrix to be singular. In this case, all subsequent results should be viewed with caution.

In addition, the "Fit Statistics" table lists three information criteria: AIC, AICC, and BIC. All these criteria are in smaller-is-better form and are described in Table 17.17.

Table 17.17 Information Criteria

Criterion	Formula	Reference
AIC	$-2\ell + 2d$	Akaike (1974)
AICC	$-2\ell + 2dn^*/(n^* - d - 1)$	Hurvich and Tsai (1989) Burnham and Anderson (1998)
BIC	$-2\ell + d \log n$ for $n > 0$	Schwarz (1978)

Here ℓ denotes the maximum value of the (possibly restricted) log likelihood; d is the dimension of the model; and n equals the number of effective subjects as displayed in the "Dimensions" table unless this value equals 1, in which case n equals the number of levels of the first random effect that is specified in the first RANDOM statement or the number of levels of the interaction of the first random effect with the noncommon

subject effect that is specified in the first RANDOM statement. If the number of effective subjects equals 1 and you have no RANDOM statements, then n equals the number of valid observations for maximum likelihood estimation or equals $n - p$ for restricted maximum likelihood estimation, where p equals the rank of $\mathbf{X}$. For AICC (a finite-sample corrected version of AIC), n^* equals the number of valid observations for maximum likelihood estimation and $n - p$ equals the number of valid observations for restricted maximum likelihood estimation, unless this number is less than $d + 2$, in which case it equals $d + 2$. When $n = 0$, the value of the BIC is -2ℓ. For restricted likelihood estimation, d equals q, the effective number of estimated covariance parameters. For maximum likelihood estimation, d equals $q + p$.

Timing Information

If you specify the TIMING option in the PROC LMIXED statement, the procedure also produces a "Timing" table in which the elapsed time for each main task of the procedure is displayed.

OutputCasTables Table

The OutputCasTables table is a special table that has information about each CAS table that is created during a CAS action execution. The information for each CAS table consists of the CAS table name, the caslib in which the table resides, and the number of columns and rows in the CAS table. Because this table is not a typical ODS table that contains analytical results, you cannot include it in the *table-spec-list* in the DISPLAYOUT statement.

ODS Table Names

Each table that the LMIXED procedure creates has a name associated with it, and you must use this name to refer to the table when you use ODS statements. These names are listed in Table 17.18.

NOTE: The EFFECT statement also create tables which are not listed in this section. For information about these tables, see "ODS Table Names" on page 33 in Chapter 2, "Shared Concepts."

Table 17.18 ODS Tables Produced by PROC LMIXED

Table Name	Description	Required Statements	Required Option
Anova	Anova table for model without random effects	Default output for fixed models	
CholR	Cholesky root of blocks of the estimated **R** matrix	REPEATED	RC
ClassInfo	Level information from the CLASS	CLASS	
CollectionLevelInfo	Level information for collection effects	EFFECT with COLLECTION *effect-type*	DETAILS
Coef	**L** matrix coefficients	CONTRAST, ESTIMATE	E
Contrasts	Results from the CONTRAST statements	CONTRAST	
ConvergenceStatus	Convergence status	Default output	

Table 17.18 *continued*

Table Name	Description	Required Statement	Required Option
CovParms	Estimated covariance parameters	Default output	
Diffs	Differences of LS-means	LSMEANS	DIFF or ADJUST
Dimensions	Dimensions of the model	Default output	
Estimates	Results from ESTIMATE statements	ESTIMATE	
FitStatistics	Fit statistics	Default output	
InvCholR	Inverse Cholesky root of blocks of the estimated **R** matrix	REPEATED	RCI
InvR	Inverse of blocks of the estimated **R** matrix	REPEATED	RI
IterHistory	Iteration history	Default output	
Margins	Predictive margins	MARGINS	
MarginsDiffs	Differences of predictive margins	MARGINS	DIFF=
MarginsSliceDiffs	Differences of predictive margins by slices	MARGINS	SLICEBY= and SLICEDIFF=
MarginsSliceTests	The *F* test for predictive margins by slices	MARGINS	SLICEBY=
LSMeans	LS-means	LSMEANS	
LSMEstimates	LS-means estimates	LSMESTIMATE	
LSMFTest	LS-means *F* test	LSMESTIMATE	FTEST
MMEq	Mixed model equations	PROC LMIXED	MMEQ
MMLevelInfo	Level information for multimember effects	EFFECT with MULTIMEMBER *effect-type*	DETAILS
ModelInfo	Model information	Default output	
NObs	Number of observations read and used	Default output	
OptInfo	Optimization information	Default output	
ParameterEstimates	Fixed-effects solution vector	MODEL	SOLUTION
ParmSearch	Parameter search values	PARMS	
PolyDetails	Number of variables and columns, polynomial degree, and standardization method	EFFECT with POLYNOMIAL *effect-type*	DETAILS
Ranks	Rank of designed matrix X	PROC LMIXED	RANKS

Table 17.18 *continued*

Table Name	Description	Required Statement	Required Option
R	Blocks of the estimated **R** matrix	REPEATED	R=
RCorr	Correlation matrix from blocks of the estimated **R** matrix	REPEATED	RCORR=
SimDetails	Details of difference quantile simulation	LSMEANS	ADJUST=SIMULATE(REPORT)
SimResults	Evaluation of difference quantile simulation	LSMEANS	ADJUST=SIMULATE(REPORT)
SimpleStatistics	Descriptive statistics for dependent variable and covariate variables	PROC LMIXED	SIMPLE
Slices	Tests of LS-means slices	LSMEANS	SLICE=
SliceDiffs	Differences of LS-means slices	SLICE	DIFF=
SliceMeans	LS-means by slices	SLICE	MEANS
SliceTests	Tests of LS-means slices	SLICE	
SolutionR	Random-effects solution vector	RANDOM	SOLUTION
SplineKnots	Knot and boundary knot values	EFFECT with SPLINE *effect-type*	BASIS=BSPLINE and DETAILS
Subsets	Listing of the sequential subsets	LSMEANS	ADJUST= STEPDOWN(REPORT)
SubsetSim	Step-down simulation results	LSMEANS	ADJUST=SIMULATE STEPDOWN(REPORT)
Tests3	Type III tests of fixed effects	TEST	
Timing	Timing breakdown by task	PROC LMIXED	TIMING
TPFSplineDetails	Truncated power function spline basis details	EFFECT with SPLINE *effect-type*	BASIS=TPF or BASIS=NATURALCUBIC and DETAILS

Examples: LMIXED Procedure

The examples in this chapter are available in the GitHub repository located at https://github.com/sascommunities/doc-supplement-casstat.

Example 17.1: Mixed Model Analysis of Microarray Data

(View the complete code for this example (lmixedeg1.sas) in the example repository.)

Microarray experiments are an advanced genomic technique that is used in the discovery of new treatments for diseases. Microarray analysis enables tens of thousands of genes to be detected in a single DNA sample. A microarray is a glass slide or membrane that has been spotted or "arrayed" with DNA fragments (oligonucleotides) that represent specific genes. The response of the gene that is detected by a spot is proportional to the intensity of fluorescence associated with that spot. These gene responses can indicate associations with disease conditions, but they can also be affected by systematic biases and different treatments such as sex and genotypes. Statistical models for microarray data attempt to assess the significance and magnitude of gene effects across treatments while adjusting for these systematic biases and to evaluate the significance of differences between treatments.

There are two statistical approaches frequently used in mixed model analysis for microarray data. The first approach is to fit multiple gene-specific models to data that are normalized for systematic biases (Wolfinger et al. 2001; Gibson and Wolfinger 2004). This approach is based on assuming that the biases are independent from the gene effects. If this assumption is untenable, then a second approach fits a single model that combines both the systematic biases and the gene effects (Kerr, Martin, and Churchill 2000; Churchill 2002; Littell et al. 2006). When the number of genes is very large (several hundreds to tens of thousands), this is an analysis for which the sparse matrix approach that the LMIXED procedure implements is well suited.

The following SAS statements simulate a microarray experiment with a so-called loop design structure, which is commonly used in such studies. There are 500 genes, each gene occurs in six arrays, and each array has two dyes. You are interested in the best five and worst five genes based on the best linear unbiased predicted values (BLUP) of these 500 genes.

```
%let narray  = 6;
%let ndye    = 2;
%let nrow    = 4;
%let ngene   = 500;
%let ntrt    = 6;
%let npin    = 4;
%let ndip    = 4;
%let no      = %eval(&ndye*&nrow*&ngene);
%let tno     = %eval(&narray*&no);

data mylib.microarray;
   keep Gene MArray Dye Trt Pin Dip log2i;
   array PinDist{&tno};
   array DipDist{&tno};
   array GeneDist{&tno};
```

```
array ArrayEffect{&narray};
array ArrayGeneEffect{%eval(&narray*&ngene)};
array ArrayDipEffect{%eval(&narray*&ndip)};
array ArrayPinEffect{%eval(&narray*&npin)};

do i = 1 to &tno;
   PinDist{i}  = 1 + int(&npin*ranuni(12345));
   DipDist{i}  = 1 + int(&ndip*ranuni(12345));
   GeneDist{i} = 1 + int(&ngene*ranuni(12345));
end;

igene = 0;
idip = 0;
ipin = 0;
do i = 1 to &narray;
   ArrayEffect{i} = sqrt(0.014)*rannor(12345);
   do j = 1 to &ngene;
      igene = igene+1;
      ArrayGeneEffect{igene} = sqrt(0.0017)*rannor(12345);
   end;
   do j = 1 to &ndip;
      idip = idip + 1;
      ArrayDipEffect{idip} = sqrt(0.0033)*rannor(12345);
   end;
   do j = 1 to &npin;
      ipin = ipin + 1;
      ArrayPinEffect{ipin} = sqrt(0.037)*rannor(12345);
   end;
end;

i = 0;
do MArray = 1 to &narray;
   do Dye = 1 to &ndye;
      do Row = 1 to &nrow;
         do k = 1 to &ngene;
            if MArray=1 and Dye = 1 then do;
               Trt = 0;
               trtc = 0;
            end;
            else do;
               if trtc >= &no then trtc = 0;
               if trtc = 0 then do;
                  Trt = Trt + 1;
                  if Trt >= &ntrt then do;
                     Trt = 0;
                     trtc = 0;
                  end;
               end;
               trtc = trtc + 1;
            end;
            i = i + 1;
            Pin = PinDist{i};
            Dip = DipDist{i};
            Gene = GeneDist{i};
```

```
                a   = ArrayEffect{MArray};
                ag  = ArrayGeneEffect{(MArray-1)*&ngene+Gene};
                ad  = ArrayDipEffect{(MArray-1)*&ndip+Dip};
                ap  = ArrayPinEffect{(MArray-1)*&npin+Pin};
                log2i   = 1 +
                         + Dye
                         + Trt
                         + Gene/1000.0
                         + Dye*Gene/1000.0
                         + Trt*Gene/1000.0
                         + Pin
                         + a
                         + ag
                         + ad
                         + ap
                         + sqrt(0.02)*rannor(12345);
                output;
             end;
          end;
       end;
    end;
run;
```

A linear mixed model for fitting the log intensity data Y_{ijkmnr} from such a design is described by Littell et al. (2006) as follows:

Y_{ijkmnr}	$=$		**Fixed Effects**
		μ	Overall mean
	$+$	λ_i	Gene
	$+$	τ_j	Treatment
	$+$	δ_k	Dye
	$+$	$(\tau\lambda)_{ij}$	Treatment-by-gene
	$+$	$(\delta\lambda)_{ik}$	Dye-by-gene
	$+$	p_r	Pin
			Random Effects
	$+$	a_m	Microarray
	$+$	$(a\lambda)_{im}$	Microarray-by-gene
	$+$	$d(a)_{mn}$	Dip-within-microarray
	$+$	$(ap)_{mr}$	Microarray-by-pin
	$+$	e_{ijkmnr}	Residual noise

You can use the LMIXED procedure with the following statements to fit this model:

```
ods select Dimensions covParms SolutionR;
proc lmixed data=mylib.microarray dmmethod=sparse;
   class marray dye trt gene pin dip;
   model log2i = dye trt gene dye*gene trt*gene pin;
   random int gene dip pin/subject=marray s;
   ods output solutionr=BLUPs;
run;
```

By default, because the program has a RANDOM statement with a SUBJECT= effect, PROC LMIXED uses the dense method (see the section "Common Subject Effect" on page 943) in the restricted maximum

likelihood (REML) estimation. However, because the design matrices are excessively large with **X** having 4,513 columns and **Z** having 3,054 columns (see table Output 17.1.1), the program can potentially exceed your computer's memory capacity. Here the sparse method (DMMETHOD=SPARSE) is used to improve the handling of these large and sparse design matrices.

Output 17.1.1 Mixed Model Dimensions

The LMIXED Procedure

Dimensions	
G-side Covariance Parameters	4
R-side Covariance Parameters	1
Columns in X	4513
Columns in Z per Subject	509
Subjects (Blocks in V)	6

Output 17.1.2 shows the "Covariance Parameter Estimation" table, which lists the estimations of covariances across each microarray.

Output 17.1.2 Covariance Parameter Estimation

Covariance Parameter Estimates		
Cov Parm	Subject	Estimate
Intercept	MArray	0.003096
Gene	MArray	0.001901
Dip	MArray	0.001924
Pin	MArray	0.03371
Residual		0.02008

The following program ranks genes on the basis of their BLUP estimates.

```
ods output off;
proc sort data=BLUPs(firstobs=2 obs=501);
    by descending Estimate;
run;

data BLUPs; set BLUPs;
   Rank = _N_;
run;
ods output on;

proc print data=BLUPs;
   where ((Rank <= 5) | (Rank >= 496));
   var Gene Estimate;
run;
```

The best five genes and worst five genes with their BLUP estimates are shown in Output 17.1.3.

Output 17.1.3 Highest and Lowest Gene BLUPs

Obs	Gene	Estimate
1	198	0.06753
2	268	0.06071
3	354	0.05124
4	489	0.04883
5	478	0.04558
496	383	-0.04642
497	333	-0.04684
498	351	-0.04846
499	54	-0.05353
500	144	-0.05541

References

Akaike, H. (1974). "A New Look at the Statistical Model Identification." *IEEE Transactions on Automatic Control* AC-19:716–723.

Bates, D. M., Lindstrom, M. J., Wahba, G., and Yandell, B. S. (1987). "GCVPACK—Routines for Generalized Cross Validation." *Communications in Statistics—Simulation and Computation* 16:263–297.

Browne, M. W. (1982). "Covariance Structures." In *Topics in Applied Multivariate Analysis*, edited by D. M. Hawkins, 72–141. Cambridge: Cambridge University Press.

Burdick, R. K., and Graybill, F. A. (1992). *Confidence Intervals on Variance Components*. New York: Marcel Dekker.

Burnham, K. P., and Anderson, D. R. (1998). *Model Selection and Inference: A Practical Information-Theoretic Approach*. New York: Springer-Verlag.

Carlin, B. P., and Louis, T. A. (1996). *Bayes and Empirical Bayes Methods for Data Analysis*. London: Chapman & Hall.

Chang, I.-M., Gelman, R., and Pagano, M. (1982). "Corrected Group Prognostic Curves and Summary Statistics." *Journal of Chronic Diseases* 35:669–674.

Churchill, G. A. (2002). "Fundamentals of Experimental Design for cDNA Microarray." *Nature Genetics* 32:490–495.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). "Maximum Likelihood from Incomplete Data via the EM Algorithm." *Journal of the Royal Statistical Society, Series B* 39:1–38.

Edwards, D., and Berry, J. J. (1987). "The Efficiency of Simulation-Based Multiple Comparisons." *Biometrics* 43:913–928.

Fuller, W. A. (1976). *Introduction to Statistical Time Series*. New York: John Wiley & Sons.

Gibson, G., and Wolfinger, R. D. (2004). "Gene Expression Profiling Using Mixed Models." In *Genetic Analysis of Complex Traits Using SAS*, edited by A. M. Saxton, 251–278. Cary, NC: SAS Institute Inc.

Giesbrecht, F. G. (1989). *A General Structure for the Class of Mixed Linear Models*. Southern Cooperative Series Bulletin 343, Louisiana Agricultural Experiment Station, Baton Rouge.

Hartley, H. O., and Rao, J. N. K. (1967). "Maximum-Likelihood Estimation for the Mixed Analysis of Variance Model." *Biometrika* 54:93–108.

Harville, D. A. (1977). "Maximum Likelihood Approaches to Variance Component Estimation and to Related Problems." *Journal of the American Statistical Association* 72:320–338.

Harville, D. A. (1988). "Mixed-Model Methodology: Theoretical Justifications and Future Directions." In *Proceedings of the Statistical Computing Section*, 41–49. Alexandria, VA: American Statistical Association.

Harville, D. A. (1990). "BLUP (Best Linear Unbiased Prediction), and Beyond." In *Advances in Statistical Methods for Genetic Improvement of Livestock*, edited by D. Gianola and K. Hammond, 239–276. Vol. 18 of Advanced Series in Agricultural Sciences. Berlin: Springer-Verlag.

Henderson, C. R. (1984). *Applications of Linear Models in Animal Breeding*. Guelph, ON: University of Guelph.

Henderson, C. R. (1990). "Statistical Method in Animal Improvement: Historical Overview." In *Advances in Statistical Methods for Genetic Improvement of Livestock*, 1–14. New York: Springer-Verlag.

Holm, S. (1979). "A Simple Sequentially Rejective Multiple Test Procedure." *Scandinavian Journal of Statistics* 6:65–70.

Hsu, J. C. (1992). "The Factor Analytic Approach to Simultaneous Inference in the General Linear Model." *Journal of Computational and Graphical Statistics* 1:151–168.

Hsu, J. C., and Nelson, B. L. (1998). "Multiple Comparisons in the General Linear Model." *Journal of Computational and Graphical Statistics* 7:23–41.

Hurvich, C. M., and Tsai, C.-L. (1989). "Regression and Time Series Model Selection in Small Samples." *Biometrika* 76:297–307.

Huynh, H., and Feldt, L. S. (1970). "Conditions Under Which Mean Square Ratios in Repeated Measurements Designs Have Exact F-Distributions." *Journal of the American Statistical Association* 65:1582–1589.

Jennrich, R. I., and Schluchter, M. D. (1986). "Unbalanced Repeated-Measures Models with Structured Covariance Matrices." *Biometrics* 42:805–820.

Kenward, M. G. (1987). "A Method for Comparing Profiles of Repeated Measurements." *Journal of the Royal Statistical Society, Series C* 36:296–308.

Kerr, M. K., Martin, M., and Churchill, G. A. (2000). "Analysis of Variance for Gene Expression Microarray Data." *Journal of Computational Biology* 7:819–837.

Kramer, C. Y. (1956). "Extension of Multiple Range Tests to Group Means with Unequal Numbers of Replications." *Biometrics* 12:307–310.

Laird, N. M., Lange, N. T., and Stram, D. O. (1987). "Maximum Likelihood Computations with Repeated Measures: Application of the EM Algorithm." *Journal of the American Statistical Association* 82:97–105.

Laird, N. M., and Ware, J. H. (1982). "Random-Effects Models for Longitudinal Data." *Biometrics* 38:963–974.

LaMotte, L. R. (1973). "Quadratic Estimation of Variance Components." *Biometrics* 29:311–330.

Lane, P. W., and Nelder, J. A. (1982). "Analysis of Covariance and Standardization as Instances of Prediction." *Biometrics* 38:613–621.

Lindstrom, M. J., and Bates, D. M. (1988). "Newton-Raphson and EM Algorithms for Linear Mixed-Effects Models for Repeated-Measures Data." *Journal of the American Statistical Association* 83:1014–1022.

Littell, R. C., Milliken, G. A., Stroup, W. W., Wolfinger, R. D., and Schabenberger, O. (2006). *SAS for Mixed Models*. 2nd ed. Cary, NC: SAS Institute Inc.

Little, R. J. A. (1995). "Modeling the Drop-Out Mechanism in Repeated-Measures Studies." *Journal of the American Statistical Association* 90:1112–1121.

Macchiavelli, R. E., and Arnold, S. F. (1994). "Variable Order Ante-dependence Models." *Communications in Statistics—Theory and Methods* 23:2683–2699.

McLean, R. A., and Sanders, W. L. (1988). "Approximating Degrees of Freedom for Standard Errors in Mixed Linear Models." In *Proceedings of the Statistical Computing Section*, 50–59. Alexandria, VA: American Statistical Association.

McLean, R. A., Sanders, W. L., and Stroup, W. W. (1991). "A Unified Approach to Mixed Linear Models." *American Statistician* 45:54–64.

Milliken, G. A., and Johnson, D. E. (1992). *Designed Experiments.* Vol. 1 of Analysis of Messy Data. Reprint edition. New York: Chapman & Hall.

Nelson, P. R. (1982). "Exact Critical Points for the Analysis of Means." *Communications in Statistics—Theory and Methods* 11:699–709.

Nelson, P. R. (1991). "Numerical Evaluation of Multivariate Normal Integrals with Correlations $\rho_{lj} = -\alpha_l \alpha_j$." In *Frontiers of Statistical Scientific Theory and Industrial Applications: Proceedings of the ICOSCO I Conference*, edited by A. Öztürk and E. C. van der Meulen, 97–114. Columbus, OH: American Sciences Press.

Nelson, P. R. (1993). "Additional Uses for the Analysis of Means and Extended Tables of Critical Values." *Technometrics* 35:61–71.

Ott, E. R. (1967). "Analysis of Means: A Graphical Procedure." *Industrial Quality Control* 24:101–109. Reprinted in *Journal of Quality Technology* 15 (1983): 10–18.

Patel, H. I. (1991). "Analysis of Incomplete Data from a Clinical Trial with Repeated Measurements." *Biometrika* 78:609–619.

Patterson, H. D., and Thompson, R. (1971). "Recovery of Inter-block Information When Block Sizes Are Unequal." *Biometrika* 58:545–554.

Rao, C. R. (1972). "Estimation of Variance and Covariance Components in Linear Models." *Journal of the American Statistical Association* 67:112–115.

Robinson, G. K. (1991). "That BLUP Is a Good Thing: The Estimation of Random Effects." *Statistical Science* 6:15–51.

Rubin, D. B. (1976). "Inference and Missing Data." *Biometrika* 63:581–592.

Schabenberger, O., Gregoire, T. G., and Kong, F. (2000). "Collections of Simple Effects and Their Relationship to Main Effects and Interactions in Factorials." *American Statistician* 54:210–214.

Schwarz, G. (1978). "Estimating the Dimension of a Model." *Annals of Statistics* 6:461–464.

Searle, S. R. (1971). *Linear Models*. New York: John Wiley & Sons.

Searle, S. R. (1982). *Matrix Algebra Useful for Statisticians*. New York: John Wiley & Sons.

Searle, S. R., Casella, G., and McCulloch, C. E. (1992). *Variance Components*. New York: John Wiley & Sons.

Serfling, R. J. (1980). *Approximation Theorems of Mathematical Statistics*. New York: John Wiley & Sons.

Shaffer, J. P. (1986). "Modified Sequentially Rejective Multiple Test Procedures." *Journal of the American Statistical Association* 81:826–831.

Snedecor, G. W., and Cochran, W. G. (1980). *Statistical Methods*. 7th ed. Ames: Iowa State University Press.

Steel, R. G. D., Torrie, J. H., and Dickey, D. A. (1997). *Principles and Procedures of Statistics: A Biometrical Approach*. 3rd ed. New York: McGraw-Hill.

Swallow, W. H., and Monahan, J. F. (1984). "Monte Carlo Comparison of ANOVA, MIVQUE, REML, and ML Estimators of Variance Components." *Technometrics* 26:47–57.

Verbeke, G., and Molenberghs, G., eds. (1997). *Linear Mixed Models in Practice: A SAS-Oriented Approach*. New York: Springer.

Verbeke, G., and Molenberghs, G. (2000). *Linear Mixed Models for Longitudinal Data*. New York: Springer.

Westfall, P. H. (1997). "Multiple Testing of General Contrasts Using Logical Constraints and Correlations." *Journal of the American Statistical Association* 92:299–306.

Westfall, P. H., Tobias, R. D., Rom, D., Wolfinger, R. D., and Hochberg, Y. (1999). *Multiple Comparisons and Multiple Tests Using the SAS System*. Cary, NC: SAS Institute Inc.

Westfall, P. H., and Young, S. S. (1993). *Resampling-Based Multiple Testing: Examples and Methods for* p*-Value Adjustment*. New York: John Wiley & Sons.

Winer, B. J. (1971). *Statistical Principles in Experimental Design*. 2nd ed. New York: McGraw-Hill.

Wolfinger, R. D. (1996). "Heterogeneous Variance-Covariance Structures for Repeated Measures." *Journal of Agricultural, Biological, and Environmental Statistics* 1:205–230.

Wolfinger, R. D., Gibson, G., Wolfinger, E., Bennett, L., Hamadeh, H., Bushel, P., Afshari, C., and Paules, R. S. (2001). "Assessing Gene Significance from cDNA Microarray Expression Data via Mixed Models." *Journal of Computational Biology* 8:625–637.

Wolfinger, R. D., Tobias, R. D., and Sall, J. (1994). "Computing Gaussian Likelihoods and Their Derivatives for General Linear Mixed Models." *SIAM Journal on Scientific Computing* 15:1294–1310.

Chapter 18
LOGSELECT Procedure

Contents

Overview: LOGSELECT Procedure **960**
PROC LOGSELECT Features 961
PROC LOGSELECT Compared with Other SAS Procedures 962
Using CAS Sessions and CAS Engine Librefs 963
Getting Started: LOGSELECT Procedure **964**
Binary Logistic Regression 964
Syntax: LOGSELECT Procedure **970**
PROC LOGSELECT Statement 970
BY Statement 980
CLASS Statement 980
CODE Statement 980
DISPLAY Statement 981
DISPLAYOUT Statement 982
EFFECT Statement 983
FREQ Statement 984
LSMEANS Statement 985
MODEL Statement 986
ODDSRATIO Statement 991
OUTPUT Statement 993
PARTITION Statement 999
REPEATED Statement 1000
RISK Statement 1004
SELECTION Statement 1006
STORE Statement 1008
WEIGHT Statement 1009
Details: LOGSELECT Procedure **1009**
Missing Values 1009
Response Distributions 1010
Log-Likelihood Functions 1011
Existence of Maximum Likelihood Estimates 1013
The LASSO Method of Model Selection 1014
Generalized Estimating Equations 1015
Partition Fit Statistics 1020
Model Fit and Assessment Statistics 1020
Confidence Intervals for Parameters 1024
Predicted Probabilities and Regression Diagnostics 1026

Joint Tests and Type 3 Tests . . . 1031
The Hosmer-Lemeshow Goodness-of-Fit Test . . . 1032
Odds Ratio Estimation . . . 1033
Risk Differences and Relative Risks . . . 1034
Multithreading . . . 1035
Optimization Algorithms . . . 1035
Memory Usage . . . 1036
Displayed Output . . . 1036
ODS Table Names . . . 1044
ODS Graphics . . . 1046
Examples: LOGSELECT Procedure . . . **1048**
Example 18.1: Model Selection . . . 1048
Example 18.2: Modeling Binomial Data . . . 1055
Example 18.3: Partitioning Data . . . 1062
Example 18.4: Ordinal Logistic Regression . . . 1063
Example 18.5: Odds Ratios . . . 1068
Example 18.6: Binary Model with Repeated Measures . . . 1073
Example 18.7: Restoring a Stored Model . . . 1076
Example 18.8: Modeling Microarray Data with Elastic Net . . . 1083
Example 18.9: Risks . . . 1085
References . . . **1086**

Overview: LOGSELECT Procedure

The LOGSELECT procedure fits and performs model selection for logistic regression models in SAS Viya, including binary, binomial, and multinomial response models.

The models that PROC LOGSELECT supports can contain main effects that consist of both continuous and classification variables and interaction effects of these variables. The models can also include constructed effects such as splines. The procedure offers a number of effect-selection methods, including stepwise methods and modern LASSO methods. It also offers extensive capabilities for customizing the model selection by using a wide variety of selection and stopping criteria, from computationally efficient significance-level-based criteria to modern, computationally intensive validation-based criteria. PROC LOGSELECT also provides a variety of logistic regression diagnostics that are conditional on the selected model.

Logistic regression analysis is often used to investigate the relationship between discrete responses and a set of explanatory variables. The LOGSELECT procedure fits logistic regression models in the broader sense; the procedure permits several link functions and can handle ordinal and nominal response data that have more than two response categories (multinomial data).

The logistic model shares a common feature with a more general class of linear models: a function $g = g(\mu)$ of the mean of the response variable is assumed to be linearly related to the explanatory variables. Because the mean μ implicitly depends on the stochastic behavior of the response and the explanatory variables are assumed to be fixed, the function g provides the link between the random (stochastic) component and the

systematic (deterministic) component of the response variable Y. For this reason, Nelder and Wedderburn (1972) refer to $g(\mu)$ as a link function. The LOGSELECT procedure fits a broad class of binary response models of the form

$$g(\pi \mid \mathbf{x}) = \alpha + \boldsymbol{\beta}'\mathbf{x}$$

where the link functions that are available in PROC LOGSELECT and that are widely used in practice are the logit, probit, log-log, and complementary log-log functions, and the predicted probability of an event, π, is the mean μ of the response variable. One advantage of the logit function over other link functions is that differences on the logistic scale are interpretable regardless of whether the data are sampled prospectively or retrospectively (McCullagh and Nelder 1989, Chapter 4).

If your response Y has more than two values and they can be ordered (for example, $Y \in \{1, \ldots, J-1, J\}$), then the LOGSELECT procedure uses the preceding link functions and fits ordinal response models of the form

$$g(\pi_j \mid \mathbf{x}) = \alpha_j + \boldsymbol{\beta}'\mathbf{x}, \quad j = 1, \ldots, J-1$$

where $\pi_j = \Pr(Y \leq j)$ are cumulative probabilities of the ordered response categories.

If your response Y has more than two values and they have no natural ordering, then the LOGSELECT procedure fits a generalized or baseline-category logit model, which has the form

$$\log\left(\frac{\Pr(Y = j \mid \mathbf{x})}{\Pr(Y = J \mid \mathbf{x})}\right) = \alpha_j + \boldsymbol{\beta}_j'\mathbf{x}, \quad j = 1, \ldots, J-1$$

where the $\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_{J-1}$ are J–1 vectors of slope parameters. These models are a special case of the discrete choice or conditional logit models introduced by McFadden (1974).

The LOGSELECT procedure can also fit models to correlated responses by using the generalized estimating equations (GEE) method. PROC LOGSELECT supports many of the correlation structures from Liang and Zeger (1986).

Texts that discuss logistic regression include Agresti (2013); Allison (2012); Collett (2003); Cox and Snell (1989); Hosmer and Lemeshow (2013); Stokes, Davis, and Koch (2012).

PROC LOGSELECT Features

The LOGSELECT procedure estimates the parameters of a logistic regression model by using maximum likelihood techniques. It also does the following:

- provides model-building syntax with the CLASS, EFFECT, and effect-based MODEL statements, which are familiar from SAS/STAT analytic procedures (in particular, the GLM, LOGISTIC, GLIMMIX, and MIXED procedures)
- provides response-variable options as in the LOGISTIC procedure
- performs maximum likelihood estimation
- provides the logit, probit, log-log, and complementary log-log link functions
- provides cumulative link models for ordinal response data and generalized logit modeling for unordered multinomial data

- supports repeated measures analysis by using generalized estimating equations (GEEs) for binomial and binary response functions
- enables model building (variable selection) through the SELECTION statement
- provides a WEIGHT statement for weighted analysis
- provides a FREQ statement for grouped analysis
- provides a CODE statement to produce SAS code that can score a new data set
- provides an OUTPUT statement to produce a data table that contains predicted probabilities and other observationwise statistics
- uses ODS Graphics to create model selection plots as part of its output. For more information about ODS Graphics, see the section "ODS Graphics" on page 1046.

Because the LOGSELECT procedure runs on CAS, it also does the following:

- enables you to run on a cluster of machines that distribute the data and the computations
- enables you to run in single-machine mode on CAS
- exploits all the available cores and concurrent threads. For information about how PROC LOGSELECT uses threads, see the section "Multithreading" on page 102 in Chapter 2, "Shared Concepts."

PROC LOGSELECT Compared with Other SAS Procedures

The LOGSELECT procedure provides logistic regression modeling functionality that is comparable to that of the HPLOGISTIC and LOGISTIC procedures in SAS/STAT software.

PROC LOGSELECT Compared with the HPLOGISTIC Procedure

The functionality of the LOGSELECT procedure closely resembles that of the HPLOGISTIC procedure, which is a high-performance procedure. The LOGSELECT procedure is the next generation of the HPLOGISTIC procedure, and it was developed specifically for SAS Viya. Both procedures are designed to run on a cluster of machines that distribute the data and the computations. Both procedures perform computations in multiple threads.

The LOGSELECT procedure provides all the model-fitting methods and postprocessing results that the HPLOGISTIC procedure provides. In addition, PROC LOGSELECT provides the full set of parameterizations available in PROC LOGISTIC and other modeling procedures; the ability to create complex constructed effects, including univariate spline and polynomial expansions; LASSO selection; elastic net selection; selection plots by using ODS Graphics; more regression diagnostics than PROC HPLOGISTIC produces; and odds ratios.

PROC LOGSELECT Compared with the LOGISTIC Procedure

The LOGSELECT procedure provides the full set of parameterizations that are available in PROC LOGISTIC. The LOGSELECT procedure uses the GLM parameterization for the CLASS variables by default. The LOGISTIC procedure uses the EFFECT parameterization for the CLASS variables by default. In either procedure, you can use the PARAM= option in the CLASS statement to change the parameterization.

The LOGISTIC procedure uses Fisher scoring by default. The LOGSELECT procedure uses a modification of the Newton-Raphson algorithm with a ridged Hessian by default. You can choose different optimization

techniques, including first-order methods that do not require a crossproducts matrix or Hessian, by using the TECHNIQUE= option in the PROC LOGSELECT statement.

The LOGSELECT procedure provides a richer set of model selection options than the LOGISTIC procedure, including LASSO selection, elastic net selection, information-criterion-based selection and stopping criteria, and validation-based criteria.

The LOGSELECT procedure uses ODS Graphics to display plots that help interpret the selection process; the LOGISTIC procedure displays many diagnostic and fit plots.

The LOGISTIC procedure offers a wide variety of postfitting analyses, such as contrasts, estimates, tests of model effects, and least squares means. The LOGSELECT procedure is limited in postfitting functionality, because for large data sets the focus is primarily on model fitting and scoring.

The LOGISTIC procedure provides an AGGREGATE option to compress your data before computing chi-square goodness-of-fit diagnostic statistics. To compute the same diagnostics with the LOGSELECT procedure, you have to aggregate your data appropriately before invoking the procedure, and you have to specify your model by using the events/trials or multinomial-trial syntax.

The LOGSELECT procedure is designed to operate in SAS Viya and performs computations in multiple threads. The LOGISTIC procedure executes in a single thread on a single machine.

Using CAS Sessions and CAS Engine Librefs

SAS Cloud Analytic Services (CAS) is the analytic server and associated cloud services in SAS Viya. This section describes how to create a CAS session and set up a CAS engine libref that you can use to connect to the CAS session. It assumes that you have a CAS server already available; contact your system administrator if you need help starting and terminating a server. This CAS server is identified by specifying the host on which it runs and the port on which it listens for communications. To simplify your interactions with this CAS server, the host information and port information for the server are stored as SAS option values that are retrieved automatically whenever this CAS server needs to be accessed. You can examine the host and port values for the server at your site by using the following statements:

```
proc options option=(CASHOST CASPORT);
run;
```

In addition to starting a CAS server, your system administrator might also have created a CAS session and a CAS engine libref for your use. You can define your own sessions and CAS engine librefs that connect to the CAS server as shown in the following statements:

```
cas mysess;
libname mylib cas sessref=mysess;
```

The CAS statement creates the CAS session named mysess, and the LIBNAME statement creates the mylib CAS engine libref that you use to connect to this session. It is not necessary to explicitly name the CASHOST and CASPORT of the CAS server in the CAS statement, because these values are retrieved from the corresponding SAS option values.

If you have created the mysess session, you can terminate it by using the TERMINATE option in the CAS statement as follows:

```
cas mysess terminate;
```

For more information about the CAS and LIBNAME statements, see the section "Introduction to Shared Concepts" on page 6 in Chapter 2, "Shared Concepts."

Getting Started: LOGSELECT Procedure

Binary Logistic Regression

(View the complete code for this example at https://github.com/sascommunities/doc-supplement-casstat/tree/main/Examples/loggs1.sas.)

NOTE: Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections "Using CAS Sessions and CAS Engine Librefs" on page 7 and "Loading a SAS Data Set onto a CAS Server" on page 7 in Chapter 2, "Shared Concepts."

The following DATA step creates the data table getStarted, which consists of 100 observations on a dichotomous response variable (y), a character variable (C), and 10 continuous variables (x1–x10):

```
data mylib.getStarted;
   input C$ y x1-x10;
   datalines;
   D  0  10.2  6  1.6  38  15  2.4  20  0.8  8.5  3.9
   F  1  12.2  6  2.6  42  61  1.5  10  0.6  8.5  0.7
   D  1   7.7  1  2.1  38  61    1  90  0.6  7.5  5.2
   J  1  10.9  7  3.5  46  42  0.3   0  0.2    6  3.6
   E  0  17.3  6  3.8  26  47  0.9  10  0.4  1.5  4.7
   A  0  18.7  4  1.8   2  34  1.7  80    1  9.5  2.2
   B  0   7.2  1  0.3  48  61  1.1  10  0.8  3.5    4
   D  0   0.1  3  2.4   0  65  1.6  70  0.8  3.5  0.7
   H  1   2.4  4  0.7  38  22  0.2  20    0    3  4.2
   J  0  15.6  7  1.4   0  98  0.3   0    1    5  5.2
   J  0  11.1  3  2.4  42  55  2.2  60  0.6  4.5  0.7
   F  0     4  6  0.9   4  36  2.1  30  0.8    9  4.6
   A  0   6.2  2  1.8  14  79  1.1  70  0.2    0  5.1
   H  0   3.7  3  0.8  12  66  1.3  40  0.4  0.5  3.3
   A  1   9.2  3  2.3  48  51  2.3  50    0    6  5.4
   G  0    14  3    2  18  12  2.2   0    0    3  3.4
   E  1  19.5  6  3.7  26  81  0.1  30  0.6    5  4.8
   C  0    11  3  2.8  38   9  1.7  50  0.8  6.5  0.9
   I  0  15.3  7  2.2  20  98  2.7 100  0.4    7  0.8
   H  1   7.4  4  0.5  28  65  1.3  60  0.2  9.5  5.4
   F  0  11.4  2  1.4  42  12  2.4  10  0.4    1  4.5
   C  1  19.4  1  0.4  42   4  2.4  10    0  6.5  0.1
   G  0   5.9  4  2.6  12  57  0.8  50  0.4    2  5.8
   G  1  15.8  6  3.7  34   8  1.3  90  0.6  2.5  5.7
   I  0    10  3  1.9  16  80    3  90  0.4  9.5  1.9
   E  0  15.7  1  2.7  32  25  1.7  20  0.2  8.5    6
```

```
G  0    11  5  2.9  48  53  0.1  50    1  3.5  1.2
J  1  16.8  0  0.9  14  86  1.4  40  0.8    9    5
D  1    11  4  3.2  48  63  2.8  90  0.6    0  2.2
J  1   4.8  7  3.6  24   1  2.2  20    1  8.5  0.5
J  1  10.4  5    2  42  56    1  20    0  3.5  4.2
G  0  12.7  7  3.6   8  56  2.1  70    1  4.5  1.5
G  0   6.8  1  3.2  30  27  0.6   0  0.8    2  5.6
E  0   8.8  0  3.2   2  67  0.7  10  0.4    1    5
I  1   0.2  0  2.9  10  41  2.3  60  0.2    9  0.3
J  1   4.6  7  3.9  50  61  2.1  50  0.4    3  4.9
J  1   2.3  2  3.2  36  98  0.1  40  0.6  4.5  4.3
I  0  10.8  3  2.7  28  58  0.8  80  0.8    3    6
B  0   9.3  2  3.3  44  44  0.3  50  0.8  5.5  0.4
F  0   9.2  6  0.6   4  64  0.1   0  0.6  4.5  3.9
D  0   7.4  0  2.9  14   0  0.2  30  0.8  7.5  4.5
G  0  18.3  3  3.1   8  60  0.3  60  0.2    7  1.9
F  0   5.3  4  0.2  48  63  2.3  80  0.2    8  5.2
C  0   2.6  5  2.2  24   4  1.3  20    0    2  1.4
F  0  13.8  4  3.6   4   7  1.1  10  0.4  3.5  1.9
B  1  12.4  6  1.7  30  44  1.1  60  0.2    6  1.5
I  0   1.3  1  1.3   8  53  1.1  70  0.6    7  0.8
F  0  18.2  7  1.7  26  92  2.2  30    1  8.5  4.8
J  0   5.2  2  2.2  18  12  1.4  90  0.8    4  4.9
G  1   9.4  2  0.8  22  86  0.4  30  0.4    1  5.9
J  1  10.4  2  1.7  26  31  2.4  10  0.2    7  1.6
J  0    13  1  1.8  14  11  2.3  50  0.6  5.5  2.6
A  0  17.9  4  3.1  46  58  2.6  90  0.6  1.5  3.2
D  1  19.4  6    3  20  50  2.8 100  0.2    9  1.2
I  0  19.6  3  3.6  22  19  1.2   0  0.6    5  4.1
I  1     6  2  1.5  30  30  2.2  20  0.4  8.5  5.3
G  0  13.8  1  2.7   0  52  2.4  20  0.8    6    2
B  0  14.3  4  2.9  30  11  0.6  90  0.6  0.5  4.9
E  0  15.6  0  0.4  38  79  0.4  80  0.4    1  3.3
D  0    14  2    1  22  61    3  90  0.6    2  0.1
C  1   9.4  5  0.4  12  53  1.7  40    0    3  1.1
H  0  13.2  1  1.6  40  15  0.7  40  0.2    9  5.5
A  0  13.5  5  2.4  18  89  1.6  20  0.4  9.5  4.7
E  0   2.6  4  2.3  38   6  0.8  20  0.4    5  5.3
E  0  12.4  3  1.3  26   8  2.8  10  0.8    6  5.8
D  0   7.6  2  0.9  44  89  1.3  50  0.8    6  0.4
I  0  12.7  1  2.3  42   6  2.4  10  0.4    1    3
C  1  10.7  4  3.2  28  23  2.2  90  0.8  5.5  2.8
H  0  10.1  2  2.3  10  62  0.9  50  0.4  2.5  3.7
C  1  16.6  1  0.5  12  88  0.1  20  0.6  5.5  1.8
I  1   0.2  3  2.2   8  71  1.7  80  0.4  0.5  5.5
C  0  10.8  4  3.5  30  70  2.3  60  0.4  4.5  5.9
F  0   7.1  4    3  14  63  2.4  70    0    7  3.1
D  0  16.5  1  3.3  30  80  1.6  40    0  3.5  2.7
H  0  17.1  7  2.1  30  45  1.5  60  0.6  0.5  2.8
D  0   4.3  1  1.5  24  44    0  70    0    5  0.5
H  0    15  2  0.2  14  87  1.8  50    0  4.5  4.7
G  0  19.7  3  1.9  36  99  1.5  10  0.6    3  1.7
H  1   2.8  6  0.6  34  21    2  60    1    9  4.7
G  0  16.6  3  3.3  46   1  1.4  70  0.6  1.5  5.3
```

```
   E  0  11.7  5  2.7  48   4  0.9  60  0.8  4.5  1.6
   F  0  15.6  3  0.2   4  79  0.5   0  0.8  1.5  2.9
   C  1   5.3  6  1.4   8  64    2  80  0.4    9  4.2
   B  1   8.1  7  1.7  40  36  1.4  60  0.6    6  3.9
   I  0  14.8  2  3.2   8  37  0.4  10    0  4.5    3
   D  0   7.4  4    3  12   3  0.6  60  0.6    7  0.7
   D  0   4.8  3  2.3  44  41  1.9  60  0.2    3  3.1
   A  0   4.5  0  0.2   4  48  1.7  80  0.8    9  4.2
   D  0   6.9  6  3.3  14  92  0.5  40  0.4  7.5    5
   B  0   4.7  4  0.9  14  99  2.4  80    1  0.5  0.7
   I  1   7.5  4  2.1  20  79  0.4  40  0.4  2.5  0.7
   C  0   6.1  0  1.4  38  18  2.3  60  0.8  4.5  0.7
   C  0  18.3  1    1  26  98  2.7  20    1  8.5  0.5
   F  0  16.4  7  1.2  32  94  2.9  40  0.4  5.5  2.1
   I  0   9.4  2  2.3  32  42  0.2  70  0.4  8.5  0.3
   F  1  17.9  4  1.3  32  42    2  40  0.2    1  5.4
   H  0  14.9  3  1.6  36  74  2.6  60  0.2    1  2.3
   C  0  12.7  0  2.6   0  88  1.1  80  0.8  0.5  2.1
   F  0   5.4  4  1.5   2   1  1.8  70  0.4  5.5  3.6
   J  1  12.1  4  1.8  20  59  1.3  60  0.4    3  3.8
;
```

These statements assume that the CAS engine libref is named mylib, but you can substitute any appropriately named CAS engine libref.

The following statements fit a logistic model to these data by using a classification effect for variable C and 10 regressor effects for x1–x10. The ITHIST option displays a table that summarizes the steps of the optimization.

```
proc logselect data=mylib.getStarted ithist;
   class C;
   model y = C x1-x10;
run;
```

The output from this analysis is presented in Figure 18.1 through Figure 18.10.

Figure 18.1 displays the "Model Information" table. The LOGSELECT procedure uses a Newton-Raphson algorithm to model a binary distribution with a logit link function for the variable y. The CLASS variable C is parameterized using the GLM parameterization, which is the default.

Figure 18.1 Model Information

The LOGSELECT Procedure

Model Information	
Data Source	GETSTARTED
Response Variable	y
Distribution	Binary
Link Function	Logit
Optimization Technique	Newton-Raphson with Ridging

Figure 18.2 displays the "Number of Observations" table. All 100 observations in the data table are used in the analysis.

Figure 18.2 Number of Observations

Number of Observations Read	100
Number of Observations Used	100

The "Response Profile" table in Figure 18.3 is produced by default. It shows the breakdown of the response variable levels by frequency. By default for binary data, the LOGSELECT procedure models the probability of the event with the lower OrderedValue value in the "Response Profile" table, as indicated by the note that follows the table. In this example, the values that are represented by y = 0 are modeled as the "successes" in the Bernoulli experiments.

Figure 18.3 Response Profile

Response Profile

Ordered Value	y	Total Frequency
1	0	69
2	1	31

Probability modeled is y = 0.

You can use the response-variable options in the MODEL statement to choose which value of the response variable to model.

The CLASS variable C has 10 unique formatted levels that are displayed in the "Class Level Information" table in Figure 18.4.

Figure 18.4 Class Level Information

Class Level Information

Class	Levels	Values
C	10	A B C D E F G H I J

The "Iteration History" table is shown in Figure 18.5. The Newton-Raphson algorithm with ridging converged after four iterations, not counting the initial setup iteration.

Figure 18.5 Iteration History

Iteration History

Iteration	Evaluations	Objective Function	Change	Maximum Gradient
0	4	0.4493546916		0.410972
1	2	0.4436453992	0.00570929	0.081339
2	2	0.4435038109	0.00014159	0.003302
3	2	0.4435035933	0.00000022	5.623E-6
4	2	0.4435035933	0.00000000	1.59E-11

Figure 18.6 displays the final convergence status of the Newton-Raphson algorithm. The GCONV= relative convergence criterion is satisfied.

Figure 18.6 Convergence Status

Convergence criterion (GCONV=1E-8) satisfied.

Figure 18.7 displays the "Dimensions" table for this model. This table summarizes some important sizes of various model components. For example, it shows that the design matrix **X** has 21 columns, which correspond to 1 column for the intercept, 10 columns for the effect associated with the classification variable C, and 1 column each for the continuous variables x1–x10. However, the rank of the crossproducts matrix is only 20. Because the classification variable C uses GLM parameterization and because the model contains an intercept, there is one singularity in the crossproducts matrix of the model. Consequently, only 20 parameters enter the optimization.

Figure 18.7 Dimensions in Binary Logistic Regression

Dimensions	
Columns in Design	21
Number of Effects	12
Max Effect Columns	10
Rank of Design	20
Parameters in Optimization	20

Figure 18.8 shows the global test for the null hypothesis that all model effects jointly do not affect the probability of success of the binary response. The test is significant ($p = 0.0135$). One or more of the model effects thus significantly affect the probability of observing an event.

Figure 18.8 Null Test

Testing Global Null Hypothesis: BETA=0			
Test	**DF**	**Chi-Square**	**Pr > ChiSq**
Likelihood Ratio	19	35.1194	0.0135

The "Fit Statistics" table is shown in Figure 18.9. The –2 log likelihood at the converged estimates is 88.7007. You can use this value to compare the model to nested model alternatives by means of a likelihood ratio test. To compare models that are not nested, you can use information criteria such as AIC (Akaike's information criterion), AICC (Akaike's bias-corrected information criterion), and SBC (Schwarz Bayesian information criterion). These criteria penalize the –2 log likelihood for the number of parameters. Because of the large number of parameters relative to the number of observations, the discrepancy between the –2 log likelihood and, for example, AIC is substantial in this case.

Figure 18.9 Fit Statistics

Fit Statistics	
-2 Log Likelihood	88.70072
AIC (smaller is better)	128.70072
AICC (smaller is better)	139.33363
SBC (smaller is better)	180.80412

However, the "Parameter Estimates" table in Figure 18.10 shows that many parameters have fairly large p-values, indicating that one or more of the model effects might not be necessary.

Figure 18.10 Parameter Estimates

Parameter Estimates

Parameter	DF	Estimate	Standard Error	Chi-Square	Pr > ChiSq
Intercept	1	1.210139	1.750750	0.4778	0.4894
C A	1	3.434071	1.613142	4.5318	0.0333
C B	1	2.163830	1.427083	2.2990	0.1295
C C	1	0.655205	1.081024	0.3674	0.5445
C D	1	2.494481	1.109366	5.0560	0.0245
C E	1	3.244860	1.432140	5.1336	0.0235
C F	1	3.605371	1.307025	7.6091	0.0058
C G	1	2.084085	1.189764	3.0684	0.0798
C H	1	2.936779	1.293876	5.1518	0.0232
C I	1	1.378522	1.031899	1.7847	0.1816
C J	0	0	.	.	.
x1	1	0.032181	0.057100	0.3176	0.5730
x2	1	-0.367741	0.153824	5.7152	0.0168
x3	1	0.314640	0.357414	0.7750	0.3787
x4	1	-0.051957	0.024428	4.5239	0.0334
x5	1	-0.006828	0.010555	0.4184	0.5177
x6	1	0.253852	0.378462	0.4499	0.5024
x7	1	-0.007233	0.010734	0.4541	0.5004
x8	1	2.537042	0.994180	6.5122	0.0107
x9	1	-0.167502	0.106789	2.4603	0.1168
x10	1	-0.222155	0.157743	1.9834	0.1590

Finally, the procedure displays the table in Figure 18.11, which shows the amount of time (in seconds) that PROC LOGSELECT required to perform different tasks in the analysis.

Figure 18.11 Procedure Timing

Task Timing

Task	Seconds	Percent
Setup and Parsing	0.07	24.60%
Levelization	0.06	20.95%
Model Initialization	0.01	2.04%
SSCP Computation	0.01	3.94%
Model Fitting	0.13	47.57%
Display	0.00	0.57%
Cleanup	0.00	0.00%
Total	0.27	100.00%

Syntax: LOGSELECT Procedure

The following statements are available in the LOGSELECT procedure:

PROC LOGSELECT < *options* > **;**
BY *variables* **;**
CLASS *variable* < **(***options***)** >. . . < *variable* < **(***options***)** > > < / *global-options* > **;**
CODE < *options* > **;**
DISPLAY < *table-list* > < / *options* > **;**
DISPLAYOUT *table-spec-list* < / *options* > **;**
EFFECT *name***=***effect-type***(***variables* < / *options* >**)** **;**
FREQ *variable* **;**
MODEL *response*< **(***response-options***)** > **=** < *effects* > < / *model-options* > **;**
MODEL *events/trials* **=** < *effects* > < / *model-options* > **;**
MODEL *response-level1 response-level2*< . . . *response-levelJ* > **=** < *effects* > < / *model-options* > **;**
LSMEANS *model-effects* < / *options* > **;**
ODDSRATIO < *variable* < **(***options***)** > >. . . < *variable* < **(***options***)** > > < / *global-options* > **;**
OUTPUT OUT=*libref.data-table* < *options* > < *keyword* < **=***name* > >. . . < *keyword* < **=***name* > > **;**
PARTITION *partition-options* **;**
REPEATED SUBJECT=*subject-effect* < / *repeated-options* > **;**
RISK < *variable-list* > < / *options* > **;**
SELECTION < **METHOD=***method*< **(***method-options***)** > >< *options* > **;**
STORE < **OUT=** >*libref.data-table* **;**
WEIGHT *variable* < / **NORMALIZE** > **;**

The CLASS statement can appear multiple times. If a CLASS statement is specified, it must precede the MODEL statement. If an LSMEANS statement is specified, it must follow the MODEL statement.

PROC LOGSELECT Statement

PROC LOGSELECT < *options* > **;**

The PROC LOGSELECT statement invokes the procedure. Table 18.1 summarizes the available *options* in the PROC LOGSELECT statement by function.

Table 18.1 PROC LOGSELECT Statement Options

Option	Description
ALPHA=	Specifies a global significance level
APPLYROWORDER	Uses group and order information from the input data table
BINEPS=	Specifies the precision for computing association and classification statistics
DATA=	Specifies the input data table
FITDATA	Declares the DATA= table to be the same input data table used to build the model
MAXRESPONSELEVELS=	Specifies the maximum number of response levels allowed when you have a polytomous response variable

Table 18.1 *continued*

Option	Description
MULTIPASS=	Specifies whether to levelize the data table every time it is read
PAGEOBS=	Specifies the maximum number of observations to be computed in each batch
RESTORE=	Specifies the input analytic store
SEED=	Specifies the random number seed for LS-means and partition fractions
Output Options	
ASSOCIATION	Produces association statistics
CLB	Constructs confidence limits for the parameter estimates
CORRB	Displays the "Parameter Estimates Correlation Matrix" table
COVB	Displays the "Parameter Estimates Covariance Matrix" table
CTABLE	Produces classification (ROC) statistics
ITHIST	Displays the "Iteration History" table
LACKFIT	Produces the Hosmer-Lemeshow goodness-of-fit test
NOCHECK	Disables checking for infinite parameters
NOCLPRINT	Limits or suppresses the display of class levels
NOSTDERR	Suppresses computation of the covariance matrix and standard errors
NOXPX	Suppresses computation of the Hessian matrices
PARTFIT	Displays the fit statistics that are produced when your data are partitioned
PLCONV=	Controls the convergence criterion for profile-likelihood confidence limits
PLMAXITER=	Specifies the maximum number of iterations for profile-likelihood confidence limits
PLSINGULAR=	Specifies the singularity tolerance for profile-likelihood confidence limits
STB	Displays standardized estimates
TYPE3	Produces the Type 3 or joint tests of effects
USELASTITER	Continues computations when the optimization fails
Optimization Options	
ABSCONV=	Tunes the absolute function convergence criterion
ABSFCONV=	Tunes the absolute function difference convergence criterion
ABSGCONV=	Tunes the absolute gradient convergence criterion
ABSXCONV=	Tunes the absolute parameter convergence criterion
FCONV=	Tunes the relative function difference convergence criterion
FCONV2=	Tunes the second relative function difference convergence criterion
GCONV=	Tunes the relative gradient convergence criterion
GCONV2=	Tunes the second relative gradient convergence criterion
INPARMEST=	Specifies starting values for the optimization
XCONV=	Tunes the relative gradient convergence criterion
MAXFUNC=	Specifies the maximum number of function evaluations in any optimization
MAXITER=	Specifies the maximum number of iterations in any optimization
MAXTIME=	Specifies the upper limit of CPU time (in seconds) for any optimization
MINITER=	Specifies the minimum number of iterations in any optimization
NORMALIZE=	Specifies whether the objective function is normalized during optimization
TECHNIQUE=	Selects the optimization technique

Table 18.1 *continued*

Option	Description
LASSO Options	
LASSORHO=	Specifies the base regularization parameter for the LASSO method
LASSOSTEPS=	Specifies the maximum number of steps for the LASSO method
LASSOTOL=	Specifies the convergence criterion for the LASSO method

The optimization options are fully described in the section "Optimization Options" on page 57 in Chapter 2, "Shared Concepts." The following list describes the other options available in the PROC LOGSELECT statement:

ALPHA=*number*

specifies a global significance level for the construction of confidence intervals. The confidence level is 1–*number*. The value of *number* must be between 0 and 1. You can override the global specification by specifying the ALPHA= option in the OUTPUT statement. By default, ALPHA=0.05.

APPLYROWORDER

uses group and order information from the input data table. You can add this information to the input table by using the **`partition`** action in the **`table`** action set. For more information, see the section "The APPLYROWORDER Option" on page 107 in Chapter 2, "Shared Concepts." By default, the APPLYROWORDER option is not enabled. If you specify this option but the table does not contain group and order information, the procedure terminates with an error.

NOTE: You can use group and order information for a repeated measures analysis if the group information meets certain criteria. For more information, see the section "Using a Preexisting Table Order" on page 1019.

ASSOCIATION

displays measures of association between predicted probabilities and observed responses for binary, binomial, and ordinal response models. These measures assess the predictive ability of the model. The displayed statistics are the concordance index *c* (the area under the ROC curve, AUC), Somers' *D* statistic (Gini's coefficient), the Goodman-Kruskal gamma statistic, and Kendall's tau-*a* statistic. These statistics are based on the number of pairs of observations that have different response values, the number of concordant pairs, and the number of discordant pairs. For more information, see the section "Association Statistics" on page 1024.

BINEPS=*number*

specifies the precision to use for the association, classification, and LACKFIT option computations. The predicted probabilities or scores are rounded to the nearest multiple of *number*, which effectively bins and sorts the observations. You can specify a *number* between 0 and 1; if *number* does not evenly divide the unit interval, then it is reduced to a valid value. By default, BINEPS=1E–5. For more information, see the section "Classification Table and ROC Curves" on page 1022.

CLB< =WALD | PL >
displays Wald confidence limits for each of the parameter estimates. If you specify CLB=PL, then profile-likelihood confidence limits are also displayed. The confidence level is 0.95 by default; you can change it by specifying the ALPHA= option. This option is not available when you use either LASSO selection or elastic net selection.

CORRB
creates the "Parameter Estimates Correlation Matrix" table. The correlation matrix is computed by normalizing the covariance matrix $\mathbf{\Sigma}$. That is, if σ_{ij} is an element of $\mathbf{\Sigma}$, then the corresponding element of the correlation matrix is $\sigma_{ij}/\sigma_i\sigma_j$, where $\sigma_i = \sqrt{\sigma_{ii}}$. The option is not available when you use either LASSO selection or elastic net selection.

COVB
creates the "Parameter Estimates Covariance Matrix" table. The covariance matrix is computed as the inverse of the negative of the matrix of second derivatives of the log-likelihood function with respect to the model parameters (the Hessian matrix). The option is not available when you use either LASSO selection or elastic net selection.

CTABLE< (*options*) >
displays a table for binary or binomial response models that, for a set of probabilities or cutpoints, contains the frequencies of observations that are correctly and incorrectly classified as events and nonevents, and optionally the sensitivity, the 1–specificity, the positive and negative predictive values, the correct classification rate, the error rate, and the lift.

Because the number of cutpoints that are generated from your data can be very large, you can write the classification table to an output data table, or you can specify your own list of cutpoints.

If you specify the PRIOR= option, then the reported predicted probabilities are unchanged, but the statistics that are specified by the keywords PPV, NPV, PC, ACCURACY, MISCLASS, and LIFT are modified.

If you specify a PARTITION statement, then a table is created for each role. If you do not specify the PRIOR= option, then the training proportions are used as the prevalences for computing the statistics that are specified by the keywords PPV, NPV, PC, ACCURACY, MISCLASS, and LIFT for the validation and test data.

You can specify the following *options*. For more information, see the section "Classification Table and ROC Curves" on page 1022.

ALL
requests all the statistics that are listed in Table 18.2.

CUTPT=*number-list*
specifies cutpoints to include in the classification table.

NOCOUNTS
suppresses the classification table columns that contain the number of true positives, the number of true negatives, the number of false positives, and the number of false negatives.

OUT=*libref.data-table*
suppresses the display of the classification table and names the output data table in which to store the classification table. *libref.data-table* is a two-level name, where *libref* refers to the library, and *data-table* specifies the name of the output data table. For more information about this two-level name, see the DATA= option and the section "Using CAS Sessions and CAS Engine Librefs" on page 963.

keyword **<** **=***name* **>**
specifies statistics to include in the classification table and optionally names the statistics. If you do not provide a *name*, the LOGSELECT procedure uses the *keyword* as the *name*. Table 18.2 lists the available *keywords*.

Table 18.2 CTABLE Option Keywords

Keyword	Description
ACCURACY	Requests and names the accuracy
FNF	Requests and names the false negative fraction
FPF	Requests and names the false positive fraction. Also called 1–specificity.
LIFT	Requests and names the lift
MISCLASS	Requests and names the misclassification rate. Also called the error rate.
NPV	Requests and names the negative predictive value
PC	Requests and names the proportion classified correctly
PPV	Requests and names the positive predictive value. Also called the precision.
TNF	Requests and names the true negative fraction. Also called the specificity.
TPF	Requests and names the true positive fraction. Also called the sensitivity or recall.

DATA=*libref.data-table*
names the input data table for PROC LOGSELECT to use. The default is the most recently created data table. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to the data, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 963.

data-table specifies the name of the input data table.

FITDATA
declares the DATA= table to be the same input data table that is used for building the model, when you also specify the RESTORE= option. The FITDATA option enables you to specify the PARTFIT option to produce more fit statistics, to compute all the statistics from the OUTPUT statement that are available for your model, and to group computations according to the partition roles.

INPARMEST=*libref.data-table*
names an input data table that contains starting values. *libref.data-table* is a two-level name, where *libref* refers to the library, and *data-table* specifies the name of the output data table. For more information about this two-level name, see the DATA= option and the section "Using CAS Sessions and CAS Engine Librefs" on page 963.

The input data table must contain an Estimate column and a ParmName column, and it can optionally contain your BY variables. You can create a version of this table by using a DISPLAYOUT statement to produce the "Parameter Estimates" table from a PROC LOGSELECT program. For example:

```
displayout parameterestimates=pe;
```

You can then modify the parameter estimates in this table and copy them back in by using the following statement:

```
proc logselect inparmest=mylib.pe;
```

ITHIST
generates the "Iteration History" table.

LACKFIT<(*options***)>**
performs the Hosmer-Lemeshow goodness-of-fit test for binary and polytomous response models (Hosmer and Lemeshow 2000; Fagerland, Hosmer, and Bofin 2008; Fagerland and Hosmer 2013, 2016). The subjects are divided into at most G groups of approximately the same size. You can specify G as any integer greater than or equal to 5; by default, $G = 10$. The discrepancies between the observed and expected numbers of observations in these groups are summarized by the Pearson chi-square statistic, which is then compared to a chi-square distribution. The number of degrees of freedom that PROC LOGSELECT uses for this test depends on the specified model; you can override this number by specifying a nonnegative integer, df, and by modifying the reduction, r. A small p-value suggests that the fitted model is not an adequate model. For more information, see the section "The Hosmer-Lemeshow Goodness-of-Fit Test" on page 1032.

The following *options* are available:

CUTPT=*value-list*
specifies a list of cut points for grouping the subjects.

DF=*df*
specifies the number of degrees of freedom of the test, where $df \geq 1$. By default, $df = G - r$.

DFREDUCE=*r*
specifies the reduction in degrees of freedom. This option is ignored when you specify the DF= option. By default, DFREDUCE=2.

NGROUPS=*G*
specifies the number of groups in which to divide the subjects. You can specify $G \geq 5$. By default, NGROUPS=10.

NONCENTRALITY=*value*
requests that a noncentral chi-square test be performed, where *value* is the noncentrality parameter.

POWERADJ
for binary response data, chooses the number of groups G to maintain the same power as grouping 1,000 observations into 10 groups (Paul, Pennell, and Lemeshow 2013). If you have fewer than 1,000 observations, then $G = 10$ groups are used. If you have 1,000 to 25,000 observations, then the number of groups that are formed is

$$\max\left(10, \min\left(\frac{n_e}{2}, \frac{n_n}{2}, 2 + 8\left(\frac{n}{1000}\right)^2\right)\right)$$

where $n = n_e + n_n$ is the number of observations, n_e is the number of events, and n_n is the number of nonevents. If you have more than 25,000 observations, then this option is ignored.

LASSORHO=*r*
specifies the base regularization parameter for the LASSO model selection method. The regularization parameter for step i is r^i. By default, LASSORHO=0.8.

LASSOSTEPS=*n*
specifies the maximum number of steps for LASSO model selection. By default, LASSOSTEPS=20.

LASSOTOL=*r*
specifies the convergence tolerance for the optimization algorithm that solves for the LASSO parameter estimates at each step of LASSO model selection. By default, LASSOTOL=1E–6.

MAXRESPONSELEVELS=*value*
specifies the maximum number of response levels that are allowed when you have a polytomous response variable. By default, MAXRESPONSELEVELS=100.

MULTIPASS=TRUE | FALSE
specifies when to levelize the data—that is, when to generate the design rows for your model from the data. The default, MULTIPASS=FALSE, levelizes the data once and stores them in a temporary data table, then rereads the data from that table as needed; this method is fast but uses more memory. MULTIPASS=TRUE levelizes the data every time they are read; this method is slower but uses less memory. The MULTIPASS= option does not affect how the OUTPUT statement is processed. When MULTIPASS=TRUE, you can also use the PAGEOBS= option to improve performance.

NOCHECK
disables the checking process that determines whether maximum likelihood estimates of the regression parameters exist. For more information, see the section "Existence of Maximum Likelihood Estimates" on page 1013.

NOCLPRINT<=*number*>
suppresses the display of the "Class Level Information" table if you do not specify *number*. If you specify *number*, the values of the classification variables are displayed for only those variables whose number of levels is less than *number*. Specifying *number* helps to reduce the size of the "Class Level Information" table if some classification variables have a large number of levels.

NOSTDERR
suppresses computation of the covariance matrix and the standard errors of the regression coefficients. When the model contains many variables (thousands), the inversion of the Hessian matrix to derive the covariance matrix and the standard errors of the regression coefficients can be time-consuming. The CORRB, COVB, and TYPE3 options, and the OUTPUT statement options for confidence limits, are not available when the NOSTDERR option is specified. This option also disables the quasi-complete separation check; for more information, see the section "Existence of Maximum Likelihood Estimates" on page 1013.

NOXPX
suppresses Hessian and $\mathbf{X}'\mathbf{X}$ computations and invokes the NOSTDERR option. When the model contains many variables (thousands), computing the Hessian matrix during model fitting can be time-consuming. The CLB, CORRB, COVB, and TYPE3 options, the REPEATED statement, the OUTPUT statement options that rely on the covariance, and confidence limits for the parameters and for the ODDSRATIO, OUTPUT, RISK, and LSMEANS statements are not available when you specify the NOXPX option. The forward, backward, and stepwise selection methods are not available; however, you can specify the LASSO and elastic net methods. This option invokes the TECHNIQUE=LBFGS option by default, and the NEWRAP, NRRIDG, and TRUREG optimization techniques are not available. Because the method of identifying linearly dependent variables relies on the $\mathbf{X}'\mathbf{X}$ matrix, some parameters that should be assigned 0 degrees of freedom can be missed. This option also disables the quasi-complete separation check; for more information, see the section "Existence of Maximum Likelihood Estimates" on page 1013.

PAGEOBS=*number* | AUTO

MAXOPTBATCH=*number* | AUTO
specifies the maximum number of observations to be included in a batch. During the optimization, the LOGSELECT procedure reads at most *number* observations from the data table into memory, performs the appropriate log-likelihood, gradient, and Hessian computations on that batch of observations, then discards those observations and reads in the next batch of data for processing. Generally, a smaller *number* decreases memory usage but might lead to longer computation times, whereas a larger *number* might lead to shorter computation times but increases memory usage. The default PAGEOBS=AUTO option determines whether the entire data table can be conveniently held in memory; if it cannot, then *number* is set to 256. The PAGEOBS= option has the most benefit when you also specify the MULTIPASS=TRUE option; however, it can adversely affect performance when the default MULTIPASS=FALSE option is in effect. For more information, see the section "Memory Usage" on page 1036.

PARTFIT
displays fit statistics in the "Fit Statistics" table that are usually produced when your data are partitioned. This option is not required when you specify a PARTITION statement.

The additional statistics include the R-square, the adjusted R-square, McFadden's R-square, the average square error (Brier score), the misclassification rate, the null-model log likelihood, and the difference in means.

PLCONV=*value*
controls the convergence criterion for computing profile-likelihood confidence intervals. The *value* is a positive number. By default, PLCONV=1E–4. This option has no effect if you do not request profile-likelihood confidence intervals.

PLMAXITER=*number*
specifies the maximum number of iterations for profile-likelihood confidence interval searches. By default, PLMAXITER=25. If you do not reach convergence in *number* iterations, the confidence limit is set to missing. This option has no effect if you do not request profile-likelihood confidence intervals.

PLSINGULAR=*value*
specifies the tolerance for testing the singularity of the Hessian matrices that are computed during the profile-likelihood confidence interval searches. The test requires that a pivot for sweeping this matrix be at least the *value* times a norm of the matrix, where *value* is greater than 0. By default, *value* is the machine epsilon times 1E7, which is approximately 1E–9. This option has no effect if you do not request profile-likelihood confidence intervals.

RESTORE=*libref.data-table*
specifies the name of the analytic store that contains the context and results of a model that is selected and stored from a previous statistical analysis. The analytic store is created by a STORE statement from a previous PROC LOGSELECT call or by a **`store`** parameter that is specified in a previous **`logistic`** action call.

The displayed output can include the following content:

- information about the analytic store
- notes that are stored by the TEXT= option of the STORE statement
- a Replay section, consisting of tables that are produced when the stored model is fit
- a Restore section, consisting of tables that are produced by applying the stored model to a specified input data table

See Example 18.7 for further discussion of these outputs.

You can specify the STB and CLB options in the PROC LOGSELECT statement to modify the "Parameter Estimates" table; you can specify the CORRB, COVB, and TYPE3 options to display those tables; you can specify the CODE statement to produce SAS code for scoring; you can specify LSMEANS statements to compute least squares means; and you can specify ODDSRATIO statements to compute odds ratios. If you specify a DATA= data table, then you can also specify the ASSOCIATION, CTABLE, LACKFIT, and PARTFIT options, and you can score that data table by specifying the OUTPUT statement.

If the analytic store is created using BY-group processing, then the results are displayed according to each BY group. If you specify a DATA= input data table, then you must also specify the same BY statement that was used to create the store.

If a PARTITION statement is used to create the stored model, then the Replay section displays only information that is computed from the original training data. For the Restore section, if you create the partition for the stored model by using a ROLE variable and also specify the FITDATA option, then the ROLE variable must exist in the DATA= table and the Restore section displays results according to the roles. However, if the partitions for the stored model are created using a FRACTION option, then the roles are ignored because the roles cannot be reliably re-created; this means that output statistics that rely on the input data are not computed.

Because the selected model from a previous analysis is stored, statements and options specific to model specification, selection, and optimization are not available. In particular, the CLASS, EFFECT, FREQ, MODEL, PARTITION, REPEATED, SELECTION, STORE, and WEIGHT statements are not available to be used with the RESTORE= option, and the MULTIPASS= option is always set to TRUE.

libref.data-table is a two-level name, where *libref* refers to the library, and *data-table* specifies the name of the input data table. For more information about this two-level name, see the DATA= option and the section "Using CAS Sessions and CAS Engine Librefs" on page 963.

SEED=*number*

specifies an integer to be used to start the pseudorandom number generator for the LSMEANS statement's ADJUST=SIMULATE option and for the PARTITION statement's FRACTION option. If you do not specify a seed, or if you specify a *number* less than or equal to 0, the seed is generated by reading the time of day from the computer's clock.

STB

displays the standardized estimates of the parameters in the "Parameter Estimates" table. The standardized estimate of β_i is given by $\widehat{\beta}_i/(s/s_i)$, where s_i is the total sample standard deviation for the ith explanatory variable and

$$s = \begin{cases} \pi/\sqrt{3} & \text{LOGIT and GLOGIT links} \\ 1 & \text{PROBIT link} \\ \pi/\sqrt{6} & \text{CLOGLOG and LOGLOG links} \end{cases}$$

The sample standard deviations for parameters that are associated with CLASS variables are computed using their codings. The standardized estimates are not computed for the intercept parameters.

TYPE3

computes Wald statistics for Type 3 contrasts for each effect that you specify in the MODEL statement. This option is not available when you specify a REPEATED statement. The option is also not available when you use either LASSO selection or elastic net selection. For more information, see the section "Joint Tests and Type 3 Tests" on page 1031.

USELASTITER

continues to perform computations by using the last iteration of the optimization when the optimization fails. By default, computations cease when a failure occurs.

BY Statement

BY *variables* **;**

You can specify a BY statement in PROC LOGSELECT to obtain separate analyses of observations in groups that are defined by the values of the BY variables. If you specify more than one BY statement, only the last one specified is used. For more information, see the discussion of BY-group processing in the "Grouping Data" section of *SAS Programmers Guide: Essentials*.

CLASS Statement

CLASS *variable* < **(***options***)** >. . . < *variable* < **(***options***)** > > < / *global-options* > **;**

The CLASS statement names the classification variables to be used as explanatory variables in the analysis. You can list the response variable for binary models in the CLASS statement, but this is not required. Table 18.3 summarizes the values that you can use for either an *option* or a *global-option*. The options are fully documented in the section "CLASS Statement" on page 9 in Chapter 2, "Shared Concepts."

Table 18.3 CLASS Statement Options

Option	Description
DESCENDING	Reverses the sort order
MISSING	Treats missing values as valid levels
ORDER=	Specifies the sort order for the levels
PARAM=	Specifies the parameterization of the variable
REF=	Specifies the reference level of the variable
SPLIT	Splits levels of CLASS variables into independent effects

CODE Statement

CODE < *options* > **;**

The CODE statement writes SAS DATA step code for computing predicted values of the fitted model to a file or to a table. To score new data, you can then include the file in a DATA step, or you can specify the table in the **`runCodeTable`** action in the **`dataStep`** action set (for more information, see *SAS Viya: System Programming Guide*).

Table 18.4 summarizes the *options* available in the CODE statement.

Table 18.4 CODE Statement Options

Option	Description
COMMENT	Adds comments to the generated code
FILE=	Names the file in which to save the generated code
FORMATWIDTH=	Specifies the numeric format width for the regression coefficients
INDENTSIZE=	Specifies the number of spaces to indent the generated code
IPROB	Computes individual predicted probabilities for ordinal response models
LABELID=	Specifies a number used to construct names and labels
LINESIZE=	Specifies the line size for the generated code
NOTRIM	Compares formatted values, including blank padding
OUT=	Names an output table in which to save the generated code
PCATALL	Generates probabilities for all levels of categorical response variables

For more information about the syntax of the CODE statement, see the section "CODE Statement" on page 13 in Chapter 2, "Shared Concepts."

DISPLAY Statement

DISPLAY < *table-list* > < / *options* > **;**

The DISPLAY statement enables you to specify a list of display tables to display or exclude. This statement is similar to the ODS SELECT, ODS EXCLUDE, and ODS TRACE statements. However, the DISPLAY statement can improve performance when a large number of tables could be generated (such as in BY-group processing). The procedure processes the DISPLAY statement on a CAS server and thus sends only a subset of ODS tables to the SAS client. Because ODS statements are processed on a SAS client, first all the generated display tables are sent to the client, and then the client creates a subset.

If you use both DISPLAY and ODS statements together, the DISPLAY statement takes precedence over the ODS statements. Note that the ODS EXCLUDE statement processes tables that are sent to the client after they have been filtered by the DISPLAY statement. In some cases, it might appear that the ODS EXCLUDE statement is taking precedence because it can further filter the tables. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

You can specify the *table-list* as a list of table names, paths, partial pathnames, and regular expressions.

The table names that you can specify are listed in the section "ODS Table Names" on page 1044. A path is a table name that is prefixed with dot-separated grouping information. For example, a SelectionSummary table that a procedure produces during a selection routine might have the path *Bygroup1.Summary.SelectionSummary*. A partial pathname does not include all groups; for example, *SelectionSummary* and *Summary.SelectionSummary* are partial pathnames for *Bygroup1.Summary.SelectionSummary*.

When you specify a table name or partial pathname, all display tables whose paths end in the specified name are selected for display or exclusion. For example, both *SelectionSummary* and *Summary.SelectionSummary* select *Bygroup1.Summary.SelectionSummary*.

A regular expression is enclosed in forward slashes (/). For example, specifying "/tions/" selects all pathnames that contain the substring "tions"; in particular, the *Bygroup1.Summary.SelectionSummary* table is selected.

Specifying "!/tions/" selects all pathnames that do not contain the substring "tions"; in particular, the *Bygroup1.Summary.SelectionSummary* table is not selected.

You can specify the following *options* after a slash (/):

CASESENSITIVE
performs a case-sensitive comparison of table names in the *table-list* to display table names when tables are subsetted for display. To preserve case, you must enclose table names in the *table-list* in quotation marks.

EXCLUDE
displays all display tables except those that you specify in the *table-list*.

EXCLUDEALL
suppresses display of all tables. This option takes precedence over the other options.

TRACE
displays the display table names, labels, and paths.

DISPLAYOUT Statement

DISPLAYOUT *table-spec-list* < / *options* > **;**

The DISPLAYOUT statement enables you to create output tables from your displayed output. This statement is similar to the ODS OUTPUT statement. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

The *table-spec-list* specifies a list of output tables to create. Each entry in the list has either a *key*=*value* format or a *key* format:

key=*value*	specifies *key* as the ODS table name, path, or partial pathname, and specifies *value* as the output table name.
key	specifies *key* as the ODS table name and also as the output table name.

The ODS table names that you can specify are listed in the section "ODS Table Names" on page 1044. You cannot specify the ODS table named OutputCasTables in the *table-spec-list*.

Table names and partial pathnames are discussed under the DISPLAY statement. The DISPLAYOUT statement does not support regular expressions.

You can specify the following *options* after a slash (/):

INCLUDEALL
creates output tables for all display tables. The name of the created output table is the same as the name of the corresponding display table. If you specify this option, the *table-spec-list* specification is ignored.

NOREPLACE
does not replace any existing output table of the same name.

REPEATED
replicates all output tables on all nodes.

EFFECT Statement

EFFECT *name*=*effect-type* **(***variables* < / *options* >**) ;**

The EFFECT statement enables you to construct special collections of columns for design matrices. These collections are referred to as *constructed effects* to distinguish them from the usual model effects that are formed from continuous or classification variables, as discussed in the section "GLM Parameterization of Classification Variables and Effects" on page 67 in Chapter 2, "Shared Concepts."

You can specify the following *effect-types*:

COLLECTION specifies a collection effect that defines one or more variables as a single effect that has multiple degrees of freedom. The variables in a collection are considered as a unit for purposes of estimation and inference.

MULTIMEMBER | MM specifies a multimember classification effect whose levels are determined by one or more variables that appear in a CLASS statement.

POLYNOMIAL | POLY specifies a multivariate polynomial effect in the specified numeric variables.

SPLINE specifies a regression spline effect whose columns are univariate spline expansions of one or more variables. A spline expansion replaces the original variable with an expanded or larger set of new variables.

Table 18.5 summarizes the *options* available in the EFFECT statement.

Table 18.5 EFFECT Statement Options

Option	Description
Collection Effects Options	
DETAILS	Displays the constituents of the collection effect
Multimember Effects Options	
DETAILS	Displays the levels of the multimember effect
NOEFFECT	Specifies that observations whose levels are all missing for the multimember variables should have 0 values in the corresponding design matrix columns
STDIZE	Standardizes the design matrix entries so that each observation has a sum of 1
WEIGHT=	Specifies the weight variable for the contributions of each classification effect

Table 18.5 *continued*

Option	Description
Polynomial Effects Options	
DEGREE=	Specifies the degree of the polynomial
DETAILS	Displays details of the specified polynomial
MDEGREE=	Specifies the maximum degree of any variable in a term of the polynomial
NOSEPARATE	Treats the polynomial as a single effect with multiple degrees of freedom
STANDARDIZE=	Specifies centering and scaling suboptions for the variables that define the polynomial
Spline Effects Options	
BASIS=	Specifies the type of basis (B-spline basis or truncated power function basis) for the spline effect
DATABOUNDARY	Uses the extremes of the data as boundary knots for a B-spline basis
DEGREE=	Specifies the degree of the spline effect
DETAILS	Displays the knots and locations for each spline basis function
KNOTMAX=	Requests equally spaced right-side boundary knots starting at the variables' maximum and ending at the KNOTMAX= value
KNOTMETHOD=	Specifies how to construct the knots for the spline effect
KNOTMIN=	Requests equally spaced left-side boundary knots starting at the KNOTMIN= value and ending at the variables' minimum value
NATURALCUBIC	Specifies a natural cubic spline basis for the spline effect
SEPARATE	Treats the spline basis for each variable as a separate effect when multiple variables are specified
SPLIT	Treats each design matrix column as a separate effect for selection methods

For more information about the syntax of these *effect-types* and how columns of constructed effects are computed, see the section "EFFECT Statement" on page 18 in Chapter 2, "Shared Concepts."

FREQ Statement

FREQ *variable* **;**

The *variable* in the FREQ statement identifies a numeric variable in the input data table that contains the frequency of occurrence of each observation. PROC LOGSELECT treats each observation as if it appears f times, where f is the value of the FREQ *variable* for the observation. If f is not an integer, it is truncated to an integer. If f is less than 1 or missing, the observation is not used in the analysis. When you omit the FREQ statement, each observation is assigned a frequency of 1.

LSMEANS Statement

LSMEANS *model-effects* < / *options* > **;**

LSMEANS EFFECT=*model-effects* < / *options* > **;**

The LSMEANS statement computes and compares least squares means (LS-means) of fixed effects. LS-means are *predicted population margins*—that is, they estimate the marginal means over a balanced population. In a sense, LS-means are to unbalanced designs as class and subclass arithmetic means are to balanced designs.

LS-means can be computed for any effect in the procedure's MODEL statement that involves only classification variables. You can specify multiple *model-effects* in one or more LSMEANS statements that appear after the MODEL statement. When you restore a model from an item store, you must specify the desired *model-effects* in the EFFECT= option before the slash (/).

Table 18.6 summarizes the *options* available in the LSMEANS statement.

Table 18.6 LSMEANS Statement Options

Option	Description
Construction and Computation of LS-Means	
AT	Modifies covariate values used to compute LS-means
ATVAR	Modifies covariate values for restored models
DIFF	Computes differences of LS-means
SINGULAR=	Tunes estimability checking
***p*-Values**	
ADJUST=	Specifies the multiple comparison adjustment method for LS-means differences
ALPHA=α	Specifies the confidence level ($1-\alpha$)
Statistical Output	
CL	Constructs confidence limits for means and mean differences
CORR	Displays the correlation matrix of LS-means
COV	Displays the covariance matrix of LS-means
E	Displays the **L** matrix

For more information about the syntax of the LSMEANS statement, see the section "LSMEANS Statement" on page 34 in Chapter 2, "Shared Concepts."

LS-means analysis is not available if you use LASSO or elastic net selection.

MODEL Statement

MODEL *response* < **(***response-options***)** > **=** < *effects* > < / *model-options* > **;**

MODEL *events* / *trials* **=** < *effects* > < / *model-options* > **;**

MODEL *response-level1 response-level2* < . . . *response-levelJ* > **=** < *effects* > < / *model-options* > **;**

The MODEL statement defines the statistical model in terms of a response or target, model *effects* that are constructed from variables in the input data table, and *model-options*. An intercept is included in the model by default. You can remove the intercept by specifying the NOINT option.

Three forms of the response can be specified. For the first form, *single-trial* syntax, you can specify a single *response* variable that contains your response values. Single-trial syntax is used when each observation in the DATA= data table contains information about only a single trial, such as a single subject in an experiment, and is applicable to binary, ordinal, and nominal response data.

The second form, *events/trials* syntax, is applicable to binomial response data, where one variable contains the number of positive responses (or events) and another variable contains the number of trials. Note that the values of both *events* and (*trials* – *events*) must be nonnegative and the value of *trials* must be positive.

The third form, *multinomial-trial* syntax, is used when each observation contains information about multiple trials, and you have one variable (*response-level1*) that contains the number of responses in the first level, another variable (*response-level2*) that contains the number of responses in the second level, and so on. The last level in the response-level list is the reference level. Multinomial-trial syntax is applicable to binomial, ordinal, and nominal response data. Note that the values of the *response-level1* through *response-levelJ* variables must be nonnegative.

For information about constructing the model effects, see the section "Specification and Parameterization of Model Effects" on page 64 in Chapter 2, "Shared Concepts."

There are two sets of options in the MODEL statement. The *response-options* determine how the LOGSELECT procedure models probabilities for binary and multinomial data when you use single-trial syntax. The *model-options* control other aspects of model formation and inference. Table 18.7 summarizes these options.

Table 18.7 MODEL Statement Options

Option	Description
Response Variable Options	
DESCENDING	Reverses the response categories
EVENT=	Specifies the event category
ORDER=	Specifies the sort order
REF=	Specifies the reference category
Model Options	
CENTER	Centers and scales continuous main effects
CENTERLASSO	Centers and scales all effects for model selection by the LASSO method and the elastic net method
CLB	Requests Wald confidence limits
INCLUDE=	Includes effects in all models for model selection

Table 18.7 *continued*

Option	Description
INFORMATIVE	Models missing values by using extra indicator variables
LINK=	Specifies the link function
NOINT	Suppresses the intercept
OFFSET=	Specifies the offset variable
PRIOR=	Specifies prior probabilities
START=	Includes effects in the initial model for model selection
TYPE3	Displays the Type 3 or joint tests of effects

Response Variable Options

Response variable options determine how the LOGSELECT procedure models probabilities for binary and multinomial response data when you specify single-trial syntax. You can specify the following *response-options* by enclosing them in parentheses after the *response* or *trials* variable. These options are not available with multinomial-trial syntax.

DESCENDING
DESC
reverses the order of the response categories. If you specify both the DESCENDING and ORDER= options, PROC LOGSELECT orders the response categories according to the ORDER= option and then reverses that order.

EVENT='*category*' | **FIRST** | **LAST**
specifies the event category for the binary response model. PROC LOGSELECT models the probability of the event category. The EVENT= option has no effect when there are more than two response categories.

You can specify one of the following:

'*category*'
specifies the value (formatted, if a format is applied) of the event category in quotation marks.

FIRST
designates the first ordered category as the event.

LAST
designates the last ordered category as the event.

By default, EVENT=FIRST.

For example, the following statements specify that observations with the formatted value '1' represent events in the data. The probability that the LOGSELECT procedure models is thus the probability that the variable def takes on the (formatted) value '1'.

```
proc logselect data=mylib.MyData;
   class A B C;
   model def(event ='1') = A B C x1 x2 x3;
run;
```

ORDER=FORMATTED | FREQ | INTERNAL | '*level1*' '*level2*' . . .
specifies the sort order for the levels of the *response* variable. When ORDER=FORMATTED for numeric variables for which you have supplied no explicit format (that is, for which there is no corresponding FORMAT statement in the current PROC LOGSELECT run or in the DATA step that created the data table), the levels are ordered by their internal (numeric) value. The following table shows how to interpret the ORDER= keywords:

ORDER=	Levels Sorted By
FORMATTED	External formatted value, except for numeric variables with no explicit format, which are sorted by their unformatted (internal) value; the sort order is machine-dependent.
FREQ	Descending frequency count (levels with the most observations come first in the order)
INTERNAL	Unformatted value; the sort order is machine-dependent.

By default, ORDER=FORMATTED.

You can also specify a particular order for the response variable by listing the levels in quotes in the desired order. For example:

```
model pet(order = 'dog' 'cat') = ...;
```

Any level of the pet variable that you do not specify is added to the end of the list in formatted order.

The ORDER= option does not apply to missing values, which are always ordered first.

For more information about sort order, see the chapter on the SORT procedure in the *Base SAS Procedures Guide* and the discussion of BY-group processing in *SAS Programmers Guide: Essentials*.

REF='*category*' **| FIRST | LAST**
specifies the reference category for the generalized logit model and the binary response model. For the generalized logit model, each logit contrasts a nonreference category with the reference category. For the binary response model, specifying one response category as the reference is the same as specifying the other response category as the event. You can specify one of the following:

'*category*'
specifies the value (formatted, if a format is applied) of the reference category in quotation marks.

FIRST
designates the first ordered category as the reference.

LAST
designates the last ordered category as the reference.

By default, REF=LAST.

Model Options

CENTER

requests that continuous main effects be centered and scaled internally. (Continuous main effects are centered and scaled to aid in computing maximum likelihood estimates.) Parameter estimates and related statistics are always reported on the original scale. This option has no effect if you specify the LASSO model selection method or the elastic net selection method; instead, you should specify the CENTERLASSO option.

CENTERLASSO< =TRUE | FALSE >

specifies whether all effects, including categorical effects, are centered and scaled internally for the LASSO model selection method or the elastic net selection method. (Effects are centered and scaled to aid in model selection by the LASSO method and the elastic net method.) Parameter estimates and related statistics are always reported on the original scale. By default, CENTERLASSO=TRUE.

CLB

displays Wald confidence limits for each of the parameter estimates. The confidence level is 0.95 by default; you can change it by specifying the ALPHA= option. The CLB option is not available when you use either LASSO selection or elastic net selection.

INCLUDE=*n*

INCLUDE=*single-effect*

INCLUDE=(*effect-list***)**

forces effects to be included in all models. If you specify INCLUDE=*n*, then the first *n* effects that are listed in the MODEL statement are included in all models. If you specify INCLUDE=*single-effect* or if you specify INCLUDE=(*effect-list*), then the specified effects are forced into all models. The effects that you specify in this option must be explanatory effects that are specified in the MODEL statement before the slash (/). This option is not available if you use elastic net selection.

INFORMATIVE

models missing values by using extra model effects. These effects consist of dummy variables that take the value 1 when the value of a continuous model variable involved in the effect is missing, and take the value 0 otherwise. The missing value in the original model effect is replaced by the average value of the effect for the nonmissing values. For continuous-by-class effects, such as A*x, where A is a classification variable and x is a continuous variable, informative missingness creates multiple dummy columns and substitutes the effect mean of x that corresponds to the respective level of A. Missing values for classification variables are treated as valid levels. For more information about informative missingness, see the section "Informative Missingness" on page 99 in Chapter 2, "Shared Concepts."

LINK=*keyword*

specifies the link function for the model. The default link is the logit. The *keywords* and the associated link functions are shown in Table 18.8.

Table 18.8 Built-In Link Functions of the LOGSELECT Procedure

Keyword	Link Function	$g(\pi) = \eta =$
CLOGLOG | CLL	Complementary log-log	$\log(-\log(1-\pi))$
GLOGIT	Generalized logit	$\log(\pi_j/\pi_J)$
LOGIT	Logit	$\log(\pi/(1-\pi))$
LOGLOG	Log-log	$-\log(-\log(\pi))$
PROBIT	Probit	$\Phi^{-1}(\pi)$

For the probit and cumulative probit links, $\Phi^{-1}(\cdot)$ denotes the quantile function of the standard normal distribution.

If the response variable has more than two categories, the LOGSELECT procedure fits a model that has a cumulative link function that is based on the specified link. However, if you specify LINK=GLOGIT, the procedure assumes a generalized logit model for nominal (unordered) data.

NOINT
requests that no intercept be included in the model. An intercept is included by default. The NOINT option is not available for multinomial models.

OFFSET=*variable*
specifies a *variable* to be used as an offset to the linear predictor. An offset plays the role of an effect whose coefficient is known to be 1. The offset variable cannot appear in the CLASS statement or elsewhere in the MODEL statement. Observations that have missing values for the offset variable are excluded from the analysis.

PRIOR=*number* | *number-list*
specifies prior probabilities (prevalences) that are used for computing posterior predicted probabilities. When you know what percentage of the population has a rare event and you oversample that rare event, specifying the prior probabilities as the prevalence of events in your population enables you to produce posterior probabilities that reflect the population, not the data. The priors do not affect the model-fitting process.

If you specify a *number-list* that has the same number of entries as there are response levels, then these values are scaled to sum to 1. If you specify a *number-list* that has one entry less than there are response levels, then the specified values should sum to less than 1, and the last response level is assigned the remaining value. For binary and binomial response models, you can thus specify the probability of an event as a single *number*. The scaled priors are displayed in the ResponseProfile table.

If your response Y takes values $i = 1, \ldots, k$ that have observed empirical training probabilities $\mathrm{OldPrior}_i = \frac{n_i}{n}$, you specify priors Prior_i, and your model predicted probabilities are $\hat{p}_i$, then the posterior predicted probabilities Post_i are computed as

$$\mathrm{Post}_i = \frac{\hat{p}_i \frac{\mathrm{Prior}_i}{\mathrm{OldPrior}_i}}{\sum_{j=1}^{k} \hat{p}_j \frac{\mathrm{Prior}_j}{\mathrm{OldPrior}_j}}$$

The POST= option in the OUTPUT statement writes the posterior to the output data table. If your priors are identical to the empirical probabilities, then the posteriors are identical to the model-predicted probabilities.

The priors adjust the "Classification" table statistics PPV, NPV, adequacy, percentage correct, misclassification rate, and lift as discussed in the section "Classification Table and ROC Curves" on page 1022. If you specify a PARTITION statement, then the posterior probabilities and the preceding statistics for the validation and test data also use the specified priors.

START=*n*

START=*single-effect*

START=(*effects***)**

begins the selection process from the designated initial model for the forward and stepwise selection methods. If you specify START=*n*, then the starting model includes the first *n* effects that are listed in the MODEL statement. If you specify START=*single-effect* or if you specify START=(*effects*), then the starting model includes those specified effects. The effects that you specify in the START= option must be explanatory effects that are specified in the MODEL statement before the slash (/). This option is not available when you specify METHOD=BACKWARD in the SELECTION statement or when you specify METHOD=ELASTICNET in the SELECTION statement.

TYPE3

computes Wald statistics for Type 3 contrasts for each effect that is specified in the MODEL statement. This option is not available when you specify a REPEATED statement. For more information, see the section "Joint Tests and Type 3 Tests" on page 1031.

ODDSRATIO Statement

ODDSRATIO < *variable* < **(***options***)** > > . . . < *variable* < **(***options***)** > > < / *global-options* > **;**

The ODDSRATIO statement produces odds ratios for each odds ratio *variable* that you use in your model even when the odds ratio variable is involved in interactions with other variables or is used by a constructed effect. You can specify at most one ODDSRATIO statement. If you do not specify any odds ratio *variables*, then odds ratios are computed for every variable in the model. Odds ratio estimation is not available if you use elastic net selection. For more information, see the section "Odds Ratio Estimation" on page 1033.

If your odds ratio *variable* is continuous, then by default the odds ratios compare the base subpopulation to one where that continuous variable is one unit larger. If your odds ratio *variable* is a classification variable, then by default the odds ratios compare each level of the odds ratio variable to its reference level.

If an odds ratio *variable* interacts with another variable, then the computed odds ratio depends on the value of the interacting variable. By default, interacting continuous variables are set to their means, and interacting classification variables are set to their reference level. The computed odds ratios are independent of the parameterization of any classification variable.

You can specify *options* for each individual odds ratio variable by enclosing the options in parentheses after the variable name. You can change the default options by specifying *global-options* after a slash (/). *Global-options* are applied to all the odds ratio variables; however, individual odds ratio variable *options* override the *global-options*.

You can specify the following *options* for the individual odds ratio requests in parentheses after the odds ratio *variable*:

AT(*var***=***var-options* **<...** *var***=***var-options***>)**
enables you to specify fixed values or levels for a variable in the model when it interacts with the odds ratio variable. The following *var-options* are enabled when the *var* variable interacts with the odds ratio variable:

value-list
specifies fixed values for the variable *var* when *var* interacts with the odds ratio variable.

When *var* is a continuous variable, you can specify one or more numbers in the *value-list*. By default, continuous variables are set to their means. When *var* is a classification variable, you can specify one or more formatted levels of the variable enclosed in single quotes (for example, **A='cat' 'dog'**). By default, CLASS variables are set to their reference level. For example, if your model includes a classification variable A={cat,dog} and a continuous variable B, then specifying either of the following statements (the second uses a *global-option*) sets A to 'cat' and sets B to 7 and then 9:

```
oddsratio X ( AT( A='cat' B=7 9 ) );
oddsratio X / AT( A='cat' B=7 9 );
```

When *var* is a continuous variable, the values in the *value-list* are also used to construct SPLINE and POLYNOMIAL effects of that variable.

REF
specifies the reference level of *var* when *var* is a classification variable. This is the default for classification variables.

ALL
specifies every level of *var* when *var* is a classification variable.

UNIT=*unit-list*
specifies one or more numbers that represent the units of change for a continuous odds ratio variable so that you can produce customized odds ratios. An estimate of the corresponding odds ratio is displayed for each specified unit of change. The UNIT= *option* is ignored for CLASS *variables*. For example, specifying either of the following statements (the second uses a *global-option*) requests an odds ratio that represents the change in the odds when the variable *X* is increased by two units:

```
oddsratio X ( UNIT=2 );
oddsratio X / UNIT(X=2);
```

By default, UNIT=1.

STDERR
indicates that the units of change specified in the *unit-list* of the UNIT= option represent multiples of the sample standard error of the odds ratio variable, based on the observations that are used to fit the model. If you specify the STDERR option without the UNIT= option, then the unit of change is 1 standard error.

You can also specify the following *global-options* after a slash (/):

ALPHA=*number*
sets the significance level for producing confidence intervals. The ALPHA= value that you specify in the PROC LOGSELECT statement is the default. If neither ALPHA= value is specified, then by default ALPHA=0.05.

AT(*var***=***var-options* **< . . .** *var***=***var-options* **>)**
enables you to specify fixed levels or values for variables in the model that might interact with the odds ratio variable. See the AT= individual option for more information.

CL< =WALD | PL >
displays Wald confidence limits. If you specify CL=PL, then profile-likelihood confidence limits are also displayed. The confidence level is 0.95 by default; you can change it by specifying the ALPHA= option.

DIFF=REF | ALL
specifies whether the odds ratios for a classification odds ratio variable are computed against the reference level, or whether all pairs of the odds ratio variable's levels are compared. By default, DIFF=REF. The DIFF= option has no effect when the odds ratio variable is continuous.

UNIT(*var***=***unit-list* **< . . .** *var***=***unit-list* **> < STDERR >)**
enables you to change the default units of change for continuous odds ratio variables. You specify *var* as the name of the continuous variable and *unit-list* as a list of the units. The STDERR option indicates that the *unit-list* represents multiples of the sample standard error of the variable. See the UNIT= and STDERR individual options for more information.

OUTPUT Statement

OUTPUT OUT=*libref.data-table*
< *options* >
< *keyword* < **=***name* > > . . . < *keyword* < **=***name* > > **;**

The OUTPUT statement creates a data table that contains observationwise statistics that PROC LOGSELECT computes after fitting the model. In order to avoid data duplication for large data tables, the variables in the input data table are *not* included in the output data table unless you specify them in the COPYVAR= option.

If your response variable has more than two categories, you can request the "Basic Options" listed in Table 18.9; the other diagnostic statistics are not available unless you use multinomial-trial syntax. These statistics are computed for every response category, and the automatic variable _LEVEL_ identifies the response category on which the computed values are based. That is, every observation generates several rows in the output data set. If you also specify the OBSCAT option, then the observationwise statistics are computed only for the observed response category, which is indicated by the value of the _LEVEL_ variable. If you specify the PREDPROBS option, then only the PRED and IPRED statistics are output to the data table, and a variable is created to contain the values that correspond to each response category. PROC LOGSELECT names these variables by concatenating the PRED= or IPRED= *name* with the response category.

If you use multinomial-trial syntax, then the PREDPROBS option is invoked by default, and all the "Diagnostic Options" listed in Table 18.9 (except for the RESWORK statistic) are available as output to the data table. However, if you also specify the NOPREDPROBS option, then only the RESCHI, RESDEV, and RESRAW statistics are available.

The output statistics are computed based on the final parameter estimates. If the optimization does not converge, then the output data table is not created. If you perform elastic net or LASSO selection, or if you specify the NOXPX or NOSTDERR option, then your analysis does not generate a covariance matrix, and you cannot produce the basic and diagnostic statistics in Table 18.9 that require a covariance matrix.

For observations in which only the response variable is missing, values of the linear predictor and the predicted probabilities are computed even though these observations do not affect the model fit. This enables, for example, predicted probabilities to be computed for new observations.

You must specify the following option:

OUT=*libref.data-table*
names the output data table for PROC LOGSELECT to use. You must specify this option before any other options. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to where the data table is to be stored, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 963.

data-table specifies the name of the output data table.

You can also specify the following options:

ALL
ALLSTAT
adds all available statistics to the output data table.

ALPHA=*number*
specifies the significance level for the construction of confidence intervals in the output data table. The confidence level is $1 - number$. The value of *number* must be between 0 and 1. By default, *number* is equal to the value of the ALPHA= option in the PROC LOGSELECT statement, or 0.05 if that option is not specified.

COPYVAR=*variable*
COPYVARS=(*variables***)**
transfers one or more *variables* from the input data table to the output data table.

FIXEDOFFSET=*number*
specifies a value to use as an offset to the linear predictor instead of obtaining the value from the offset variable. This option is ignored unless you also specify the OFFSET= variable in the MODEL statement.

OBSCAT
requests (for multinomial models) that observationwise statistics be produced only for the observed response level. If you do not specify this option and the response variable has J levels, then the following outputs are created: for cumulative link models, J–1 records are output for every observation in the input data that corresponds to the J–1 lower-ordered response categories; for generalized logit models, J records are output that correspond to all J response categories. This option is not available with multinomial-trial syntax.

PREDPROBS | NOPREDPROBS

creates variables (for multinomial models) for each response category that corresponds to the requested IPRED and PREDICTED keywords. If you use multinomial-trial syntax, then the PREDPROBS option is invoked by default and enables computation of most of the "Diagnostic Options" listed in Table 18.9. You can override this behavior by specifying the NOPREDPROBS option.

keyword **<=***name***>**

specifies a statistic to include in the output data table and optionally names the variable *name*. If you do not provide a *name*, the LOGSELECT procedure assigns a default name based on the type of statistic requested. If a statistic is not available for your model then that request is ignored.

Table 18.9 summarizes the *keywords* that are available in the OUTPUT statement.

Table 18.9 OUTPUT Statement Keywords

Keyword	Description	Default Name
Basic Options		
INDIVIDUAL	Specifies the individual predicted probabilities	_IPRED_
POST	Specifies the posterior predicted probabilities	_POST_
PREDICTED	Specifies the predicted probabilities	_PRED_
RESRAW	Specifies the raw residual	_RESRAW_
XBETA	Specifies the linear predictor	_XBETA_
Basic Options Requiring the Covariance		
LCL	Specifies the lower confidence limit for the linear predictor	_LCL_
LCLM	Specifies the lower confidence limit for the event probability	_LCLM_
STDXBETA	Specifies the standard error estimate of the linear predictor	_STDXBETA_
UCL	Specifies the upper confidence limit for the linear predictor	_UCL_
UCLM	Specifies the upper confidence limit for the event probability	_UCLM_
Diagnostic Options		
RESCHI	Specifies the Pearson chi-square residual	_RESCHI_
RESDEV	Specifies the deviance residual	_RESDEV_
RESWORK	Specifies the working residual	_RESWORK_
Diagnostic Options Requiring the Covariance		
CBAR	Specifies the confidence interval displacement	_CBAR_
DIFCHISQ	Specifies the deletion chi-square goodness-of-fit change	_DIFCHISQUARE_
DIFDEV	Specifies the deletion deviance change	_DIFDEVIANCE_
H	Specifies the leverage	_HATDIAG_
RESLIK	Specifies the likelihood residual	_RESLIK_
STDRESCHI	Specifies the standardized Pearson chi-square residual	_STDRESCHI_
STDRESDEV	Specifies the standardized deviance residual	_STDRESDEV_
Option for Generalized Estimating Equations		
CLEVERAGE	Names the cluster leverage	_CLEVERAGE_

Table 18.9 *continued*

Keyword	Description	Default Names
Miscellaneous Options		
INTO	Names the level into which the observation is classified	_INTO_
LEVEL	Names the response level for a row of the output	_LEVEL_
ROLE	Names the role played by the observation in fitting the model	_ROLE_

The following list describes these *keywords*. For more information, see the section "Predicted Probabilities and Regression Diagnostics" on page 1026.

CBAR
specifies the confidence interval displacement diagnostic that measures the overall change in the global regression estimates that results from deleting an individual observation. The default name is _CBAR_.

CLEVERAGE

CH

CLEV
specifies the cluster leverage for GEEs. For more information about diagnostic statistics, see the section "Diagnostics for Models Fit by Generalized Estimating Equations (GEEs)" on page 1030. The default name is _CLEVERAGE_.

DIFCHISQ
specifies the change in the chi-square goodness-of-fit statistic that results from deleting the individual observation. The default name is _DIFCHISQUARE_.

DIFDEV
specifies the change in the deviance that results from deleting the individual observation. The default name is _DIFDEVIANCE_. This statistic is not supported for GEEs.

H
specifies the diagonal element of the hat matrix $\mathbf{H}$ (leverage) for detecting extreme points in the design space. For GEEs, the leverage value is the corresponding diagonal element of the cluster leverage matrix. The default name is _HATDIAG_.

If you use multinomial-trial syntax and do not specify the NOPREDPROBS option, then the relevant portion of the hat matrix for observation i is a J–1-dimensional submatrix $\mathbf{H}_i$. Three statistics are computed from this submatrix: the determinant of $\mathbf{I}_{J-1} - \mathbf{H}_i$, the average diagonal element of $\mathbf{H}_i$, and the potential (average diagonal element of $(\mathbf{I}_{J-1} - \mathbf{H}_i)^{-1}\mathbf{H}_i$). These statistics are identified in the output table by appending the letters 'D', 'T', and 'P', respectively, to the provided *name*; by default, their names are _H_D, _H_T, and _H_P.

INDIVIDUAL

IPRED

IPROB

IP

specifies the individual predicted values for the multinomial response levels. For a response variable Y with three levels, 1, 2, and 3, the individual probabilities are $\Pr(Y = 1)$, $\Pr(Y = 2)$, and $\Pr(Y = 3)$. The default name is _IPRED_.

INTO< (*cutpoint*) >

names the variable that contains the level of the response into which an observation is classified. The default name is _INTO_. Multinomial models classify observations into the level that has the largest model-predicted probability. For binary or binomial response variables, if the predicted probability of an observation equals or exceeds the cutpoint, the observation is classified as an event; otherwise it is classified as a nonevent. You can specify the *cutpoint* value as a number between 0 and 1. The default value is 0.5.

LCL

LOWERXBETA

names the variable that contains the lower confidence limits for the linear predictor. You can set the confidence level by specifying the ALPHA= option. The default name is _LCL_.

LCLM

LOWERMEAN

LOWER

specifies the lower confidence limits for the probability of the event. You can set the confidence level by specifying the ALPHA= option. The default name is _LCLM_.

LEVEL

names the variable that contains the level of the response for a given row of the output. The default name is _LEVEL_.

POSTERIOR

POST

specifies the posterior predicted probability of each observation. If you do not specify the PRIOR= option in the MODEL statement, then the observed proportions from the training data are used as priors, in which case the posteriors from the training data are the same as the individual predicted probabilities. The default name is _POST_.

PREDICTED

PRED

PROB

P

specifies the predicted values (predicted probabilities of events) for binary and nominal response variables and the cumulative predicted probabilities for ordinal response variables. For a response variable Y with three levels, 1, 2, and 3, the cumulative probabilities are $\Pr(Y \leq 1)$ and $\Pr(Y \leq 2)$, but by default the last level, $\Pr(Y \leq 3) = 1$, is not output. The default name is _PRED_.

RESCHI

PEARSON

specifies the Pearson residual for identifying poorly fitted observations. The default name is _RESCHI_.

RESDEV

specifies the deviance residual for identifying poorly fitted observations. The default name is _RESDEV_. This statistic is not supported for GEEs.

RESLIK

specifies the likelihood residual for identifying poorly fitted observations. The default name is _RESLIK_. This statistic is not supported for GEEs.

RESRAW

RESIDUAL

R

specifies the raw residual for identifying poorly fitted observations. The default name is _RESRAW_.

RESWORK

specifies the working residual for identifying poorly fitted observations. This statistic is not available for multinomial response models. The default name is _RESWORK_.

ROLE

specifies the numeric variable that indicates the role played by each observation in fitting the model. The default name is _ROLE_. Table 18.10 shows how this variable is interpreted for each observation.

Table 18.10 Role Interpretation

Value	Observation Role
0	Not used
1	Training
2	Validation
3	Testing

If you do not partition the input data by specifying a PARTITION statement, then the role variable value is 1 for observations that are used in fitting the model and 0 for observations that have at least one missing or invalid value for the response, regressor, frequency, or weight variables.

STDRESCHI

specifies the standardized Pearson (chi-square) residual for identifying observations that are poorly accounted for by the model. The default name is _STDRESCHI_.

STDRESDEV
specifies the standardized deviance residual for identifying poorly fitted observations. The default name is _STDRESDEV_. This statistic is not supported for GEEs.

STDXBETA
specifies the standard error estimates of XBETA. The default name is _STDXBETA_.

UCL

UPPERXBETA
specifies the variable that contains the upper confidence limits for the linear predictor. The default name is _UCL_. You can set the confidence level by specifying the ALPHA= option.

UCLM

UPPERMEAN

UPPER
specifies the variable that contains the upper confidence limits for the probability of the event response. The default name is _UCLM_. You can set the confidence level by specifying the ALPHA= option.

XBETA

LINP
specifies the linear predictor. The default name is _XBETA_.

PARTITION Statement

PARTITION *partition-option* **;**

The PARTITION statement specifies how observations in the input data set are logically partitioned into disjoint subsets for model training, validation, and testing. For more information, see the section "Using Validation and Test Data" on page 101 in Chapter 2, "Shared Concepts." Either you can designate a variable in the input data table and a set of formatted values of that variable to determine the role of each observation, or you can specify proportions to use for randomly assigning observations to each role.

You must specify exactly one of the following *partition-options*:

FRACTION(< **TEST=***fraction* **> <** **VALIDATE=***fraction* **> <** **SEED=***number* **>)**
randomly assigns specified proportions of the observations in the input data table to the roles. You specify the proportions for testing and validation by using the TEST= and VALIDATE= suboptions. If you specify both the TEST= and VALIDATE= suboptions, then the sum of the specified fractions must be less than 1 and the remaining fraction is the proportion of the observations assigned to the training role. The SEED= option specifies an integer that is used to start the pseudorandom number generator for random partitioning of data for training, testing, and validation. If you do not specify SEED=*number* or if *number* is less than or equal to 0, the seed is generated by reading the time of day from the computer's clock.

ROLE=*variable* **(< TEST='***value***' > < TRAIN='***value***' > < VALIDATE='***value***' >)**

ROLEVAR=*variable* **(< TEST='***value***' > < TRAIN='***value***' > < VALIDATE='***value***' >)**

names the *variable* in the input data table whose values are used to assign roles to each observation. This *variable* cannot also appear as an analysis variable in other statements or options. The TEST=, TRAIN=, and VALIDATE= suboptions specify the formatted values of this variable that are used to assign observation roles. If you do not specify the TRAIN= suboption, then all observations whose role is not determined by the TEST= or VALIDATE= suboption are assigned to the training role.

For more information, see the section "Partition Fit Statistics" on page 1020. For an illustration, see Example 18.3.

REPEATED Statement

REPEATED SUBJECT=*<subject-effect>* **/** *< repeated-options >* **;**

The REPEATED statement performs a repeated measures analysis by using generalized estimating equations (GEEs). GEE analysis is available for binomial and binary response models. GEE analysis is not supported when you perform model selection or when you perform a partitioned analysis.

The *repeated-options* specify the working correlation structure, output options, and controls for the iterative fitting algorithm.

Table 18.11 summarizes the *repeated-options* in the REPEATED statement by function.

Table 18.11 REPEATED Statement Options

Option	Description
Basic Options	
SUBJECT=	Identifies subjects in the input data
TYPE=	Specifies the working correlation type
WITHIN=	Specifies a within-subject effect
Options Related to Output	
CORRB	Produces both the model-based and empirical estimated regression parameter correlation matrices
CORRW	Produces the working correlation matrix
COVB	Produces both the model-based and empirical estimated regression parameter covariance matrices
ECORRB	Produces the estimated regression parameter empirical correlation matrix
ECOVB	Produces the estimated regression parameter empirical covariance matrix
MCORRB	Produces the estimated regression parameter model-based correlation matrix
MCOVB	Produces the estimated regression parameter model-based covariance matrix
MODELSE	Produces a parameter estimates table that displays and uses the model-based standard errors
PRINTMLE	Produces a parameter estimates table that displays the maximum likelihood estimates

Basic Options

SUBJECT=*subject-effect*

identifies subjects in the input data set. A *subject-effect* is required; it can contain a single variable, an interaction effect, a nested effect, or a combination of these. Each variable that the *subject-effect* contains must be listed in the CLASS statement. Each distinct level of the effect identifies a different subject. Responses from different subjects are assumed to be independent, but responses within subjects are assumed to be correlated.

You can specify the following *options* after a slash (/):

TYPE=*<type>*

specifies the working correlation type that is used in the repeated measures analysis. The working correlation structure describes the way that a subject's repeated measures are assumed to correlate with each other.

You can specify one of the following working correlation types:

AR | AR(1)
specifies the autoregressive-1 working correlation.

CS | EXCH
specifies the exchangeable working correlation.

IND
specifies the independence working correlation.

MDEP*(m)* **| TOEP***(m)*
specifies the *m*-dependent working correlation. By default, *m* = 1.

UN | UNSTR | UNSTRUCTURED
specifies the unstructured working correlation.

By default, TYPE=IND.

WITHIN=*within-effect*

WITHINSUBJECT=*within-effect*
specifies a within-subject effect for the repeated measures model. This effect must be composed of CLASS variables. The purpose of the within-subject effect is to define an order for the repeated measures that correspond to each subject. This definition can be critical for reproducibility when you use an order-sensitive correlation structure.

If you do not specify a within-subject effect, the order of the repeated measures is determined by the order in which they are read. This order can vary, depending on your computing resources. Therefore, a within-subject effect is required when you specify the autoregressive, *m*-dependent, or unstructured working correlation type. When you use one of these working correlation types, the levels of the within-subject effect must be unique within each subject.

If you use multiple variables to define the within-subject effect, the specification of the variables within this effect will have an impact on the analysis if the working correlation type depends on the ordering of the repeated measures. The autoregressive, *m*-dependent, and unstructured working correlation types use the levelization of the within-subject effect to locate observations within the correlation matrix, and depend on the order that is produced by the levelization.

In particular, if you define the within-subject effect by using the interaction of two or more variables, then the hierarchy of levelization and the resulting ordering of the repeated measures will depend on the order in which you list the variables in the CLASS statement.

For example, assume that you are using the interaction of the variables w1 and w2 as the within-subject effect in a repeated measures analysis. Furthermore, assume that w1 takes the values 'A' and 'B' and w2 takes the values 'L' and 'H'. The following statements list these variables in the order w2, w1 in the CLASS statement:

```
proc logselect data=mylib.health;
   class subject w2 w1;
   model outcome(event='1') = x1 x2;
   repeated subject=subject / type=mdep(2) within=w1*w2;
run;
```

In this case, the order of the variables in the CLASS statement results in a levelization in which the value of w1 changes faster than the value of w2. Table 18.12 shows this levelization.

Table 18.12 Levelization Order When Variable Order Is w2, w1

Value of w1	Value of w2	Levelization Order
A	L	0
B	L	1
A	H	2
B	H	3

If you reverse the order of the variables in the CLASS statement, the levelization has a different hierarchy. In this hierarchy, the value of w2 changes faster than the value of w1. Table 18.13 shows that the levelization order that is assigned to each unique combination of the variables does not match the levelization order shown in Table 18.12.

Table 18.13 Levelization Order When Variable Order Is w1, w2

Value of w1	Value of w2	Levelization Order
A	L	0
B	L	2
A	H	1
B	H	3

Because the specified working correlation structure (*m*-dependent) relies on this ordering, the parameter estimates differ from those that are produced by the ordering shown in Table 18.12. The autoregressive and unstructured working correlation structures also result in different parameter estimates for the different orderings.

If you use two variables to define the within-subject effect, you can use a nested effect to enforce a specific levelization hierarchy. However, the best practice is to use a single variable in the within-subject effect.

Options Related to Output

CORRB
produces both the model-based and empirical estimated regression parameter correlation matrices.

CORRW
produces the estimated working correlation matrix. This matrix shows the estimated correlation between pairs of repeated measurements.

COVB
produces both the model-based and empirical estimated regression parameter covariance matrices.

ECORRB
produces the empirical estimated regression parameter correlation matrix.

ECOVB
produces the empirical estimated regression parameter covariance matrix.

MCORRB
produces the model-based estimated regression parameter correlation matrix.

MCOVB
produces the model-based estimated regression parameter covariance matrix.

MODELSE
produces a table that displays the parameter estimates from the repeated measures analysis, but using the model-based standard error estimates. By default, only a table that uses the empirical standard error estimates is shown.

PRINTMLE
produces a table that displays the parameter estimates and standard errors from maximum likelihood estimation, assuming no within-subject correlation. This table corresponds to the "Parameter Estimates" table that the procedure produces when you do not request a repeated measures analysis.

RISK Statement

RISK < *variable-list* > < / *options* > **;**

The RISK statement produces relative risks or risk differences for *variables* that are used in your model. Because risk statistics directly compare the event probabilities, they provide an easier method than odds ratios for comparing your model's implications for different subjects.

Given a population profile L, recall that the model-predicted event probability from a binary logistic regression model is computed as $p = \Pr(Y = \text{event}|L) = g^{(-1)}(L'\hat{\boldsymbol{\beta}})$, where $g(\cdot)$ is your link function. The risk difference is the difference between the *risk* or event probability p_1 of a subject having one population profile, L_1, and the event probability p_2 of having a different population profile, L_2. Specifying the TYPE= option enables you to compute the relative risk, or risk ratio, which is the quotient of these two probabilities. If you do not specify any risk variables, then risk statistics are computed for every variable in the model. In a similar fashion, if you specify a multinomial response variable as a risk variable, then a common population profile, L, is used to compare the probability between two different response levels; however, if you use multinomial-trial syntax, you cannot specify those response variables as risk variables. Standard errors and confidence limits are produced by using the delta method. You can specify at most one RISK statement.

By default, all continuous variables are set to their means and all classification variables are set to their reference levels. If your risk variable is continuous, then by default the risk statistics compare the base subpopulation where the risk variable is set to its mean value to a population where that variable is one unit larger. If your risk variable is a classification variable, then by default the risk statistics compare each level of the risk variable to its reference level. If your risk variable is a multinomial response variable, then the risk statistics compare each level of the response to its reference level. You can also specify variables that are used to construct SPLINE and POLYNOMIAL effects as risk variables.

If you specify the PRIOR= option, the risk computations use the posterior predicted probabilities instead of the individual predicted probabilities, and the gradients are computed by using a finite-difference method.

For more information about the risk computations, see the section "Risk Differences and Relative Risks" on page 1034.

You can specify the following *options* after a slash (/):

ALPHA=*number*
sets the significance level for producing confidence intervals. The ALPHA= value that you specify in the PROC LOGSELECT statement is the default. If neither ALPHA= option is specified, then by default ALPHA=0.05.

AT(*var***=***var-options* **<...** *var***=***var-options***>)**
enables you to specify fixed values or levels for a variable in the model. The following *var-options* are enabled when you specify the *var* variable:

value-list
specifies fixed values for the variable *var*. When *var* is a continuous variable, you can specify one or more numbers in the *value-list*. By default, continuous variables are set to their means. When *var* is a classification variable, you can specify one or more formatted levels of the variable enclosed in single quotes (for example, **A='cat' 'dog'**). By default, classification variables are set to their reference level. For example, if your model includes a classification variable A={cat,dog} and a continuous variable B, then specifying the following statements sets A to 'cat' and sets B to 7 and then 9:

```
risk X / AT(A='cat' B=7 9);
```

REF
specifies the reference level of *var* when *var* is a classification variable. This is the default for classification variables.

ALL
specifies every level of *var* when *var* is a classification variable.

DIFF=REF | ALL
specifies whether to compute the risk statistics for a classification risk variable against the reference level (DIFF=REF) or to compare all pairs of the risk variable's levels (DIFF=ALL). By default, DIFF=REF. The DIFF= option has no effect when the risk variable is continuous.

E
displays the L_1 and L_2 vectors; zero values are displayed as blanks. For cumulative response models, the intercept parameters are displayed as blanks and the slope parameters for a given risk are displayed only once. For the generalized logit model, the slope parameters that are used for a single response function are displayed, and the same parameters are used for all the other response functions. If you have a multinomial response model and also specify the dependent variable as a risk variable, then a single L vector is displayed.

REVERSE
swaps the L_1 and L_2 vectors in the computations. The risk difference is computed as $p_2 - p_1$ instead of $p_1 - p_2$, and the risk ratio is computed as p_2/p_1 instead of p_1/p_2.

SHOWPROBS
displays the two probabilities that are used in the risk statistic computations. These two values are always included in columns named P1 and P2 when you specify the DISPLAYOUT statement to output the "Risks" table.

TYPE=DIFFERENCE | RELATIVE
requests risk differences or relative risks. By default, TYPE=DIFFERENCE.

UNIT(*variable1***=***list1* **<...** *variable2***=***list2* **>)**
specifies one or more numbers that represent the units of change for a continuous risk variable so that you can produce customized risk statistics. An estimate of the corresponding risk statistic is displayed for each specified unit of change. The UNIT option is ignored for classification *variables*. For example, the following statement requests a risk difference that represents the change in the risk when the variable *X* is increased by two units:

```
risk X / UNIT(X=2);
```

By default, the unit of change is 1.

STDERR
indicates that the units of change that you specify in the UNIT option represent multiples of the sample standard error of the risk variable, based on the observations that are used to fit the model. If you specify the STDERR option without the UNIT option, then the unit of change is 1 standard error.

SELECTION Statement

SELECTION < **METHOD=***method*< *(method-options)* > >< *options* > **;**

The SELECTION statement performs model selection by examining whether effects should be added to or removed from the model according to rules that are defined by model selection methods. The statement is fully documented in the section "SELECTION Statement" on page 42 in Chapter 2, "Shared Concepts."

The LOGSELECT procedure supports the following effect-selection *methods* in the SELECTION statement:

BACKWARD performs backward elimination. This method starts with all effects in the model and deletes effects.

BACKWARD(FAST) performs fast backward elimination. This method starts with all effects in the model and deletes effects without refitting the model.

ELASTICNET performs elastic net selection. This method selects effects by fitting a sequence of candidate models with a grid of regularization parameters. For more information, see the section "Elastic Net Selection" on page 83 in Chapter 2, "Shared Concepts."

FORWARD performs forward selection. This method starts with no effects in the model and adds effects.

LASSO performs model selection by the group LASSO method. This method adds and removes effects by using a sequence of LASSO steps. For more information, see the section "Group LASSO Selection" on page 87 in Chapter 2, "Shared Concepts."

NONE results in no model selection. This method fits the full model.

STEPWISE performs stepwise selection. This method is similar to the FORWARD method except that effects already in the model do not necessarily stay there.

By default, METHOD=STEPWISE.

You can specify the following criteria in the SELECT=, CHOOSE=, and STOP= *method-options*:

AIC uses Akaike's information criterion (Akaike 1974) computed on the training data.

AICC uses a small-sample bias-corrected version of Akaike's information criterion, as promoted in Hurvich and Tsai (1989) and Burnham and Anderson (1998), computed on the training data.

SBC | BIC uses the Schwarz Bayesian criterion (Schwarz 1978) computed on the training data.

SL uses the significance level of the score test computed on the training data as the criterion (not available for the CHOOSE= option).

VALIDATE uses the average square error (ASE) computed on the validation data as the criterion (not available for the SELECT= option).

For more information, see the section "Information Criteria" on page 1020.

For effect-selection methods other than LASSO, the SELECT=, CHOOSE=, and STOP= *method-options* default to SBC. These defaults differ from their corresponding defaults in the HPLOGISTIC procedure in SAS/STAT software.

If you specify METHOD=LASSO, then the value of the CHOOSE= option is SBC by default, you can specify CHOOSE=NONE to select the model in the last LASSO step, and the STOPHORIZON= option has no effect.

If you specify METHOD=ELASTICNET, then the HIERARCHY= and STOPHORIZON= options have no effect.

NOTE: If you use the fast backward elimination method, then the –2 log likelihood, AIC, AICC, and SBC statistics are approximated at each step where the model is not refit, and hence they do not match the values that are computed when that model is fit outside the selection routine. Similarly, if you specify SELECT=AIC, AICC, or SBC, the selection criteria are estimated (Lawless and Singhal 1978), and hence they do not match the values that are computed when that model is fit outside the selection routine.

NOTE: The default model hierarchy method is HIERARCHY=NONE for the stepwise, forward, and fast backward selection methods. The backward elimination method always uses the HIERARCHY=SINGLE *method-option*.

The LASSO method produces a summary table that displays the effects that are added or removed at each step; the LASSO regularization parameter; and the AIC, AICC, and SBC fit statistics. The elastic net method produces a similar summary table that includes the selected effects, the regularization parameter, and the value of the CHOOSE= criterion. For the other methods, you can specify the following values for the DETAILS= option:

SUMMARY produces a summary table that shows the effect that is added or removed at each step along with the SELECT=, CHOOSE=, and STOP= criteria. The summary table is produced by default if the DETAILS= option is not specified.

STEPS produces the preceding summary table and displays the results from fitting each model at each step.

ALL produces the preceding tables and a detailed listing of all candidates at each step along with their ranking in terms of the selection criterion for addition to or removal from the model.

If you specify the PLOTS=CRITERIA or PLOTS=ALL option, then a plot of the fit criterion by the selection step is created for the AIC, AICC, and SBC statistics. If you also specify a PARTITION statement or the PARTFIT option, then the same type of plot is created for the ASE (for each role), max-rescaled R-square, McFadden's R-square, misclassification rate, and difference of means statistics.

If you specify the PLOTS=FITBYROLE or PLOTS=ALL option and a PARTITION statement, then a plot of the ASE by the selection step for each role is created.

The PLOTS= option is not available for the LASSO method or the elastic net method.

Model selection is not available for repeated measures (GEE) models.

STORE Statement

STORE < **OUT=***libref.data-table* > < / **NOTE** | **TEXT=***quoted-string* < *quoted-string . . .* > > **;**

The STORE statement saves the context and results of the statistical analysis. The resulting analytic store has a binary file format that cannot be modified. An example of analytic store technology is to perform a time-consuming analysis and to store its results by using the STORE statement. Later you can perform specific statistical analysis tasks based on the saved results of the previous analysis, without having to fit the model again. You can process the contents of the analytic store by specifying the RESTORE= option, as well as by specifying the ASTORE procedure and the **`logisticAssociation`**, **`logisticCode`**, **`logisticOddsRatio`**, **`logisticScore`**, and **`logisticType3`** actions in the **`regression`** action set (see Chapter 22, "Regression Action Set" (*SAS Visual Statistics: Programming Guide*)).

You specify the name of the analytic store by using the OUT= option. You can save notes to the analytic store by specifying them as quoted strings in the NOTE= or TEXT= option. Each quoted string begins a new line, preserves initial blanks, and can be at most 200 characters long. These text strings are displayed when you specify the item store in a RESTORE= option.

libref.data-table is a two-level name, where *libref* refers to the library, and *data-table* specifies the name of the output data table. For more information about this two-level name, see the section "Using CAS Sessions and CAS Engine Librefs" on page 963.

WEIGHT Statement

WEIGHT *variable* < / **NORMALIZE** > **;**

The *variable* in the WEIGHT statement is used as a weight to perform a weighted analysis of the data. Observations that have nonpositive or missing weights are not included in the analysis. The values of the WEIGHT variable can be nonintegral and are not truncated. When you omit the WEIGHT statement, all observations that are used in the analysis are assigned a weight of 1.

You can specify the following option in the WEIGHT statement after a slash (/):

NORMALIZE

normalizes the weights that are specified by the WEIGHT *variable* so that they add up to the sample size. Weights are normalized by multiplying them by $\frac{\sum f_i t_i}{\sum w_i f_i t_i}$, where training observation i has a frequency equal to f_i and a weight equal to w_i, and t_i is the number of trials when you are using events/trials or multinomial-trial syntax; otherwise t_i is equal to 1. When you specify this option, the estimated covariance matrix of the parameter estimators is invariant to the scale of the WEIGHT *variable*.

CAUTION: PROC LOGSELECT does not compute the proper variance estimators if you are analyzing survey data and you use the WEIGHT statement to specify the sampling weights. Instead, use the SURVEYLOGISTIC procedure, which is designed to perform the necessary, and correct, computations.

Details: LOGSELECT Procedure

Missing Values

Any observation that has missing values for the response, frequency, weight, offset, or explanatory variables is excluded from the analysis; however, missing values are valid for response and explanatory variables that are specified along with the MISSING option in the CLASS statement. Observations that have a nonpositive weight or a frequency less than 1 are also excluded.

The estimated linear predictor and the fitted probabilities are not computed for any observation that has missing offset or explanatory variable values. However, if only the response value is missing, the linear predictor and the fitted probabilities can be computed and output to a data table by using the OUTPUT statement.

You can also model the missing values by specifying the INFORMATIVE option in the MODEL statement. For more information about informative missingness, see the section "Informative Missingness" on page 99 in Chapter 2, "Shared Concepts."

Response Distributions

The response distribution is the probability distribution of the response (target) variable. The LOGSELECT procedure can fit data for the following distributions:

- binary distribution
- binomial distribution
- multinomial distribution

The expressions for the log-likelihood functions of these distributions are given in the next section.

The binary (or Bernoulli) distribution is the elementary distribution of a discrete random variable that can take two values, which have the probabilities p and $1 - p$. Suppose the random variable is denoted as Y and

$$\Pr(Y = 0) = p$$
$$\Pr(Y = 1) = 1 - p$$

The value that is associated with probability p is often called the *event* or "success"; the complementary event is called the *nonevent* or "failure." A Bernoulli experiment is a random draw from a binary distribution and generates events with probability p. The following example fit a binary regression model to the response variable Y with regressors X1 and X2:

```
proc logselect;
   model Y = X1 X2;
run;
```

If $Y_1, \ldots, Y_n$ are n independent Bernoulli random variables, then their sum follows a binomial distribution. In other words, if $Y_i = 1$ denotes an event (success) in the ith Bernoulli trial, a binomial random variable is the number of events (successes) in n independent Bernoulli trials. If you use the events/trials syntax in the MODEL statement, the LOGSELECT procedure fits the model as if the data had arisen from a binomial distribution. For example, the following statements fit a binomial regression model with regressors X1 and X2. The variables E and T represent the events and trials for the binomial distribution.

```
proc logselect;
   model E/T = X1 X2;
run;
```

If the events/trials syntax is used, then both variables must be numeric and the value of the events variable cannot be less than 0 or exceed the value of the trials variable.

The multinomial distribution is a generalization of the binary distribution and allows for more than two outcome categories. Because there are more than two possible outcomes for the multinomial distribution, the terminology of "successes," "failures," "events," and "nonevents" no longer applies. With multinomial data, these outcomes are generically referred to as "categories" or levels.

If each observation represents a single trial, and the response is one of several categories, then when you specify your model with single-trial syntax, the LOGSELECT procedure determines that the response variable has more than two levels and fits the model as if the data had arisen from a multinomial distribution. For example, if the response variable Y takes values in {'a','b','c'}, then the following specification fits a multinomial logistic regression model:

```
proc logselect;
   model Y = X1 X2;
run;
```

Alternatively, if your subjects are grouped, and if the number of subjects in an observation with response Y='a', 'b', and 'c' are contained in the variables A, B, and C, respectively, then you can specify a multinomial logistic regression model by using multinomial-trial syntax as follows:

```
proc logselect;
   model A B C = X1 X2;
run;
```

By default, the procedure assumes that the response categories are ordered, and it fits a cumulative link model by applying the default or specified link function. For single-trial syntax, a cumulative link function compares the lower ordered categories to the higher ordered categories; for multinomial-trial syntax, a cumulative link function compares the first levels in the response-level list to the last levels.

If the response categories are unordered, then you should fit a generalized logit model by specifying the LINK=GLOGIT option in the MODEL statement. For single-trial syntax, the generalized logit link uses the last ordered category as the reference level by default; for multinomial-trial syntax, the last level in the response-level list is the reference level.

Log-Likelihood Functions

The LOGSELECT procedure forms the log-likelihood functions of the various models as

$$L(\boldsymbol{\mu};\mathbf{y}) = \sum_{i=1}^{n} f_i \, l(\mu_i; y_i, w_i)$$

where μ_i is the probability that the ith observation is an event, $l(\mu_i; y_i, w_i)$ is the log-likelihood contribution of the ith observation, w_i is the weight, and f_i is its frequency. For the determination of w_i and f_i, see the WEIGHT and FREQ statements. The individual log-likelihood contributions for the various distributions are described in the following sections.

Binary Distribution

The LOGSELECT procedure computes the log-likelihood function $l(\mu_i(\boldsymbol{\beta}); y_i)$ for the ith binary observation as

$$\begin{aligned} \eta_i &= \mathbf{x}_i'\boldsymbol{\beta} \\ \mu_i(\boldsymbol{\beta}) &= g^{-1}(\eta_i) \\ l(\mu_i(\boldsymbol{\beta}); y_i) &= y_i \log\{\mu_i\} + (1 - y_i)\log\{1 - \mu_i\} \end{aligned}$$

where the variable y_i takes the value 1 for an event and 0 for a nonevent. The inverse link function $g^{-1}(\cdot)$ maps from the scale of the linear predictor η_i to the scale of the mean. For example, for the logit link (the default),

$$\mu_i(\boldsymbol{\beta}) = \frac{\exp\{\eta_i\}}{1 + \exp\{\eta_i\}}$$

You can control which binary outcome in your data is modeled as the event by using the *response-options* in the MODEL statement, and you can choose the link function by using the LINK= option in the MODEL statement.

If a WEIGHT statement is specified and w_i denotes the weight for the current observation, the log-likelihood function is computed as

$$l(\mu_i(\boldsymbol{\beta}); y_i, w_i) = w_i l(\mu_i(\boldsymbol{\beta}); y_i)$$

Binomial Distribution

The LOGSELECT procedure computes the log-likelihood function $l(\mu_i(\boldsymbol{\beta}); y_i)$ for the ith binomial observation as

$$\begin{aligned} \eta_i &= \mathbf{x}_i'\boldsymbol{\beta} \\ \mu_i(\boldsymbol{\beta}) &= g^{-1}(\eta_i) \\ l(\mu_i(\boldsymbol{\beta}); y_i, w_i) &= w_i \left(y_i \log\{\mu_i\} + (n_i - y_i)\log\{1-\mu_i\}\right) \\ &\quad + w_i \log \binom{n_i}{y_i} \end{aligned}$$

where y_i and n_i are the values of the events and trials of the ith observation, respectively. The value μ_i measures the probability of events (successes) in the underlying Bernoulli distribution whose aggregate follows the binomial distribution.

Multinomial Distribution

The multinomial distribution that the LOGSELECT procedure models is a generalization of the binary distribution; it is the distribution of a single draw from a discrete distribution that has J possible values. Thus, the log-likelihood function for the ith observation is

$$l(\boldsymbol{\mu}_i; \mathbf{y}_i, w_i) = w_i \sum_{j=1}^{J} y_{ij} \log\{\mu_{ij}\}$$

In this expression, J denotes the number of response categories (the number of possible outcomes), y_{ij} takes the value 1 if the jth response is observed and 0 otherwise, and μ_{ij} is the probability that the ith observation takes on the response value associated with category j.

If you use the multinomial-trial form of the syntax, then the LOGSELECT procedure includes the constant term in the computation of the log-likelihood function

$$l(\boldsymbol{\mu}_i; \mathbf{y}_i, w_i) = w_i \sum_{j=1}^{J} y_{ij} \log\{\mu_{ij}\} + w_i \log \binom{y_i}{y_{i1} \cdots y_{iJ}}$$

where y_{ij} is the number of trials in the ith observation that have the jth ordered response and $y_i = \sum_{j=1}^{J} y_{ij}$ is the total number of trials in that observation.

The category probabilities must satisfy

$$\sum_{j=1}^{J} \mu_{ij} = 1$$

and the constraint is satisfied by modeling $J - 1$ categories. In models that have ordered response categories, the probabilities are expressed in cumulative form, so the last category is redundant. In generalized logit models (multinomial models that have unordered categories), one category is chosen as the reference category, and the linear predictor in the reference category is set to 0. For more information, see the REF=*response-option* in the MODEL statement.

Existence of Maximum Likelihood Estimates

The likelihood equation for a logistic regression model does not always have a finite solution. Sometimes there is a nonunique maximum on the boundary of the parameter space, at infinity. The existence, finiteness, and uniqueness of maximum likelihood estimates for the logistic regression model depend on the patterns of data points in the observation space (Albert and Anderson 1984; Santner and Duffy 1986).

Consider a binary response model. Let Y_j be the response of the jth subject, and let $\mathbf{x}_j$ be the vector of explanatory variables (including the constant 1 that is associated with the intercept). There are three mutually exclusive and exhaustive types of data configurations:

Complete separation There is a complete separation of data points if there exists a vector $\mathbf{b}$ that correctly allocates all observations to their response groups; that is,

$$\begin{cases} \mathbf{b}'\mathbf{x}_j > 0 & Y_j = 0 \\ \mathbf{b}'\mathbf{x}_j < 0 & Y_j = 1 \end{cases}$$

This configuration produces nonunique infinite estimates. If the iterative process of maximizing the likelihood function is allowed to continue, then the log likelihood diminishes to 0 and the dispersion matrix becomes unbounded.

Quasi-complete separation The data are not completely separable, but there is a vector $\mathbf{b}$ such that

$$\begin{cases} \mathbf{b}'\mathbf{x}_j \geq 0 & Y_j = 0 \\ \mathbf{b}'\mathbf{x}_j \leq 0 & Y_j = 1 \end{cases}$$

and equality holds for at least one subject in each response group. This configuration also yields nonunique infinite estimates. If the iterative process of maximizing the likelihood function is allowed to continue, then the dispersion matrix becomes unbounded and the log likelihood diminishes to a nonzero constant.

Overlap If neither complete nor quasi-complete separation exists in the sample points, there is an overlap of sample points. In this configuration, the maximum likelihood estimates exist and are unique.

The LOGSELECT procedure uses a simple empirical approach to recognize the data configurations that lead to infinite parameter estimates. The basis of this approach is that any convergence method of maximizing the log likelihood must yield a solution that indicates complete separation, if such a solution exists. Upon convergence, if the predicted response equals the observed response for every observation, there is a complete separation of data points.

If the data are not completely separated, if an observation is identified to have an extremely large probability (≥ 0.95) of predicting the observed response, and if there have been at least eight iterations, then there are two possible situations. First, there is overlap in the data set, the observation is an atypical observation of its own group, and the iterative process stopped when a maximum was reached. Second, there is quasi-complete separation in the data set, and the asymptotic dispersion matrix is unbounded. If any of the diagonal elements of the dispersion matrix for the standardized observation vector (all explanatory variables standardized to zero mean and unit variance) exceeds 5,000, then PROC LOGSELECT declares quasi-complete separation; if any of the diagonal elements exceeds 1,000, then the procedure displays a message indicating that quasi-complete separation might be detectable by increasing the number of iterations. If either complete separation or quasi-complete separation is detected, a note is displayed in the procedure output.

Checking for quasi-complete separation is less foolproof than checking for complete separation. If neither type of separation is discovered and your parameter estimates have large standard errors, then your data might be separable. The NOCHECK option in the MODEL statement turns off the process of checking for infinite parameter estimates; the MINITER= option in the PROC LOGSELECT statement increases the number of iterations.

The LASSO Method of Model Selection

The LOGSELECT procedure implements the group LASSO method, which is described in the section "Group LASSO Selection" on page 87 in Chapter 2, "Shared Concepts." The current section provides some background about the LASSO method that you need in order to understand the group LASSO method.

LASSO (least absolute shrinkage and selection operator) selection arises from a constrained form of ordinary least squares regression in which the sum of the absolute values of the regression coefficients is constrained to be smaller than a specified parameter. More precisely, let $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m)$ denote the matrix of covariates, and let $\mathbf{y}$ denote the response. Then for a given parameter t, the LASSO regression coefficients $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_m)$ are the solution to the following constrained least squares problem:

$$\min ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||^2 \qquad \text{subject to} \qquad \sum_{j=1}^{m} |\beta_j| \leq t$$

For generalized linear models, the LASSO regression coefficients $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_m)$ are the solution to the following constrained optimization problem,

$$\min\{-L(\boldsymbol{\mu}; \mathbf{y})\} \qquad \text{subject to} \qquad \sum_{j=1}^{m} |\beta_j| \leq t$$

where L is the log-likelihood function defined in the section "Log-Likelihood Functions" on page 1011.

Provided that the LASSO parameter t is small enough, some of the regression coefficients will be exactly zero. Hence, you can think of the LASSO method as selecting a subset of the regression coefficients for each LASSO parameter. By increasing the LASSO parameter in discrete steps, you obtain a sequence of regression coefficients for which the nonzero coefficients at each step correspond to selected parameters. For more information about the LASSO method, see, for example, Hastie, Tibshirani, and Friedman (2009).

Generalized Estimating Equations

The analysis of correlated data that arise from repeated measurements when the measurements are assumed to be multivariate normal has been studied extensively. However, the normality assumption might not always be reasonable; for example, you must use a different methodology in the data analysis when the responses are discrete and correlated. Generalized estimating equations (GEEs) provide a practical method with reasonable statistical efficiency to analyze such data.

Liang and Zeger (1986) introduced GEEs as a method of dealing with correlated data when, except for the correlation among responses, the data can be modeled as a generalized linear model. For example, in many cases you can model correlated binary and count data in this way. For more information about GEEs, see Hardin and Hilbe (2003), Diggle, Liang, and Zeger (1994), and Lipsitz et al. (1994).

Let y_{ij}, $j = 1, \ldots, n_i$, $i = 1, \ldots, K$, represent the jth measurement on the ith subject. There are n_i measurements on subject i and $\sum_{i=1}^{K} f_i\, n_i$ total measurements, where f_i is the cluster frequency that you specify in the FREQ statement. If you omit the FREQ statement, then $f_i = 1$ for all observations. The frequencies must be the same for all observations within each cluster.

You model correlated data by using the same link function and linear predictor specification that you use to model independent data. You use the same variance functions as in the independent case, but you must also specify a structure for the correlation between the repeated measurements. In the following material, $v(\mu)$ denotes the variance function for the specified distribution.

Let the vector of measurements on the ith subject be $\mathbf{Y}_i = [y_{i1}, \ldots, y_{in_i}]'$, where the corresponding vector of means is $\boldsymbol{\mu}_i = [\mu_{i1}, \ldots, \mu_{in_i}]'$, and let $\mathbf{V}_i$ be the covariance matrix of $\mathbf{Y}_i$. Let the vector of independent, or explanatory, variables for the jth measurement on the ith subject be $\mathbf{x}_{ij} = [x_{ij1}, \ldots, x_{ijp}]'$. The generalized estimating equation of Liang and Zeger (1986) for estimating the $p \times 1$ vector of regression parameters $\boldsymbol{\beta}$ is an extension of the independence estimating equation to correlated data and is given by

$$\mathbf{S}(\boldsymbol{\beta}) = \sum_{i=1}^{K} f_i \mathbf{D}_i' \mathbf{V}_i^{-1} (\mathbf{Y}_i - \boldsymbol{\mu}_i(\boldsymbol{\beta})) = \mathbf{0}$$

where

$$\mathbf{D}_i = \frac{\partial \boldsymbol{\mu}_i}{\partial \boldsymbol{\beta}}$$

The relationship between the linear predictor and the mean is specified by

$$g(\mu_{ij}) = \mathbf{x}_{ij}{}'\boldsymbol{\beta}$$

where g is the link function, and so the $p \times n_i$ matrix of partial derivatives of the mean with respect to the regression parameters for the ith subject is given by

$$\mathbf{D}_i' = \frac{\partial \boldsymbol{\mu}_i'}{\partial \boldsymbol{\beta}} = \begin{bmatrix} \dfrac{x_{i11}}{g'(\mu_{i1})} & \cdots & \dfrac{x_{in_i1}}{g'(\mu_{in_i})} \\ \vdots & & \vdots \\ \dfrac{x_{i1p}}{g'(\mu_{i1})} & \cdots & \dfrac{x_{in_ip}}{g'(\mu_{in_i})} \end{bmatrix}$$

Working Correlation Matrix

Let $\mathbf{R}_i(\boldsymbol{\alpha})$ be an $n_i \times n_i$ "working" correlation matrix that is fully specified by the vector of parameters $\boldsymbol{\alpha}$. The covariance matrix of $\mathbf{Y}_i$ is modeled as

$$\mathbf{V}_i = \mathbf{A}_i^{\frac{1}{2}} \mathbf{W}_i^{-\frac{1}{2}} \mathbf{R}(\boldsymbol{\alpha}) \mathbf{W}_i^{-\frac{1}{2}} \mathbf{A}_i^{\frac{1}{2}}$$

where $\mathbf{A}_i$ is an $n_i \times n_i$ diagonal matrix whose jth diagonal element is $v(\mu_{ij})$, and $\mathbf{W}_i$ is an $n_i \times n_i$ diagonal matrix whose jth diagonal is w_{ij}, where w_{ij} is a weight specified in the WEIGHT statement. If there is no WEIGHT statement, then $w_{ij} = 1$ for all i and j. If $\mathbf{R}_i(\boldsymbol{\alpha})$ is the true correlation matrix of $\mathbf{Y}_i$, then $\mathbf{V}_i$ is the true covariance matrix of $\mathbf{Y}_i$.

The working correlation matrix is usually unknown and must be estimated. You estimate it in the iterative fitting process by using the current value of the parameter vector $\boldsymbol{\beta}$ to compute appropriate functions of the Pearson residual

$$e_{ij} = \frac{y_{ij} - \mu_{ij}}{\sqrt{v(\mu_{ij})/w_{ij}}}$$

If you specify the working correlation as $\mathbf{R}_0 = \mathbf{I}$, which is the identity matrix, the GEE reduces to the independence estimating equation.

Table 18.14 shows the working correlation types that the LOGSELECT procedure supports and the estimators that are used to estimate the working correlations.

Table 18.14 Working Correlation Structures and Correlation Parameter Estimators

Working Correlation Structure	Estimator
Autoregressive	
AR(1)	
$\mathrm{Corr}(Y_{ij}, Y_{i,j+t}) = \alpha^t$ for $t = 0, 1, 2, \ldots, n_i - j$	$\hat{\alpha} = \frac{1}{(K_1 - p)} \sum_{i=1}^{K} f_i \sum_{j \le n_i - 1} e_{ij} e_{i,j+1}$ $K_1 = \sum_{i=1}^{K} f_i (n_i - 1)$
Exchangeable	
$\mathrm{Corr}(Y_{ij}, Y_{ik}) = \begin{cases} 1 & j = k \\ \alpha & j \neq k \end{cases}$	$\hat{\alpha} = \frac{1}{(N^* - p)} \sum_{i=1}^{K} f_i \sum_{j<k} e_{ij} e_{ik}$ $N^* = 0.5 \sum_{i=1}^{K} f_i n_i (n_i - 1)$
Independence	
$\mathrm{Corr}(Y_{ij}, Y_{ik}) = \begin{cases} 1 & j = k \\ 0 & j \neq k \end{cases}$	The working correlation is not estimated in this case.
***m*-dependent**	
$\mathrm{Corr}(Y_{ij}, Y_{i,j+t}) = \begin{cases} 1 & t = 0 \\ \alpha_t & t = 1, 2, \ldots, m \\ 0 & t > m \end{cases}$	$\hat{\alpha}_t = \frac{1}{(K_t - p)} \sum_{i=1}^{K} f_i \sum_{j \le n_i - t} e_{ij} e_{i,j+t}$ $K_t = \sum_{i=1}^{K} f_i (n_i - t)$

Table 18.14 *continued*

Working Correlation Structure	Estimator
Unstructured	
$\mathrm{Corr}(Y_{ij}, Y_{ik}) = \begin{cases} 1 & j = k \\ \alpha_{jk} & j \neq k \end{cases}$	$\hat{\alpha}_{jk} = \frac{1}{(K-p)} \sum_{i=1}^{K} f_i e_{ij} e_{ik}$

Fitting Algorithm

The following algorithm fits the specified model by using GEEs. Note that this is not in general a likelihood-based method of estimation; inferences based on likelihoods are not possible for GEE methods.

1. Compute an initial estimate of $\boldsymbol{\beta}$ by using an ordinary generalized linear model, assuming independence.
2. Compute the working correlations $\mathbf{R}$ on the basis of the standardized residuals, the current $\boldsymbol{\beta}$, and the assumed structure of $\mathbf{R}$.
3. Compute an estimate of the covariance:

$$\mathbf{V}_i = \mathbf{A}_i^{\frac{1}{2}} \mathbf{W}_i^{-\frac{1}{2}} \hat{\mathbf{R}}(\boldsymbol{\alpha}) \mathbf{W}_i^{-\frac{1}{2}} \mathbf{A}_i^{\frac{1}{2}}$$

4. Update $\boldsymbol{\beta}$:

$$\boldsymbol{\beta}_{r+1} = \boldsymbol{\beta}_r + \left[\sum_{i=1}^{K} f_i \frac{\partial \boldsymbol{\mu}_i}{\partial \boldsymbol{\beta}}' \mathbf{V}_i^{-1} \frac{\partial \boldsymbol{\mu}_i}{\partial \boldsymbol{\beta}} \right]^{-1} \left[\sum_{i=1}^{K} f_i \frac{\partial \boldsymbol{\mu}_i}{\partial \boldsymbol{\beta}}' \mathbf{V}_i^{-1} (\mathbf{Y}_i - \boldsymbol{\mu}_i) \right]$$

5. Repeat steps 2–4 until convergence.

Missing Data

See Diggle, Liang, and Zeger (1994, Chapter 11) for a discussion of missing values in longitudinal data. Suppose you intend to take measurements $Y_{i1}, \ldots, Y_{in}$ for the ith unit. Missing values for which Y_{ij} are missing whenever Y_{ik} is missing for all $j \geq k$ are called *dropouts*. Otherwise, missing values that are intermixed with nonmissing values are called *intermittent* missing values. PROC LOGSELECT can estimate the working correlation from data that contain both types of missing values by using the *all available pairs* method, in which all nonmissing pairs of data are used in the moment estimators of the working correlation parameters that are defined previously. The resulting covariances and standard errors are valid under the missing completely at random (MCAR) assumption.

For example, for the unstructured working correlation model,

$$\hat{\alpha}_{jk} = \frac{1}{(K' - p)} \sum f_i e_{ij} e_{ik}$$

where the sum is over the units that have nonmissing measurements at times j and k, and K' is the number of units that have nonmissing measurements at j and k. Estimates of the parameters for other working

correlation types are computed in a similar manner, using available nonmissing pairs in the appropriate moment estimators.

The contribution of the *i*th unit to the parameter update equation is computed by omitting the elements of $(\mathbf{Y}_i - \boldsymbol{\mu}_\mathbf{i})$, the columns of $\mathbf{D}_i' = \frac{\partial \boldsymbol{\mu}_i}{\partial \boldsymbol{\beta}}'$, and the rows and columns of $\mathbf{V}_i$ that correspond to missing measurements.

Parameter Estimate Covariances

The *model-based* estimator of $\mathrm{Cov}(\hat{\boldsymbol{\beta}})$ is given by

$$\boldsymbol{\Sigma}_m(\hat{\boldsymbol{\beta}}) = \mathbf{I}_0^{-1}$$

where

$$\mathbf{I}_0 = \sum_{i=1}^{K} f_i \frac{\partial \boldsymbol{\mu}_i}{\partial \boldsymbol{\beta}}' \mathbf{V}_i^{-1} \frac{\partial \boldsymbol{\mu}_i}{\partial \boldsymbol{\beta}}$$

This is a consistent estimator of the covariance matrix of $\hat{\boldsymbol{\beta}}$ if the mean model and the working correlation matrix are correctly specified.

The estimator

$$\boldsymbol{\Sigma}_e = \mathbf{I}_0^{-1}\mathbf{I}_1\mathbf{I}_0^{-1}$$

is called the *empirical*, or *robust*, estimator of the covariance matrix of $\hat{\boldsymbol{\beta}}$, where

$$\mathbf{I}_1 = \sum_{i=1}^{K} f_i \frac{\partial \boldsymbol{\mu}_i}{\partial \boldsymbol{\beta}}' \mathbf{V}_i^{-1} \mathrm{Cov}(\mathbf{Y}_i) \mathbf{V}_i^{-1} \frac{\partial \boldsymbol{\mu}_i}{\partial \boldsymbol{\beta}}$$

It has the property of being a consistent estimator of the covariance matrix of $\hat{\boldsymbol{\beta}}$, even if the working correlation matrix is misspecified—that is, if $\mathrm{Cov}(\mathbf{Y}_i) \neq \mathbf{V}_i$. For further information about the robust variance estimate, see Zeger, Liang, and Albert (1988), Royall (1986), and White (1982). In computing $\boldsymbol{\Sigma}_e$, the parameters $\boldsymbol{\beta}$ and ϕ are replaced by estimates, and $\mathrm{Cov}(\mathbf{Y}_i)$ is replaced by the estimate

$$(\mathbf{Y}_i - \boldsymbol{\mu}_i(\hat{\boldsymbol{\beta}}))(\mathbf{Y}_i - \boldsymbol{\mu}_i(\hat{\boldsymbol{\beta}}))'$$

Quasi-likelihood Information Criterion

The quasi-likelihood information criterion (QIC) was developed by Pan (2001) as a modification of the Akaike information criterion (AIC) to apply to models that are fit by GEEs.

Define the quasi-likelihood under the independence working correlation assumption, which is evaluated using the parameter estimates under the working correlation *R* as

$$Q(\hat{\boldsymbol{\beta}}(R)) = \sum_{i=1}^{K} f_i \sum_{j=1}^{n_i} Q(\hat{\boldsymbol{\beta}}(R); (Y_{ij}, \mathbf{X}_{ij}))$$

where the quasi-likelihood contribution of the *j*th observation in the *i*th cluster is defined in the section "Quasi-likelihood Functions" on page 1019 and $\hat{\boldsymbol{\beta}}(R)$ are the parameter estimates obtained from GEEs that use the working correlation *R*.

QIC is defined as

$$\mathrm{QIC}(R) = -2Q(\hat{\boldsymbol{\beta}}(R)) + 2\mathrm{trace}(\hat{\Omega}_I \hat{V}_R)$$

where $\hat{V}_R$ is the robust covariance estimate and $\hat{\Omega}_I$ is the inverse of the model-based covariance estimate under the independence working correlation assumption, evaluated at $\hat{\boldsymbol{\beta}}(R)$, which are the parameter estimates obtained from GEEs that use the working correlation R.

PROC LOGSELECT also computes an approximation to $\mathrm{QIC}(R)$, which Pan (2001) defines as

$$\mathrm{QIC}_u(R) = -2Q(\hat{\boldsymbol{\beta}}(R)) + 2p$$

where p is the number of regression parameters.

Pan (2001) notes that QIC is appropriate for selecting regression models and working correlations, whereas QIC_u is appropriate only for selecting regression models.

Quasi-likelihood Functions

For detailed discussions of quasi-likelihood functions, see McCullagh and Nelder (1989) and Hardin and Hilbe (2003). The contribution of observation j in cluster i to the quasi-likelihood function that is evaluated at the regression parameters $\boldsymbol{\beta}$ is given by $Q(\boldsymbol{\beta}; (Y_{ij}, \mathbf{X}_{ij})) = Q_{ij}$, where

$$Q_{ij} = w_{ij}[r_{ij}\log(p_{ij}) + (n_{ij} - r_{ij})\log(1 - p_{ij})]$$

These are used in the computation of the quasi-likelihood information criteria (QIC) for the goodness of fit of models that are fit using GEEs. The w_{ij} are prior weights, if any, that are specified in the WEIGHT or FREQ statement.

GEE analysis is not supported when you perform model selection or when you perform a partitioned analysis.

Using a Preexisting Table Order

You can add group and order information to a table by using the **partition** action in the **table** action set. If you want to use the group and order information in a table to achieve a reproducible processing order and you also want to do a repeated measures analysis, the group information must follow specific rules. In particular, the variables that you specify in the **groupBy** parameter of the **partition** action must match the variables that you specify in the SUBJECT option of the REPEATED statement.

There are some variations to this requirement, depending on which type of subject effect you specify and which other options you use. If you specify a single-variable subject effect or an interaction subject effect, you must use all these variables in the **groupBy** parameter of the **partition** action. If you specify a nested subject effect, you must include the nested variable in the **groupBy** parameter. You can also include the nesting variables in the **groupBy** parameter, but you cannot include any other variables in the **groupBy** parameter. However, if you specify the MULTIPASS option in the PROC LOGSELECT statement, you must include the nested variable and all nesting variables in the **groupBy** parameter. You cannot include any other variables in the **groupBy** parameter.

If you use an order-sensitive working correlation structure, you do not need to use the variables that form the within-subject effect in the **orderBy** parameter of the **partition** action. These sets of variables can overlap, but it is not required that the preexisting table order correspond to the ordering for the within-subject effect.

Partition Fit Statistics

Specifying a PARTITION statement modifies the display of many tables by adding separate rows or columns for the training, validation, and test data tables and adds statistics to the "Fit Statistics" table. These additional statistics are useful for assessing the model and should be very similar for the different roles when the training data are representative of the other data partitions: average square error, misclassification rate, and the difference of means. The R^2, maximum-rescaled R^2, and McFadden's R^2 are also computed. For more information, see the section "Model Fit and Assessment Statistics" on page 1020.

Model Fit and Assessment Statistics

The statistics that are defined in this section are useful for assessing the fit of the model to your data; they are displayed in the "Fit Statistics" table. The statistics are computed for each data role when you specify a PARTITION statement.

Information Criteria

The calculation of the information criteria uses the following formulas, where p denotes the number of effective parameters in the candidate model, F denotes the sum of frequencies used, and l is the log likelihood evaluated at the converged estimates:

$$\begin{aligned}
\text{AIC} &= -2l + 2p \\
\text{AICC} &= \begin{cases} -2l + 2pF/(F-p-1) & \text{when } F > p+2 \\ -2l + 2p(p+2) & \text{otherwise} \end{cases} \\
\text{SBC} &= -2l + p\log(F)
\end{aligned}$$

If you do not specify a FREQ statement, F equals n, the number of observations used.

Generalized Coefficient of Determination

The goal of a coefficient of determination, also known as an R-square measure, is to express the agreement between a stipulated model and the data in terms of variation in the data that is explained by the model. In linear models, the R-square measure is based on residual sums of squares; because these are additive, a measure bounded between 0 and 1 is easily derived.

In more general models where parameters are estimated by the maximum likelihood principle, Cox and Snell (1989, pp. 208–209) and Magee (1990) proposed the following generalization of the coefficient of determination:

$$R^2 = 1 - \left\{ \frac{L(\mathbf{0})}{L(\widehat{\boldsymbol{\beta}})} \right\}^{\frac{2}{n}}$$

Here, $L(\mathbf{0})$ is the likelihood of the intercept-only model, $L(\widehat{\boldsymbol{\beta}})$ is the likelihood of the specified model, and n denotes the number of observations used in the analysis. This number is adjusted for frequencies if a FREQ statement is present, and it is based on the trials variable for binomial models.

As discussed in Nagelkerke (1991), this generalized R-square measure has properties similar to those of the coefficient of determination in linear models. If the model effects do not contribute to the analysis, $L(\widehat{\boldsymbol{\beta}})$ approaches $L(\mathbf{0})$ and R^2 approaches zero.

However, R^2 does not have an upper limit of 1. Nagelkerke suggested a rescaled generalized coefficient of determination, R^2_N, which achieves an upper limit of 1 by dividing R^2 by its maximum value:

$$R^2_{\max} = 1 - \{L(\mathbf{0})\}^{\frac{2}{n}}$$
$$R^2_N = \frac{R^2}{R^2_{\max}}$$

Another measure, from McFadden (1974), is also bounded by 0 and 1:

$$R^2_M = 1 - \left(\frac{\log L(\widehat{\boldsymbol{\beta}})}{\log L(\mathbf{0})}\right)$$

These measures are most useful for comparing competing models that are not necessarily nested—that is, models that cannot be reduced to one another by simple constraints on the parameter space. Larger values of the measures indicate better models.

Average Square Error

The average square error (ASE) is the average of the squared differences between the responses and the predictions. When you have a discrete number of response levels, the ASE is modified as shown in Table 18.15 (Brier 1950; Murphy 1973); it is also called the Brier score or Brier reliability.

Table 18.15 Average Square Error Computations

Response Type	ASE (Brier Score)
Binary	$\frac{1}{F}\sum_i f_i(y_i(1-\widehat{\pi}_i)^2 + (1-y_i)\widehat{\pi}_i^2)$
Binomial	$\frac{1}{F}\sum_i f_i(r_i/t_i - \widehat{\pi}_i)^2$
Multinomial	$\frac{1}{F}\sum_i f_i \sum_j (y_{ij}/y_i - \widehat{\pi}_{ij})^2$

In Table 18.15, $F = \sum_i f_i$, r_i is the number of events, t_i is the number of trials in binomial response models, $y_i = 1$ for events and 0 for nonevents in binary response models, and $\widehat{\pi}_i$ is the predicted probability of an event. For polytomous response models specified with single-trial syntax, $y_{ij} = 1$ if the ith observation has response level j, $y_i = 1$, and π_{ij} is the model-predicted probability of response level j for observation i. However, if you specify multinomial-trial syntax, y_{ij} is the number of trials in the ith observation that have response j, and the total number of trials is $y_i = \sum_{j=1}^{J} y_{ij}$.

Difference of Means

For a binary response model, write the mean of the model-predicted probabilities of event (Y=0) observations as $\overline{X}_1 = \frac{\sum_{i=1}^{n}(\widehat{\pi}_i | y_i=0)}{n_0}$ and of nonevent (Y=1) observations as $\overline{X}_2 = \frac{\sum_{i=1}^{n}(\widehat{\pi}_i | y_i=1)}{n_0}$, where $\widehat{\pi}_i$ is the predicted probability of an event. The difference of means is $\overline{X}_1 - \overline{X}_2$, which Tjur (2009) relates to other R-square measures and calls the *coefficient of discrimination*, because it is a measure of the model's ability to distinguish between the event and nonevent distributions. The difference of means is also the d' or Δm statistic (with unit standard error) that is discussed in the signal detection literature (McNicol 2005), and it is also referred to as *Tjur's R-square*.

Classification Table and ROC Curves

For binary response data, the response Y is either an event or a nonevent; let the response Y take the value 1 for an event and 2 for a nonevent. From the fitted model, a predicted event probability $\hat{\pi}_i$ can be computed for each observation i. Define your decision rule as follows: if the predicted event probability equals or exceeds a cutpoint value $z \in [0, 1]$, the observation is classified as an event; otherwise, it is classified as a nonevent. Suppose n_1 of n individuals experience an event, such as a disease, and the remaining $n_2 = n - n_1$ individuals do not experience that event (are nonevents). The 2×2 classification (confusion, decision, error) matrix in Table 18.16 is obtained by cross-classifying the observed and predicted responses, where n_{ij} is the total number of observations that are observed to have $Y = i$ and are classified into j. In this table, let $Y = 1$ denote an observed event and $Y = 2$ denote a nonevent, and let $D = 1$ indicate that the observation is classified as an event and $D = 2$ denote that the observation is classified as a nonevent.

Table 18.16 Classification Matrix

	$D=1$ ($\hat{\pi} \geq z$)	$D=2$ ($\hat{\pi} < z$)	**Total**
$Y=1$ (event)	n_{11}	n_{12}	n_1
$Y=2$ (nonevent)	n_{21}	n_{22}	n_2

The cells of the classification matrix of Table 18.16 are as follows:

- n_{11} = the number of *true positives*, which is the number of event observations that are correctly classified as events
- n_{21} = the number of *false positives*, which is the number of nonevent observations that are incorrectly classified as events
- n_{12} = the number of *false negatives*, which is the number of event observations that are incorrectly classified as nonevents
- n_{22} = the number of *true negatives*, which is the number of nonevent observations that are correctly classified as nonevents
- n_1 = the total number of actual events
- n_2 = the total number of actual nonevents

The accuracy of the classification is measured by its ability to predict events and nonevents correctly. *Sensitivity* (true positive fraction, TPF, recall) is the proportion of event responses that are predicted to be events. *Specificity* (true negative fraction, 1–FPF) is the proportion of nonevent responses that are predicted to be nonevents.

You can also measure accuracy by how well the classification predicts the response. The *positive predictive value* (precision, PPV) is the proportion of observations classified as events that are correctly classified. The *negative predictive value* (NPV) is the proportion of observations classified as nonevents that are correctly classified. The *correct classification rate* (accuracy, PC) is the proportion of observations that are correctly classified, whereas the *misclassification rate* (error rate) is the proportion of observations that are incorrectly classified. The *lift* is the ratio of the proportion of correctly classified events to the proportion of observations classified as events.

The *prevalence*, $\Pr(Y = 1)$, is the prior probability of an event. If the prevalence is different from the observed empirical event probability in the training data, n_1/n, then applying Bayes' theorem shows that the PPV, NPV, accuracy, PC, misclassification rate, and lift equations depend on the prevalence (Fleiss, Levin, and Paik 2003). For a stratified sampling situation in which n_1 and n_2 are chosen a priori, n_1/n is not a desirable estimate of $\Pr(Y = 1)$. You can specify the prevalence by using the PRIOR= option in the MODEL statement. If you specify a PARTITION statement without specifying the PRIOR= option, then the observed empirical probabilities for the training data are used as prevalences for computations of the validation and test statistics.

PROC LOGSELECT constructs the data for a receiver operating characteristic (ROC) curve by initially rounding the predicted probabilities to the nearest multiple of the value of the BINEPS= option. This effectively sorts the observations in increasing order of their estimated event probability. A classification matrix is created for each of these bins by using the rounded probability as the cutpoint. As the cutpoint moves from 0 to 1, those cutpoints for which the classification matrix changes are selected.

Alternatively, if you do not want to generate the entire ROC curve, you can use the CUTPT= option to specify your own list of cutpoints. In this case, the specified cutpoints are used to generate the classification matrices without rounding the predicted probabilities.

You can specify the CTABLE option to produce a classification table that includes these cutpoints and, for each cutpoint, any of the statistics in Table 18.17 that you request.

Table 18.17 Statistics from the Classification Matrix with Cutpoint z

Statistic	Equation	Column Name
Cutpoint	z	ProbLevel
Number of true positives	n_{11}	TruePos
Number of true negatives	n_{22}	TrueNeg
Number of false positives	n_{21}	FalsePos
Number of false negatives	n_{12}	FalseNeg
True positive fraction (sensitivity)	n_{11}/n_1	TPF
False positive fraction (1–specificity)	n_{21}/n_2	FPF
True negative fraction	n_{22}/n_2	TNF
False negative fraction	n_{12}/n_1	FNF
Correct classification rate	$(n_{11} + n_{22})/n$	Accuracy
Percentage correct (PC)	100×Accuracy	PC
Misclassification rate	1 – Accuracy	Misclass
Positive predictive value	$n_{11}/(n_{11} + n_{21})$	PPV
Negative predictive value	$n_{22}/(n_{12} + n_{22})$	NPV
Lift	TPF $/((n_{11} + n_{21})/n)$	Lift

You can output this classification table by specifying the OUT= option, and you can display the ROC curve by using the SGPLOT procedure as shown in Example 18.2.

The area under the ROC curve (AUC), as determined by the trapezoidal rule, is given by the concordance index c, which is described in the section "Association Statistics" on page 1024.

For more information about the topics in this section, see Pepe (2003).

Association Statistics

If you specify the ASSOCIATION option in the PROC LOGSELECT statement, PROC LOGSELECT displays measures of association between predicted probabilities and observed responses for binary, binomial, and ordinal response models. These measures assess the predictive ability of a model.

For ordinal response data, let the predicted mean score of an observation be the sum of the OrderedValue values (shown in the "Response Profile" table) minus one, weighted by the corresponding predicted probabilities for that observation; that is, the predicted means score $= \sum_{i=1}^{k+1}(i-1)\widehat{\pi}_i$, where $k+1$ is the number of response levels and $\widehat{\pi}_i$ is the predicted probability of the ith (ordered) response.

For binary and binomial responses, let the predicted mean score be the predicted event probability.

The predicted mean score is rounded to the nearest multiple of the value that you specify in the BINEPS= option. This effectively sorts the observations in increasing order of their predicted mean score. Of the n pairs of observations in the data that have different responses, let n_c be the number of pairs where the observation that has the lower-ordered response value has a lower predicted mean score, let n_d be the number of pairs where the observation that has the lower-ordered response value has a higher predicted mean score, and let $n_t = n - n_c - n_d$ be the rest of the observations. Let N be the sum of observation frequencies in the data. Then the following statistics are reported:

concordance index c (AUC)	=	$(n_c + 0.5n_t)/n$
Somers' D (Gini coefficient)	=	$(n_c - n_d)/n$
Goodman-Kruskal gamma	=	$(n_c - n_d)/(n_c + n_d)$
Kendall's tau-a	=	$(n_c - n_d)/(0.5N(N-1))$

If there are no ties, then Somers' D (Gini's coefficient) = $2c - 1$. For binary responses, the concordance index, c, is an estimate of the AUC, which is the area under the ROC curve.

If you specify a PARTITION statement, then PROC LOGSELECT displays a column for each of the roles.

Confidence Intervals for Parameters

There are two methods of computing confidence intervals for the regression parameters. One method is based on the profile-likelihood function, and the other method is based on the asymptotic normality of the parameter estimators. The latter method is not as time-consuming as the former method, because it does not involve an iterative scheme; however, it is not thought to be as accurate as the former method, especially for a small sample size.

Likelihood-Ratio-Based Confidence Intervals

The likelihood-ratio-based confidence interval is also known as the profile-likelihood confidence interval. The construction of this interval is derived from the asymptotic χ^2 distribution of the generalized likelihood ratio test (Venzon and Moolgavkar 1988).

Suppose that the parameter vector is $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_s)'$, and you want to compute a confidence interval for β_j. The profile-likelihood function for $\beta_j = \gamma$ is defined as

$$l_j^*(\gamma) = \max_{\boldsymbol{\beta} \in \mathcal{B}_j(\gamma)} l(\boldsymbol{\beta})$$

where $\mathcal{B}_j(\gamma)$ is the set of all $\boldsymbol{\beta}$ with the jth element fixed at γ, and $l(\boldsymbol{\beta})$ is the log-likelihood function for $\boldsymbol{\beta}$. If $l_{\max} = l(\widehat{\boldsymbol{\beta}})$ is the log likelihood evaluated at the maximum likelihood estimate $\widehat{\boldsymbol{\beta}}$, then $2(l_{\max} - l_j^*(\beta_j))$ has a limiting chi-square distribution with one degree of freedom if β_j is the true parameter value. Let $l_0 = l_{\max} - 0.5\chi_1^2(1-\alpha)$, where $\chi_1^2(1-\alpha)$ is the $100(1-\alpha)$th percentile of the chi-square distribution with one degree of freedom. A $100(1-\alpha)\%$ confidence interval for β_j is

$$\{\gamma : l_j^*(\gamma) \geq l_0\}$$

You find the endpoints of the confidence interval by solving numerically for values of β_j that satisfy equality in the preceding relation. To obtain an iterative algorithm for computing the confidence limits, you approximate the log-likelihood function in a neighborhood of $\boldsymbol{\beta}$ by using the quadratic function

$$\tilde{l}(\boldsymbol{\beta} + \boldsymbol{\delta}) = l(\boldsymbol{\beta}) + \boldsymbol{\delta}'\mathbf{g} + \frac{1}{2}\boldsymbol{\delta}'\mathbf{V}\boldsymbol{\delta}$$

where $\mathbf{g} = \mathbf{g}(\boldsymbol{\beta})$ is the gradient vector and $\mathbf{V} = \mathbf{V}(\boldsymbol{\beta})$ is the Hessian matrix. You obtain the increment $\boldsymbol{\delta}$ for the next iteration by solving the likelihood equations

$$\frac{d}{d\boldsymbol{\delta}}\{\tilde{l}(\boldsymbol{\beta} + \boldsymbol{\delta}) + \lambda(\mathbf{e}_j'\boldsymbol{\delta} - \gamma)\} = \mathbf{0}$$

where λ is the Lagrange multiplier, $\mathbf{e}_j$ is the jth unit vector, and γ is an unknown constant. The solution is

$$\boldsymbol{\delta} = -\mathbf{V}^{-1}(\mathbf{g} + \lambda\mathbf{e}_j)$$

By substituting this $\boldsymbol{\delta}$ into the equation $\tilde{l}(\boldsymbol{\beta} + \boldsymbol{\delta}) = l_0$, you can estimate λ as

$$\lambda = \pm\left(\frac{2(l_0 - l(\boldsymbol{\beta}) + \frac{1}{2}\mathbf{g}'\mathbf{V}^{-1}\mathbf{g})}{\mathbf{e}_j'\mathbf{V}^{-1}\mathbf{e}_j}\right)^{\frac{1}{2}}$$

You compute the upper confidence limit for β_j by starting at the maximum likelihood estimate of $\boldsymbol{\beta}$ and iterating with positive values of λ until you reach convergence. You repeat the process for the lower confidence limit by using negative values of λ.

You control convergence by the value ϵ that you specify in the PLCONV= option in the PROC LOGSELECT statement (the default value of ϵ is 1E–4). Convergence is declared if the following two conditions are satisfied:

$$|l(\boldsymbol{\beta}) - l_0| \leq \epsilon$$

and

$$(\mathbf{g} + \lambda\mathbf{e}_j)'\mathbf{V}^{-1}(\mathbf{g} + \lambda\mathbf{e}_j) \leq \epsilon$$

Wald Confidence Intervals

Wald confidence intervals are sometimes called the normal confidence intervals. They are based on the asymptotic normality of the parameter estimators. The $100(1-\alpha)\%$ Wald confidence interval for β_j is given by

$$\widehat{\beta}_j \pm z_{1-\alpha/2}\widehat{\sigma}_j$$

where z_p is the $100p$th percentile of the standard normal distribution, $\widehat{\beta}_j$ is the maximum likelihood estimate of β_j, and $\widehat{\sigma}_j$ is the standard error estimate of $\widehat{\beta}_j$.

Predicted Probabilities and Regression Diagnostics

For binary response data, you can produce observationwise predicted probabilities, confidence limits, and regression diagnostics developed by Pregibon (1981) by specifying the OUTPUT statement. For multinomial response data, you can likewise produce observationwise predicted probabilities, confidence limits, and raw residuals. If you use multinomial-trial syntax so that each observation represents an aggregate of trials, then extensions of Pregibon's diagnostics are also available (Lesaffre and Albert 1989; Williams 1987; Gupta, Nguyen, and Pardo 2008; Martín 2015).

For a binary response model, given a vector of covariates $\mathbf{x}_i$ for the ith observation in your data table and the model-predicted parameter estimates $\hat{\boldsymbol{\beta}}$, you can write the linear predictor $\hat{\eta}_i = \mathbf{x}_i'\hat{\boldsymbol{\beta}}$. The mean of the ith observation $\mu_i(\hat{\boldsymbol{\beta}})$, or the model-predicted event probability $\hat{\pi}_i$, is $\mu_i(\hat{\boldsymbol{\beta}}) = \hat{\pi}_i = g^{-1}(\eta_i)$, where you choose the link function g by specifying the LINK= option. The variance of the binary distribution is $V(\mu) = \mu(1-\mu) = \hat{\pi}_i(1-\hat{\pi}_i) = V(\pi)$, and $\boldsymbol{\Sigma}$ is the estimated covariance of $\hat{\boldsymbol{\beta}}$. Denote the frequency of the ith observation as f_i and the weight as w_i.

For ordinal response models, the predicted cumulative probabilities are computed in the same fashion by using the appropriate model-predicted intercept parameters $\hat{\alpha}_j$ and letting $\boldsymbol{\beta}$ consist of the slope parameters: $\hat{\eta}_{ij} = g(\Pr(Y \le j\,|\mathbf{x}_i) = \hat{\alpha}_j + \mathbf{x}_i'\hat{\boldsymbol{\beta}}$ and $\hat{\pi}_{ij} = \Pr(Y \le j\,|\mathbf{x}_i) = g^{-1}(\eta_{ij})$ for $1 \le j < J$.

For nominal response models, the predicted probabilities are computed by using the appropriate model-predicted intercept parameters $\hat{\alpha}_j$ and letting $\boldsymbol{\beta}_j$ consist of the slope parameters: $\hat{\eta}_{ij} = g(\Pr(Y = j\,|\mathbf{x}_i)) = \hat{\alpha}_j + \mathbf{x}_i'\hat{\boldsymbol{\beta}}_j$ and $\hat{\pi}_{ij} = \Pr(Y = j\,|\mathbf{x}_i) = g^{-1}(\eta_{ij})$ for $1 \le j < J$.

If you perform elastic net or LASSO selection, or if you specify the NOXPX or NOSTDERR option, then your analysis does not generate a covariance matrix, and you can produce only the predicted probabilities (individual, cumulative, and posterior when available) and the raw, Pearson, deviance, and working residuals.

Confidence Intervals

Approximate confidence intervals for predicted probabilities can be computed as follows. The variance of the linear predictor is estimated by

$$\hat{\sigma}^2(\eta_i) = \mathbf{x}_i'\boldsymbol{\Sigma}\mathbf{x}_i$$

For multinomial models, the variance also depends on the response function. Let $\boldsymbol{\delta}_j$ be a $(J–1)$ column vector whose jth entry is equal to 1 and all other entries are equal to 0. Redefine $\mathbf{x}_i = (\boldsymbol{\delta}_j', \mathbf{x}_i')'$, $\eta_i = \eta_{ij}$, and $\pi_i = \pi_{ij}$. Then

$$\hat{\sigma}^2(\eta_i) = \mathbf{x}_i'\boldsymbol{\Sigma}\mathbf{x}_i$$

The asymptotic $100(1-\alpha)\%$ confidence interval for η_i is

$$\hat{\eta}_i \pm z_{\alpha/2}\hat{\sigma}(\hat{\eta}_i)$$

where $z_{\alpha/2}$ is the $100(1-\alpha/2)$th percentile point of a standard normal distribution.

The predicted probability and the $100(1-\alpha)\%$ confidence limits for π_i are obtained by back-transforming the corresponding measures for the linear predictor. So the confidence limits are

$$g^{-1}\left[\hat{\eta}_i \pm z_{\alpha/2}\hat{\sigma}(\hat{\eta}_i)\right]$$

Hat-Matrix Diagonals

The diagonal elements of the hat matrix are useful in detecting extreme points in the design space, where they tend to have larger values. For the generalized linear model, the variance of the *i*th individual observation is

$$v_i = \frac{V(\pi_i)}{f_i w_i}$$

For the *i*th observation, let

$$w_{ei} = v_i^{-1}(g'(\pi_i))^{-2}$$

where $g'(\pi_i)$ is the derivative of the link function evaluated at π_i. The weight matrix $\mathbf{W}_e$ is a diagonal matrix, with w_{ei} denoting the *i*th diagonal element, which is used in computing the expected information matrix. Define the leverage, or hat-matrix diagonal, h_i, as the *i*th diagonal element of the matrix

$$\mathbf{W}_e^{\frac{1}{2}}\mathbf{X}(\mathbf{X}'\mathbf{W}_e\mathbf{X})^{-1}\mathbf{X}'\mathbf{W}_e^{\frac{1}{2}}$$

If the estimated probability is extreme (less than 0.1 and greater than 0.9, approximately), then the hat-matrix diagonal might be greatly reduced in value. Consequently, when an observation has a very large or very small estimated probability, its leverage is not a good indicator of the observation's distance from the design space (Hosmer and Lemeshow 2000, p. 171).

When you use multinomial-trial syntax to model aggregated multinomial response data and the PREDPROBS option is in effect, the relevant elements of the resulting hat matrix for observation i are a J–1×J–1 matrix $\mathbf{H}_i$. The following are three scalar representations of the leverage based on $\mathbf{H}_i$ and $\mathbf{M}_i = \mathbf{I}_{J-1} - \mathbf{H}_i$:

determinant:	$h_{Di} = \det(\mathbf{M_i})$
trace:	$h_{Ti} = \frac{1}{J-1}\mathrm{trace}(\mathbf{H}_i)$
potential:	$h_{Pi} = \frac{1}{J-1}\mathrm{trace}(\mathbf{M}_i^{-1}\mathbf{H}_i)$

Residuals

Residuals are useful in identifying observations that are not explained well by the model. For binary and binomial response data, the raw residuals are

$$r_i = y_i/t_i - \hat{\pi}_i$$

where y_i is the number of event responses out of t_i trials for the *i*th observation. For single-trial syntax, $t_i = 1$ and $y_i = 1$ if the ordered response is 1 and $y_i = 0$ otherwise.

For multinomial response data, the raw residuals are

$$r_{ij} = y_{ij}/y_i - \hat{\pi}_{ij}$$

where $\hat{\pi}_{ij}$ are the model-predicted probabilities of the ordered response j for observation i. With single-trial syntax, $y_{ij} = 1$ if the ith observation has ordered response j and $y_{ij} = 0$ otherwise, and $y_i = 1$. With multinomial-trial syntax, y_{ij} is the number of trials in the ith observation which have ordered response j, and $y_i = \sum_j y_{ij}$ is the total number of trials for the ith observation. If the PREDPROBS option is in effect and you use multinomial-trial syntax, the raw residuals are accumulated across the response levels as follows:

$$r_i = \sqrt{\sum_{j=1}^{J} y_{ij} r_{ij}^2}$$

Pearson residuals are the square root of the ith observation's contribution to Pearson's chi-square:

$$r_{Pi} = r_i \sqrt{\frac{w_i}{V(\pi_i)}}$$

where $V(\pi) = \hat{\pi}_i(1 - \hat{\pi}_i)/t_i$. If you use multinomial-trial syntax, the Pearson residuals for each response level j are

$$r_{Pij} = \frac{\sqrt{w_i}\, y_i\, r_{ij}}{\sqrt{y_i \hat{\pi}_{ij}}}$$

and if the PREDPROBS option is in effect, these values are accumulated across the response levels as

$$r_{Pi} = \sqrt{\sum_{j=1}^{J} r_{Pij}^2}$$

Deviance residuals are the square root of the contribution of the ith observation to the deviance, with the sign of the raw residual,

$$r_{Di} = \text{sign}(r_i)\sqrt{d_i}$$

where

$$d_i = 2w_i t_i \left[y_i/t_i \log\left(\frac{y_i/t_i}{\hat{\pi}_i}\right) + (1 - y_i/t_i) \log\left(\frac{1 - y_i/t_i}{1 - \hat{\pi}_i}\right)\right]$$

If you use multinomial-trial syntax, the deviance residuals for each response level j are

$$r_{Dij} = \text{sign}(r_{ij})\sqrt{2w_i y_{ij} \left|\log\left(\frac{y_{ij}}{y_i \hat{\pi}_{ij}}\right)\right|}$$

and if the PREDPROBS option is in effect, these values are accumulated across the response levels as

$$r_{Di} = \pm \sqrt{\left|\sum_{j=1}^{J} \text{sign}(r_{Dij}) r_{Dij}^2\right|}$$

where $\pm$ is the sign of the sum.

The working residuals are

$$r_{Wi} = r_i \left(\frac{\partial \pi_i}{\partial \eta_i} \right)^{-1}$$

Working residuals are not available for multinomial-response models when you specify multinomial-trial syntax or the PREDPROBS option.

The Pearson residuals, standardized to have unit asymptotic variance, are

$$r_{SPi} = \frac{r_{Pi}}{\sqrt{1 - h_i}}$$

If you use multinomial-trial syntax and the PREDPROBS option is in effect, the standardized Pearson residuals are

$$r_{SPi} = \sqrt{\mathbf{r}'_{Pi-J} \mathbf{M}_i^{-1} \mathbf{r}_{Pi-J}}$$

where $\mathbf{r}_{Pi-J} = (r_{Pi1}, \ldots, r_{Pi,J-1})'$.

The deviance residuals, standardized to have unit asymptotic variance, are

$$r_{SDi} = \frac{r_{Di}}{\sqrt{1 - h_i}}$$

If you use multinomial-trial syntax and the PREDPROBS option is in effect, the standardized deviance residuals are

$$r_{SDi} = \text{sign}(\mathbf{r}'_{Di-J} \mathbf{M}_i^{-1} \mathbf{r}_{Di-J}) \sqrt{|\mathbf{r}'_{Di-J} \mathbf{M}_i^{-1} \mathbf{r}_{Di-J}|}$$

where $\mathbf{r}_{Di-J} = (r_{Di1}, \ldots, r_{Di,J-1})'$.

The likelihood residuals, which estimate components of a likelihood ratio test of deleting an individual observation, are a weighted combination of the standardized Pearson and deviance residuals:

$$r_{Li} = \text{sign}(r_i) \sqrt{h_i r_{SPi}^2 + (1 - h_i) r_{SDi}^2}$$

If you use multinomial-trial syntax and the PREDPROBS option is in effect, the likelihood residuals are computed as

$$r_{Li} = \sqrt{\mathbf{r}'_{Pi-J} \mathbf{H}_i \mathbf{M}_i^{-1} \mathbf{r}_{Pi-J} + \mathbf{r}'_{Di-J} \mathbf{r}_{Di-J}} = \sqrt{\text{DIFDEV}_i}$$

Other Regression Diagnostics

The CBAR statistic is a confidence interval displacement diagnostic that provides a scalar measure of the influence of an individual observation on $\widehat{\boldsymbol{\beta}}$. This diagnostic is based on the same idea as the Cook distance in linear regression theory (Cook and Weisberg 1982), but it uses the one-step estimate:

$$\overline{C}_i = r_{Pi}^2 h_i / (1 - h_i)$$

If you use multinomial-trial syntax and the PREDPROBS option is in effect, this diagnostic is computed as follows:

$$\overline{C}_i = \mathbf{r}'_{Pi-J}\mathbf{M}_i^{-1}\mathbf{H}_i\mathbf{r}_{Pi-J}$$

The DIFDEV and DIFCHISQ statistics are diagnostics for detecting ill-fitted observations—observations that contribute heavily to the disagreement between the data and the predicted values of the fitted model. DIFDEV is the change in the deviance that results from deleting an individual observation, and DIFCHISQ is the change in the Pearson chi-square statistic that results from the same deletion. By using the one-step estimate, DIFDEV and DIFCHISQ for the ith observation are computed as follows:

$$\begin{aligned} \text{DIFDEV}_i &= r_{Di}^2 + \overline{C}_i \\ \text{DIFCHISQ}_i &= \overline{C}_i / h_i \end{aligned}$$

If you use multinomial-trial syntax and the PREDPROBS option is in effect, these statistics are computed as follows:

$$\begin{aligned} \text{DIFDEV}_i &= \mathbf{r}'_{Di-J}\mathbf{r}_{Di-J} + \overline{C}_i = r_{Li}^2 \\ \text{DIFCHISQ}_i &= \mathbf{r}'_{Pi-J}\mathbf{r}_{Pi-J} + \overline{C}_i \end{aligned}$$

Diagnostics for Models Fit by Generalized Estimating Equations (GEEs)

The diagnostic statistics in this section were developed by Preisser and Qaqish (1996). See the section "Generalized Estimating Equations" on page 1015 for further information and notation for generalized estimating equations (GEEs). The following additional notation is used in this section.

Partition the design matrix $\mathbf{X}$ and response vector $\mathbf{Y}$ by cluster; that is, let $\mathbf{X} = (X_1', \ldots, X_K')'$ and $\mathbf{Y} = (Y_1', \ldots, Y_K')'$, corresponding to the K clusters.

Let n_i be the number of responses for cluster i, and denote the total number of observations as $N = \sum_{i=1}^{K} n_i$. Denote the $n_i \times n_i$ diagonal matrix as A_i, where $V(\mu_{ij})$ is the jth diagonal element. If there is a WEIGHT statement, the diagonal element of A_i is $V(\mu_{ij})/w_{ij}$, where w_{ij} is the specified weight of the jth observation in the ith cluster. Let $\mathbf{B}$ be the $N \times N$ diagonal matrix with $g'(\mu_{ij})$ as diagonal elements, $i = 1, \ldots, K$, $j = 1, \ldots, n_i$. Let $\mathbf{B}_i$ be the $n_i \times n_i$ diagonal matrix that corresponds to cluster i, where $g'(\mu_{ij})$ is the jth diagonal element.

Let $\mathbf{W}$ be the $N \times N$ block diagonal weight matrix whose ith block, corresponding to the ith cluster, is the $n_i \times n_i$ matrix

$$\mathbf{W}_{ei} = \mathbf{B}_i^{-1}\mathbf{A}_i^{-\frac{1}{2}}\mathbf{R}_i^{-1}(\hat{\boldsymbol{\alpha}})\mathbf{A}_i^{-\frac{1}{2}}\mathbf{B}_i^{-1}$$

where $\mathbf{R}_i(\hat{\boldsymbol{\alpha}})$ is the working correlation matrix for cluster i that is computed using the estimated correlation parameters $\hat{\boldsymbol{\alpha}}$.

Let

$$\mathbf{Q}_i = \mathbf{X}_i(\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}\mathbf{X}_i'$$

where $\mathbf{X}_i$ is the $n_i \times p$ design matrix that corresponds to cluster i.

The cluster leverage statistic represents the leverage of cluster i and is contained in the matrix $\mathbf{H}_i = \mathbf{Q}_i\mathbf{W}_{ei}$. The leverage of cluster i is summarized by the trace of $\mathbf{H}_i$:

$$ch_i = \mathrm{tr}(\mathbf{H}_i)$$

The leverage h_t of the tth observation in the ith cluster is the tth diagonal element of $\mathbf{H}_i$.

Joint Tests and Type 3 Tests

Linear hypotheses for $\boldsymbol{\beta}$ are expressed in matrix form as

$$H_0\colon \mathbf{L}\boldsymbol{\beta} = \mathbf{c}$$

where $\mathbf{L}$ is a matrix of coefficients for the linear hypotheses and $\mathbf{c}$ is a vector of constants. The vector of regression coefficients $\boldsymbol{\beta}$ includes slope parameters and intercept parameters. The Wald chi-square statistic for testing H_0 is computed as

$$\chi^2_W = (\mathbf{L}\widehat{\boldsymbol{\beta}} - \mathbf{c})'[\mathbf{L}\widehat{\mathbf{V}}(\widehat{\boldsymbol{\beta}})\mathbf{L}']^{-1}(\mathbf{L}\widehat{\boldsymbol{\beta}} - \mathbf{c})$$

where $\widehat{\mathbf{V}}(\widehat{\boldsymbol{\beta}})$ is the estimated covariance matrix. Under H_0, χ^2_W has an asymptotic chi-square distribution with r degrees of freedom, where r is the rank of $\mathbf{L}$.

For models that use less-than-full-rank parameterization (as specified by the PARAM=GLM option in the CLASS statement), a Type 3 test of an effect of interest (main effect or interaction) is a test of the Type III estimable functions that are defined for that effect. When the model contains no missing cells, the Type 3 test of a main effect is equivalent to testing the hypothesis of equal marginal means. For more information about Type III estimable functions, see the chapter "The GLM Procedure" and the section "The Four Types of Estimable Functions" in *SAS/STAT User's Guide*. Also see Littell, Freund, and Spector (1991).

For models that use full-rank parameterization, all parameters are estimable when there are no missing cells, so it is unnecessary to define estimable functions. The standard test of an effect of interest in this case is the joint test that the values of the parameters associated with that effect are 0. For a model that uses effects parameterization (as specified by the PARAM=EFFECT option in the CLASS statement), the joint test for a main effect is equivalent to testing the equality of marginal means. For a model that uses reference parameterization (as specified by the PARAM=REF option in the CLASS statement), the joint test is equivalent to testing the equality of cell means at the reference level of the other model effects. For more information about the coding scheme and the associated interpretation of results, see Muller and Fetterman (2002, Chapter 14).

If there is no interaction term, the Type 3 test of an effect for a model that uses GLM parameterization is the same as the joint test of the effect for the model that uses full-rank parameterization. In this situation, the joint test is also called the Type 3 test. For a model that contains an interaction term and no missing cells, the Type 3 test of a component main effect under GLM parameterization is the same as the joint test of the component main effect under effect parameterization. Both test the equality of cell means. But this Type 3 test differs from the joint test under reference parameterization, which tests the equality of cell means at the reference level of the other component main effect. If some cells are missing, you can obtain meaningful tests only by testing a Type III estimation function, so in this case you should use GLM parameterization.

The results of a Type 3 test or a joint test do not depend on the order in which you specify the terms in the MODEL statement.

The Hosmer-Lemeshow Goodness-of-Fit Test

Sufficient replication within subpopulations is required to make the Pearson and deviance goodness-of-fit tests valid. When there are one or more continuous predictors in the model, the data are often too sparse to use these statistics. Hosmer and Lemeshow (2000) proposed a statistic that they show, through simulation, is distributed as chi-square when there is no replication in any of the subpopulations—in particular, when you have one or more continuous predictors in your model. Fagerland, Hosmer, and Bofin (2008) and Fagerland and Hosmer (2013, 2016) extend this test to polytomous response models.

The observations are sorted in increasing order of a scored value. For binary response variables, the scored value of an observation is its estimated event probability. The event is the response level that is specified in the response variable EVENT= option, the response level that is not specified in the REF= option, or, if neither of these options is specified, the response level identified in the "Response Profiles" table as "Ordered Value 1." For nominal response variables (LINK=GLOGIT), the scored value of an observation is 1 minus the estimated probability of the reference level (specified using the REF= option). For ordinal response variables, the scored value of an observation is $\sum_{i=1}^{K} i\,\widehat{\pi}_i$, where K is the number of response levels and $\widehat{\pi}_i$ is the predicted probability of the ith ordered response. This scored value is then rounded according to the value of the BINEPS= option.

The observations (and frequencies) are then combined into G groups. By default $G = 10$, but you can specify $G \geq 5$ in the NGROUPS= option of the LACKFIT option. For single-trial syntax, observations that have identical scored values are combined and are placed in the same group. Let F be the total frequency. The target frequency for each group is $T = \lfloor F/G + 0.5 \rfloor$, which is the integer part of $F/G + 0.5$. Load the first group ($g_j, j = 1$) with the observation that has the smallest scored value and with frequency f_1, and let the next-smallest observation have a frequency of f. PROC LOGSELECT performs the following steps for each observation to create the groups:

1. If $j = G$, then add this observation to group g_j.
2. Otherwise, if $f_j < T$ and $f_j + \lfloor f/2 \rfloor \leq T$, then add this observation to group g_j.
3. Otherwise, start loading the next group (g_{j+1}) with $f_{j+1} = f$, and set $j = j + 1$.

If the final group g_j has frequency $f_j < \frac{F}{2G}$, then add these observations to the preceding group. The total number of groups that are actually created, g, can be less than G. There must be at least three groups in order for the Hosmer-Lemeshow statistic to be computed.

For binary response variables, the Hosmer-Lemeshow goodness-of-fit statistic is obtained by calculating the Pearson chi-square statistic from the $2 \times g$ table of observed and expected frequencies, where g is the number of groups. The statistic is written

$$\chi^2_{\mathrm{HL}} = \sum_{j=1}^{g} \frac{(O_j - F_j \bar{\pi}_j)^2}{F_j \bar{\pi}_j (1 - \bar{\pi}_j)}$$

where F_j is the total frequency of subjects in the jth group, O_j is the total frequency of event outcomes in the jth group, and $\bar{\pi}_j$ is the average estimated predicted probability of an event outcome for the jth group. The Hosmer-Lemeshow statistic is then compared to a chi-square distribution with $(g - r)$ degrees of freedom, where the value of r can be specified in the DFREDUCE= suboption of the LACKFIT option. The default is $r = 2$.

For polytomous response variables, the Pearson chi-square statistic is computed from a $2K \times g$ table of observed and expected frequencies,

$$\chi^2_{\mathrm{HL}} = \sum_{j=1}^{g} \sum_{k=1}^{K} \frac{(O_{jk} - E_{jk})^2}{E_{jk}}$$

where O_{jk} is the sum of the observed frequencies and E_{jk} is the sum of the model-predicted probabilities of the observations in group j with response k. The Hosmer-Lemeshow statistic is then compared to a chi-square distribution. The number of degrees of freedom for this test of cumulative and adjacent-category logit models with the equal-slopes assumption is given by Fagerland and Hosmer (2013) and Fagerland and Hosmer (2016) as $(g–r)(K–1)+(K–2)$. The number of degrees of freedom for this test of the generalized logit model is given by Fagerland, Hosmer, and Bofin (2008) as $(g–r)(K–1)$, where K is the number of response levels. The degrees of freedom can also be specified using the DF= suboption of the LACKFIT option.

Large values of χ^2_{HL} (and small p-values) indicate a lack of fit of the model.

Odds Ratio Estimation

An odds ratio can be computed by exponentiating the difference of the logits between any two population profiles. Such computations are available regardless of parameterizations and interactions.

Specify one or more main-effects variables for which you want odds ratios to be computed in the ODDSRATIO statement. You can also specify variables on which constructed effects are based.

If your odds ratio variable is a classification variable, then the odds ratios that compare each pairwise difference between the levels of the odds ratio variable and its reference level are produced. You can compare all pairs of levels by setting the DIFF= option to ALL.

If your odds ratio variable is continuous, then the odds ratios compare the base subpopulation to one where that continuous variable is one unit larger. However, the change in odds for some amount other than one unit is often of greater interest. For example, a change of one pound in body weight might be too small to be considered important, whereas a change of 10 pounds might be more meaningful. You can control the units by specifying the UNIT suboption. If you want your units to represent a multiple of the odds ratio variable's standard error, then specify the STDERR suboption.

The odds ratio for a given odds ratio variable depends on the values of any other covariate with which it interacts. By default, interacting classification covariates are set to their reference level, and interacting continuous covariates are set to their sample means. You can specify the AT option to modify these values. Note that covariates that you specify in the AT option are ignored unless they interact with the odds ratio variable.

Because the log-odds ratio is a linear function of the parameters, the Wald confidence interval for the log-odds ratio can be derived from the parameter estimates and the estimated covariance matrix. Wald confidence intervals for the odds ratios are obtained by exponentiating the corresponding confidence limits for the log-odds ratios. Profile-likelihood confidence intervals are computed as in "Confidence Intervals for Parameters" on page 1024 by replacing the jth unit vector (e_j) with the difference of the logits.

A p-value for an odds ratio is the significance level such that the two-sided Wald confidence interval for that odds ratio has "1" as one of its endpoints. The percentage change in the odds is computed as $100(\psi - 1)$, where ψ is the odds ratio. See Allison (2012) for more information about these statistics.

The odds ratios are uniquely labeled by concatenating the following terms to the odds ratio variable's name:

1. If your model has a generalized logit link, then an odds ratio is computed for each response function, and the odds ratio variable's name is prefixed by the response variable name and the level that describes the logit, followed by a colon; for example, "Y 0:".
2. If the odds ratio variable is continuous and the UNIT suboption provides a value, then append "units=value"; otherwise, if the odds ratio variable is a classification variable, then append the levels that are being contrasted; for example, "cat vs dog".
3. Append all interacting covariates, preceded by "at"; for example, "at X=1.2 A=cat".

Risk Differences and Relative Risks

For a base profile L_1 and a comparison profile L_2 with corresponding model-predicted probabilities $p_1 = g^{-1}(L_1'\boldsymbol{\beta})$ and $p_2 = g^{-1}(L_2'\boldsymbol{\beta})$, where $g^{-1}(\cdot)$ is the inverse link function, the following two risk statistics are available:

Relative risk	$f_r = \frac{p_1}{p_2}$
Risk difference	$f_d = p_1 - p_2$

The covariance of these functions is given by the delta method as $\mathbf{g}(\boldsymbol{\beta})'\mathbf{V}(\boldsymbol{\beta})\mathbf{g}(\boldsymbol{\beta})$, where $\mathbf{V}(\boldsymbol{\beta})$ is the covariance matrix of the model parameters and $\mathbf{g}(\boldsymbol{\beta})$ is the gradient of the function f_r or f_d:

$$\frac{df_r}{d\boldsymbol{\beta}} = \frac{p_2\frac{dp_1}{d\boldsymbol{\beta}} - p_1\frac{dp_2}{d\boldsymbol{\beta}}}{p_2^2}$$
$$\frac{df_d}{d\boldsymbol{\beta}} = \frac{dp_1}{d\boldsymbol{\beta}} - \frac{dp_2}{d\boldsymbol{\beta}}$$

If your risk *variable* is a multinomial response variable, then the risk statistics compare the probability of different response levels by using a common population profile, L. That is, either $f_r = \frac{\Pr(Y=i|L)}{\Pr(Y=J|L)}$ or $f_d = \Pr(Y=i|L) - \Pr(Y=J|L)$ is computed for $i = 1, \ldots, J-1$.

The risks are uniquely labeled in the same fashion as the odds ratios as described in the section "Odds Ratio Estimation" on page 1033, except that all variables that are not set to their default value are included in the label.

Multithreading

The LOGSELECT procedure allocates data to different threads and calculates the likelihood function, gradient, and Hessian by accumulating the values from all threads. For more information about how PROC LOGSELECT uses threads, see the section "Multithreading" on page 102 in Chapter 2, "Shared Concepts."

Optimization Algorithms

Several optimization techniques are available in PROC LOGSELECT. You can choose a particular optimizer by using the TECHNIQUE= option in the PROC LOGSELECT statement. Table 18.18 summarizes the optimization techniques available in PROC LOGSELECT.

Table 18.18 Optimization Techniques

TECHNIQUE=	Algorithm
TRUREG	Trust region method
NEWRAP	Newton-Raphson method with line search
NRRIDG	Newton-Raphson method with ridging
QUANEW	Quasi-Newton methods
DBLDOG	Double-dogleg method
CONGRA	Conjugate gradient methods
LBFGS	Limited-memory BFGS quasi-Newton method
NMSIMP	Nelder-Mead simplex method

There is no algorithm for optimizing general nonlinear functions that always finds the global optimum for a general nonlinear optimization problem in a reasonable amount of time. Because no single optimization technique is always superior to others, PROC LOGSELECT provides a variety of optimization techniques that work well in various circumstances. However, you can devise problems for which none of the techniques in PROC LOGSELECT can find the correct solution. Moreover, nonlinear optimization can be computationally expensive in terms of time and memory, so you must be careful when matching an algorithm to a problem. The section "Choosing an Optimization Algorithm" on page 103 in Chapter 2, "Shared Concepts," is helpful in choosing a suitable optimization algorithm.

Memory Usage

In order to use memory efficiently in computing the components for optimization, PROC LOGSELECT reads your data table in multiple batches, or pages; each page contains at most a certain number of observations. During the optimization, the procedure reads the first page of observations from the data table into memory, creates the design rows, performs the appropriate log-likelihood, gradient, and Hessian computations on that page of observations, and then discards those observations and reads in the next page of data for processing.

Generally, smaller pages use less memory but can lead to longer computation times, whereas larger pages can run faster but use more memory. In particular, with sufficient memory, the optimization is typically fastest if you can fit all your data on a single page, because then you do not have to repeatedly reaccess the data table and recompute the design rows.

The default PAGEOBS=AUTO option determines that you have enough memory to use one page for the optimization if the number of observations in your data table for each thread on each machine node is less than $\min(b, 1\text{GB})/(8prt)$

where

- b = the approximate number of bytes available to a machine node, or 1GB
- t = the number of threads available to a machine node
- p = the number of parameters in your model
- r = $2n + 10$ for continuous response models
- r = $2n + y + 9$ for categorical response models
- n = the number of observations in the data table
- y = the number of response levels for categorical response models

If PROC LOGSELECT determines that you do not have enough memory, then PAGEOBS=256 is chosen to optimize the matrix computations, and a note is displayed in the SAS log that gives the value of the PAGEOBS= option that is required if you want to store all the data for a machine node on a single page.

Displayed Output

The following sections describe the output that PROC LOGSELECT produces. The output is organized into various tables, which are discussed in their order of appearance.

Store Information

The "Store Information" table displays basic information about the stored model: the item store name, the names of the library and data table used to create the model, the names of the action and procedure that created the item store, and the date when the model was stored. This table is not displayed when you fit a model.

Store Text

The "Store Text" table displays notes that are stored along with the stored model. This table is not displayed when you fit a model.

Model Information

The "Model Information" table displays basic information about the model, such as the response variable, the frequency variable, the link function, and the distribution of the data that is assumed by the LOGSELECT procedure. For information about how PROC LOGSELECT determines the response distribution, see the section "Response Distributions" on page 1010. If you specify the CODE statement, then the table displays the name of the variable in the scoring code that contains the classifications. If you do not also specify the PCATALL option, then the table also displays the name of the variable in the scoring code that contains the predicted probabilities.

Number of Observations

The "Number of Observations" table displays the number of observations that are read from the input data table and the number of observations that are used in the analysis. If you specify a FREQ statement, the table displays the sum of the frequencies that are read and used. If you specify a WEIGHT statement, the table displays the sum of $f_i w_i$ that are read and used. If you specify a PARTITION statement, the table displays the values for each role.

Response Profile

The "Response Profile" table displays the ordered value from which the LOGSELECT procedure determines the probability that is being modeled as an event in a binary model and displays the ordering of categories in a multinomial model. For each response category level, the frequency that is used in the analysis is reported. If you also specify the PRIOR= option in the MODEL statement, then the scaled prior probabilities are also displayed. You can affect the ordering of the response values by specifying response-options in the MODEL statement. For a binary model, the note that follows the "Response Profile" table indicates which outcome is modeled as the event. For an ordinal multinomial model, the note that follows the "Response Profile" table indicates how the ordered response levels are accumulated; for a nominal multinomial model, the note indicates the reference response level. If you specify a PARTITION statement, the table displays the values for each role.

Class Level Information

The "Class Level Information" table lists the levels of every variable specified in the CLASS statement. You should check this information to make sure that the data are correct. You can adjust the order of the levels by using the ORDER= option in the CLASS statement. You can suppress the "Class Level Information" table completely or partially by using the NOCLPRINT= option in the PROC LOGSELECT statement.

If the classification variables use a nonsingular parameterization, the "Class Level Information" table also displays the reference value for each variable.

Selection Information

When you specify the SELECTION statement, the LOGSELECT procedure by default produces a series of tables that display information about the model selection. The "Selection Information" table informs you about the model selection method, selection and stop criteria, and other parameters that govern the selection. You can suppress this table by specifying the DETAILS=NONE option in the SELECTION statement.

Iteration History

When you specify the ITHIST option in the PROC LOGSELECT statement, the "Iteration History" table displays, for each iteration of the optimization, the number of function evaluations (including gradient and Hessian evaluations), the value of the objective function, the change in the objective function from the previous iteration, and the absolute value of the largest (projected) gradient element. The objective function that is used in the optimization in the LOGSELECT procedure is normalized by default to enable comparisons across data tables that have different sampling intensity. You can control normalization by specifying the NORMALIZE= option in the PROC LOGSELECT statement.

Convergence Status

The convergence status table is a small ODS table that appears as a message that indicates whether the optimization succeeded and which convergence criterion was met. If the optimization fails, the message indicates the reason for the failure. If you save the convergence status table to an output data set, a numeric Status variable is added that enables you to assess convergence programmatically. The values of the Status variable indicate the following:

0 Convergence was achieved, or an optimization was not performed (because the TECHNIQUE=NONE option is specified).

1 The objective function could not be improved.

2 Convergence was not achieved because of a user interruption or because a limit was exceeded, such as the maximum number of iterations or the maximum number of function evaluations. To modify these limits, see the MAXITER=, MAXFUNC=, and MAXTIME= options in the PROC LOGSELECT statement.

3 Optimization failed to converge because function or derivative evaluations failed at the starting values or during the iterations or because a feasible point that satisfies the parameter constraints could not be found in the parameter space.

A numeric Separation variable is also added to the output convergence status table unless the NOCHECK option in the MODEL statement is specified. The values of this variable indicate the following:

2 Complete separation is detected.

1 Quasi-complete separation is detected.

0 Separation is not detected.

-1 Quasi-complete separation is not detected, but at least one of the diagonal elements of the dispersion matrix for the standardized observation vector is larger than 1,000.

-2 Quasi-complete separation is not checked because the standard errors are not available.

-3 Quasi-complete separation is not checked because there are fewer than eight iterations.

Entry and Removal Candidates

When you specify the DETAILS=ALL or DETAILS=STEPS option in the SELECTION statement, the LOGSELECT procedure produces the "Entry Candidates" or "Removal Candidates" table, which displays the effect names and values of the criterion used to select entering or departing effects at each step of the selection process. For each step, the effects are displayed in sorted order from best to worst of the selection criterion.

Selection Summary

When you specify the SELECTION statement, the LOGSELECT procedure produces the "Selection Summary" table, which displays information about which effects were added to or removed from the model in the various steps of the model selection process. The statistic that led to the entry or removal decision is also displayed. You can request further details about the model selection steps by specifying the DETAILS=STEPS or DETAILS=ALL option in the SELECTION statement. You can suppress the display of the "Selection Summary" table by specifying the DETAILS=NONE option in the SELECTION statement.

If you specify the LASSO selection method, then this table displays information about which effect was added to the model, the number of effects in the model, the lambda value, and the information criteria.

Stop Reason

When you specify the SELECTION statement, the LOGSELECT procedure produces a simple table that tells you why model selection stopped.

Selection Reason

When you specify the SELECTION statement, the LOGSELECT procedure produces a simple table that tells you why the final model was selected.

Selected Effects

When you specify the SELECTION statement, the LOGSELECT procedure produces a simple table that tells you which effects are in the final model.

Dimensions

The "Dimensions" table displays size measures that are derived from the model. It displays the number of columns in the current design matrix, the number of effects in the current design, the largest number of design columns associated with an effect, the rank of the matrix, and the number of parameters in the current model, including any scale parameters.

Global Test

The "Global Test" table provides a likelihood ratio test for the hypothesis of whether the final model provides a better fit than a model without effects (an "intercept-only" model). This table is not produced when the NOSTDERR option is specified nor when LASSO selection is performed.

If you specify the NOINT option in the MODEL statement, the reference model is one where the linear predictor is 0 for all observations.

Fit Statistics

The "Fit Statistics" table displays a variety of likelihood-based measures of fit. The values that the "Fit Statistics" table displays are not based on a normalized log-likelihood function. If you specify the PARTITION statement or the PARTFIT option, the table displays the values for each role along with statistics for comparing the training, validation, and testing results.

For more information about the statistics displayed in this table, see the section "Model Fit and Assessment Statistics" on page 1020.

This table is not produced in a GEE analysis.

Parameter Estimates

The parameter estimates, their estimated (asymptotic) standard errors, and p-values for the hypothesis that the parameter is 0 are presented in the "Parameter Estimates" table. If you request confidence intervals by specifying the CLB option in the MODEL statement, confidence limits are produced for the estimates.

This table is not produced by default in a GEE analysis. You can produce this table by specifying the PRINTMLE option in the REPEATED statement. In this case, the table is labeled "Analysis of Initial Parameter Estimates."

Parameter Estimates Covariance Matrix

When you specify the COVB option in the PROC LOGSELECT statement, the LOGSELECT procedure displays the covariance matrix of the parameter estimates. The covariance matrix is computed as the inverse of the negative of the matrix of second derivatives of the log-likelihood function with respect to the model parameters (the Hessian matrix), evaluated at the parameter estimates.

This table is not produced in a GEE analysis.

Parameter Estimates Correlation Matrix

When you specify the CORRB option in the PROC LOGSELECT statement, the LOGSELECT procedure displays the correlation matrix of the parameter estimates.

This table is not produced in a GEE analysis.

GEE Model Information

When you specify the REPEATED statement, the "GEE Model Information" table displays basic information about the GEE model, including the subject effects, any within-subject effects, the working correlation type, and details about the size of the working correlation matrix.

GEE Parameter Estimates Covariance Matrix Using Model-Based Covariance

When you specify the MCOVB or COVB option in the REPEATED statement, the "GEE Parameter Estimates Covariance Matrix Using Model-Based Covariance" table displays the parameter estimates covariance matrix from the GEE analysis that uses the model-based covariance estimate. For more information about the covariance estimates that are used for GEE analysis, see the section "Parameter Estimate Covariances" on page 1018.

GEE Parameter Estimates Correlation Matrix Using Model-Based Covariance

When you specify the MCORRB or CORRB option in the REPEATED statement, the "GEE Parameter Estimates Correlation Matrix Using Model-Based Covariance" table displays the parameter estimates correlation matrix from the GEE analysis that uses the model-based covariance estimate. For more information about the covariance estimates that are used for GEE analysis, see the section "Parameter Estimate Covariances" on page 1018.

GEE Parameter Estimates Covariance Matrix Using Empirical Covariance

When you specify the ECOVB or COVB option in the REPEATED statement, the "GEE Parameter Estimates Covariance Matrix Using Empirical Covariance" table displays the parameter estimates covariance matrix from the GEE analysis that uses the empirical covariance estimate. For more information about the covariance estimates that are used for GEE analysis, see the section "Parameter Estimate Covariances" on page 1018.

GEE Parameter Estimates Correlation Matrix Using Empirical Covariance

When you specify the ECORRB or CORRB option in the REPEATED statement, the "GEE Parameter Estimates Correlation Matrix Using Empirical Covariance" table displays the parameter estimates correlation matrix from the GEE analysis that uses the empirical covariance estimate. For more information about the covariance estimates that are used for GEE analysis, see the section "Parameter Estimate Covariances" on page 1018.

GEE Working Correlation Matrix

When you specify the CORRW option in the REPEATED statement, the "GEE Working Correlation Matrix" table displays the correlation matrix of the repeated measures. Its dimension depends on the number of repeated measures and any within-subject effects.

GEE Exchangeable Correlation

When you specify the CORRW and TYPE=EXCH options in the REPEATED statement, the "GEE Exchangeable Correlation" table displays the exchangeable correlation parameter. For more information about the working correlation structures, see the section "Working Correlation Matrix" on page 1016.

GEE Fit Criteria

When you specify the REPEATED statement, the "GEE Fit Criteria" table displays the information criteria for the GEE model. For more information about the fit statistics that are used with GEE, see the section "Quasi-likelihood Information Criterion" on page 1018.

GEE Parameter Estimates Using Empirical Standard Errors

When you specify the REPEATED statement, the "GEE Parameter Estimates Using Empirical Standard Errors" table presents the parameter estimates from the GEE analysis. The table displays the standard errors that are derived from the empirical covariance matrix. The table includes the parameter estimates, their estimated standard errors, and p-values for the hypothesis that the parameter is 0. If you request confidence intervals by specifying the CLB option in the MODEL statement, confidence limits are produced for the estimates.

For more information about the covariance estimates that are used for GEE analysis, see the section "Parameter Estimate Covariances" on page 1018.

GEE Parameter Estimates Using Model-Based Standard Errors

When you specify the MODELSE option in the REPEATED statement, the "GEE Parameter Estimates Using Model-Based Standard Errors" table presents the parameter estimates from the GEE analysis. The table displays the standard errors that are derived from the model-based covariance matrix. The table includes the parameter estimates, their estimated standard errors, and p-values for the hypothesis that the parameter is 0. If you request confidence intervals by specifying the CLB option in the MODEL statement, confidence limits are produced for the estimates.

For more information about the covariance estimates that are used for GEE analysis, see the section "Parameter Estimate Covariances" on page 1018.

Model Analysis of Variance (Type III)

When you specify the TYPE3 option in the MODEL statement, the LOGSELECT procedure produces the "Model Analysis of Variance (Type III)" table. This table displays tests that all parameters for a particular effect are 0. For more information, see the section "Joint Tests and Type 3 Tests" on page 1031.

This table is not produced in a GEE analysis.

Postfit Information

The "Postfit Information" table displays the name of the data table that the stored model uses to create the results that are displayed in the Restore section of the output, the number of observations that are read from this data table, and the number of observations that are used in the analyses. If you specify a FREQ statement, the table displays the sum of the frequencies that are read and used. If you specify a WEIGHT statement, the table displays the sum of $f_i w_i$ that are read and used. If the stored model is created using a PARTITION statement, the table displays the values for each role. This table is not displayed when you fit a model. This table is followed by "Response Profile" and "Fit Statistics" tables that are computed by using the input data table.

Association Statistics

When you specify the ASSOCIATION option in the PROC LOGSELECT statement and have binary, binomial, or ordinal response data, the "Association Statistics" table displays the concordance index c (the area under the ROC curve, AUC), Somers' D statistic (Gini's coefficient), Goodman-Kruskal's gamma statistic, and Kendall's tau-a statistic, along with raw counts of pairs of observations. If you also specify a PARTITION statement, a row is displayed for each role. For more information, see the section "Association Statistics" on page 1024.

Classification Table

The "Classification" table is displayed if you specify the CTABLE option without specifying an output data set. If you also specify a PARTITION statement, a table is displayed for each role. For more information, see the section "Classification Table and ROC Curves" on page 1022.

Partition for the Hosmer and Lemeshow Test

If you specify the LACKFIT option in the PROC LOGSELECT statement, the "Partition for the Hosmer and Lemeshow Test" table displays the grouping that is used in the Hosmer-Lemeshow test. If you also specify a PARTITION statement, the grouping is displayed for each role. For more information, see the section "The Hosmer-Lemeshow Goodness-of-Fit Test" on page 1032. For examples of using this partition, see Hosmer and Lemeshow (2000).

Hosmer and Lemeshow Goodness-of-Fit Test

If you specify the LACKFIT option in the PROC LOGSELECT statement, the "Hosmer and Lemeshow Goodness-of-Fit Test" table is displayed. It provides a test of the fit of the model; small p-values reject the null hypothesis that the fitted model is adequate. If you also specify a PARTITION statement, a test is displayed for each role. For more information, see the section "The Hosmer-Lemeshow Goodness-of-Fit Test" on page 1032.

Odds Ratios

The "Odds Ratios" table is displayed when you specify the ODDSRATIO statement. It displays the odds ratio estimates, the optional Wald confidence limits, a p-value, and the percentage change in odds. For more information, see the section "Odds Ratio Estimation" on page 1033.

Coefficients for Risk Computations

The "Coefficients for Risk Computations" table is displayed when you specify the E option in the RISK statement. It displays the profile coefficients that are used by the risk computations. For more information, see the section "Risk Differences and Relative Risks" on page 1034.

Risks

The "Risks" table is displayed when you specify the RISK statement. It displays the risk estimates, confidence limits, and a p-value. For more information, see the section "Risk Differences and Relative Risks" on page 1034.

Score Code Variables for Predicted Probability

When you specify the PCATALL option in the CODE statement, PROC LOGSELECT produces the "Score Code Variables for Predicted Probability" table. For each level of the response variable, this table displays the corresponding variable in the scoring code that contains the predicted probability.

Timing

The "Timing" table displays the amount of time (in seconds) that PROC LOGSELECT required in order to perform different tasks in the analysis.

OutputCasTables Table

The OutputCasTables table is a special table that has information about each CAS table that is created during a CAS procedure execution. The information for each CAS table consists of the CAS table name, the caslib in which the table resides, and the number of columns and rows in the CAS table. Because this table is not a typical ODS table that contains analytical results, you cannot include it in the *table-spec-list* in the DISPLAYOUT statement.

ODS Table Names

Each table that the LOGSELECT procedure creates has a name associated with it. You must use this name to refer to the table when you use the DISPLAY statement, the DISPLAYOUT statement, or ODS statements. These names are listed in Table 18.19.

The EFFECT and LSMEANS statements also create tables that are not listed in Table 18.19. For information about these tables, see the corresponding sections of Chapter 2, "Shared Concepts."

Table 18.19 ODS Tables Produced by PROC LOGSELECT

Table Name	Description	Statement	Options
Association	Association of predicted probabilities and observed responses	PROC LOGSELECT	ASSOCIATION
Classification	Classification table	PROC LOGSELECT	CTABLE
ClassInfo	Level information from the CLASS statement	CLASS	Default
ConvergenceStatus	Status of optimization at conclusion of optimization	Default	
CorrB	Correlation matrix of parameter estimates	PROC LOGSELECT	CORRB
CovB	Covariance matrix of parameter estimates	PROC LOGSELECT	COVB
Dimensions	Model dimensions	Default	
EntryCandidates	Details about candidates for entry into the model	SELECTION	METHOD=FORWARD or STEPWISE and DETAILS=STEP
FitStatistics	Fit statistics	Default	
GEEEmpPEst	GEE parameter estimates using empirical standard errors	REPEATED	Default
GEEExchCorr	Exchangeable correlation parameter	REPEATED	TYPE=EXCH
GEEFitCriteria	Measures of fit for GEE models	REPEATED	Default
GEEModInfo	Information about GEE model	REPEATED	Default
GEEModPEst	GEE parameter estimates using model-based standard errors	REPEATED	MODELSE

Table 18.19 *continued*

Table Name	Description	Statement	Options
GEENCorr	Correlation matrix of GEE parameter estimates using model-based covariance	REPEATED	CORRB or MCORRB
GEENCov	Covariance matrix of GEE parameter estimates using model-based covariance	REPEATED	COVB or MCOVB
GEERCorr	Correlation matrix of GEE parameter estimates using empirical covariance	REPEATED	CORRB or ECORRB
GEERCov	Covariance matrix of GEE parameter estimates using empirical covariance	REPEATED	COVB or ECOVB
GEEWCorr	Working correlation matrix of repeated measures	REPEATED	CORRW
GlobalTest	Test of the model versus the null model	Default	
IterHistory	Iteration history	PROC LOGSELECT	ITHIST
LackFitChiSq	Hosmer-Lemeshow chi-square test results	PROC LOGSELECT	LACKFIT
LackFitPartition	Partition for the Hosmer-Lemeshow test	PROC LOGSELECT	LACKFIT
LVectors	Coefficients for risk computations	RISK	E
ModelInfo	Information about the modeling environment	Default	
ModelAnova	Model analysis of variance (Type III)	MODEL	TYPE3
NObs	Number of observations read and used, and number of events and trials, if applicable	Default	
OddsRatios	Odds ratio estimates and Wald confidence intervals	ODDSRATIO	
OutputCasTables	See the section "OutputCasTables Table" on page 1044	OUTPUT DISPLAYOUT	OUT=
ParameterEstimates	Solutions for the parameter estimates associated with effects in MODEL statements	Default	
PostfitInfo	Input data table name and number of observations read and used for the restored model	PROC LOGSELECT	RESTORE= and DATA=

Table 18.19 *continued*

Table Name	Description	Statement	Options
PredProbName	Displays the predicted probability variable in the scoring code associated with each response level	CODE	PCATALL
RemovalCandidates	Details about candidates for removal from the model	SELECTION	METHOD=BACKWARD or STEPWISE and DETAILS=STEP
ResponseProfile	Response categories for binary and multinomial data	Default	
Risks	Risk estimates and Wald confidence intervals	RISK	
SelectedEffects	List of effects selected for the model	SELECTION	Default
SelectionInfo	Information about the settings for model selection	SELECTION	Default
SelectionReason	Reason why the particular model was selected	SELECTION	Default
SelectionSummary	Summary information about model selection steps	SELECTION	Default
StopReason	Reason for termination of model selection	SELECTION	Default
StoreInfo	Information about the restored model	PROC LOGSELECT	RESTORE=
StoreText	Notes stored along with the model	PROC LOGSELECT	RESTORE=
Timing	Absolute and relative times for tasks performed by the procedure	Default	

ODS Graphics

Statistical procedures use ODS Graphics to create graphs as part of their output. ODS Graphics is described in detail in Chapter 24, "Statistical Graphics Using ODS" (*SAS/STAT User's Guide*).

When ODS Graphics is enabled, the SELECTION statement can produce plots to help evaluate the selection process. For information about these plots, see the section "Model Selection Plots" on page 93 in Chapter 2, "Shared Concepts."

PROC LOGSELECT assigns a name to each graph that it creates using ODS. You can use these names to reference the graphs when using ODS. The names are listed in Table 18.20.

Table 18.20 Graphs Produced by PROC LOGSELECT

ODS Graph Name	Plot Description	PLOTS Option
CoefficientPanel	Coefficients and CHOOSE= criterion by step	COEFFICIENTS
ChooseCriterionPlot	CHOOSE= criterion by step	COEFFICIENTS(UNPACK)
CoefficientPlot	Coefficients by step	COEFFICIENTS(UNPACK)
CriterionPanel	Fit criteria by step	CRITERIA
AdjRSqPlot	Max-rescaled (adjusted) R-square by step	CRITERIA(UNPACK)
AICCPlot	Corrected Akaike's information criterion by step	CRITERIA(UNPACK)
AICPlot	Akaike's information criterion by step	CRITERIA(UNPACK)
DiffMeanPlot	Difference of means by step	CRITERIA(UNPACK)
McFaddenPlot	McFadden's R-square by step	CRITERIA(UNPACK)
MisclassPlot	Misclassification rate by step	CRITERIA(UNPACK)
SBCPlot	Schwarz Bayesian information criterion by step	CRITERIA(UNPACK)
TEST_ASEPlot	Average square error on testing data by step	CRITERIA(UNPACK)
TRAIN_ASEPlot	Average square error on training data by step	CRITERIA(UNPACK)
VAL_ASEPlot	Average square error on validation data by step	CRITERIA(UNPACK)
FitByRolePlot	Progression of average square error by role	FITBYROLE

Examples: LOGSELECT Procedure

The examples in this chapter are available in the GitHub repository located at https://github.com/sascommunities/doc-supplement-casstat.

NOTE: Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections "Using CAS Sessions and CAS Engine Librefs" on page 7 and "Loading a SAS Data Set onto a CAS Server" on page 7 in Chapter 2, "Shared Concepts."

Example 18.1: Model Selection

(View the complete code for this example (logex1.sas) in the example repository.)

```
title 'Example 1: Model Selection';
data mylib.getStarted;
   input C$ y x1-x10;
   datalines;
D  0  10.2  6  1.6  38  15  2.4  20  0.8  8.5  3.9
F  1  12.2  6  2.6  42  61  1.5  10  0.6  8.5  0.7
D  1   7.7  1  2.1  38  61    1  90  0.6  7.5  5.2
J  1  10.9  7  3.5  46  42  0.3   0  0.2    6  3.6
E  0  17.3  6  3.8  26  47  0.9  10  0.4  1.5  4.7
A  0  18.7  4  1.8   2  34  1.7  80    1  9.5  2.2
B  0   7.2  1  0.3  48  61  1.1  10  0.8  3.5    4
D  0   0.1  3  2.4   0  65  1.6  70  0.8  3.5  0.7
H  1   2.4  4  0.7  38  22  0.2  20    0    3  4.2
J  0  15.6  7  1.4   0  98  0.3   0    1    5  5.2
J  0  11.1  3  2.4  42  55  2.2  60  0.6  4.5  0.7
F  0     4  6  0.9   4  36  2.1  30  0.8    9  4.6
A  0   6.2  2  1.8  14  79  1.1  70  0.2    0  5.1
H  0   3.7  3  0.8  12  66  1.3  40  0.4  0.5  3.3
A  1   9.2  3  2.3  48  51  2.3  50    0    6  5.4
G  0    14  3    2  18  12  2.2   0    0    3  3.4
E  1  19.5  6  3.7  26  81  0.1  30  0.6    5  4.8
C  0    11  3  2.8  38   9  1.7  50  0.8  6.5  0.9
I  0  15.3  7  2.2  20  98  2.7 100  0.4    7  0.8
H  1   7.4  4  0.5  28  65  1.3  60  0.2  9.5  5.4
F  0  11.4  2  1.4  42  12  2.4  10  0.4    1  4.5
C  1  19.4  1  0.4  42   4  2.4  10    0  6.5  0.1
G  0   5.9  4  2.6  12  57  0.8  50  0.4    2  5.8
G  1  15.8  6  3.7  34   8  1.3  90  0.6  2.5  5.7
I  0    10  3  1.9  16  80    3  90  0.4  9.5  1.9
E  0  15.7  1  2.7  32  25  1.7  20  0.2  8.5    6
G  0    11  5  2.9  48  53  0.1  50    1  3.5  1.2
J  1  16.8  0  0.9  14  86  1.4  40  0.8    9    5
D  1    11  4  3.2  48  63  2.8  90  0.6    0  2.2
J  1   4.8  7  3.6  24   1  2.2  20    1  8.5  0.5
J  1  10.4  5    2  42  56    1  20    0  3.5  4.2
G  0  12.7  7  3.6   8  56  2.1  70    1  4.5  1.5
```

```
G  0   6.8  1  3.2  30  27  0.6   0  0.8    2  5.6
E  0   8.8  0  3.2   2  67  0.7  10  0.4    1    5
I  1   0.2  0  2.9  10  41  2.3  60  0.2    9  0.3
J  1   4.6  7  3.9  50  61  2.1  50  0.4    3  4.9
J  1   2.3  2  3.2  36  98  0.1  40  0.6  4.5  4.3
I  0  10.8  3  2.7  28  58  0.8  80  0.8    3    6
B  0   9.3  2  3.3  44  44  0.3  50  0.8  5.5  0.4
F  0   9.2  6  0.6   4  64  0.1   0  0.6  4.5  3.9
D  0   7.4  0  2.9  14   0  0.2  30  0.8  7.5  4.5
G  0  18.3  3  3.1   8  60  0.3  60  0.2    7  1.9
F  0   5.3  4  0.2  48  63  2.3  80  0.2    8  5.2
C  0   2.6  5  2.2  24   4  1.3  20    0    2  1.4
F  0  13.8  4  3.6   4   7  1.1  10  0.4  3.5  1.9
B  1  12.4  6  1.7  30  44  1.1  60  0.2    6  1.5
I  0   1.3  1  1.3   8  53  1.1  70  0.6    7  0.8
F  0  18.2  7  1.7  26  92  2.2  30    1  8.5  4.8
J  0   5.2  2  2.2  18  12  1.4  90  0.8    4  4.9
G  1   9.4  2  0.8  22  86  0.4  30  0.4    1  5.9
J  1  10.4  2  1.7  26  31  2.4  10  0.2    7  1.6
J  0    13  1  1.8  14  11  2.3  50  0.6  5.5  2.6
A  0  17.9  4  3.1  46  58  2.6  90  0.6  1.5  3.2
D  1  19.4  6    3  20  50  2.8 100  0.2    9  1.2
I  0  19.6  3  3.6  22  19  1.2   0  0.6    5  4.1
I  1     6  2  1.5  30  30  2.2  20  0.4  8.5  5.3
G  0  13.8  1  2.7   0  52  2.4  20  0.8    6    2
B  0  14.3  4  2.9  30  11  0.6  90  0.6  0.5  4.9
E  0  15.6  0  0.4  38  79  0.4  80  0.4    1  3.3
D  0    14  2    1  22  61    3  90  0.6    2  0.1
C  1   9.4  5  0.4  12  53  1.7  40    0    3  1.1
H  0  13.2  1  1.6  40  15  0.7  40  0.2    9  5.5
A  0  13.5  5  2.4  18  89  1.6  20  0.4  9.5  4.7
E  0   2.6  4  2.3  38   6  0.8  20  0.4    5  5.3
E  0  12.4  3  1.3  26   8  2.8  10  0.8    6  5.8
D  0   7.6  2  0.9  44  89  1.3  50  0.8    6  0.4
I  0  12.7  1  2.3  42   6  2.4  10  0.4    1    3
C  1  10.7  4  3.2  28  23  2.2  90  0.8  5.5  2.8
H  0  10.1  2  2.3  10  62  0.9  50  0.4  2.5  3.7
C  1  16.6  1  0.5  12  88  0.1  20  0.6  5.5  1.8
I  1   0.2  3  2.2   8  71  1.7  80  0.4  0.5  5.5
C  0  10.8  4  3.5  30  70  2.3  60  0.4  4.5  5.9
F  0   7.1  4    3  14  63  2.4  70    0    7  3.1
D  0  16.5  1  3.3  30  80  1.6  40    0  3.5  2.7
H  0  17.1  7  2.1  30  45  1.5  60  0.6  0.5  2.8
D  0   4.3  1  1.5  24  44    0  70    0    5  0.5
H  0    15  2  0.2  14  87  1.8  50    0  4.5  4.7
G  0  19.7  3  1.9  36  99  1.5  10  0.6    3  1.7
H  1   2.8  6  0.6  34  21    2  60    1    9  4.7
G  0  16.6  3  3.3  46   1  1.4  70  0.6  1.5  5.3
E  0  11.7  5  2.7  48   4  0.9  60  0.8  4.5  1.6
F  0  15.6  3  0.2   4  79  0.5   0  0.8  1.5  2.9
C  1   5.3  6  1.4   8  64    2  80  0.4    9  4.2
B  1   8.1  7  1.7  40  36  1.4  60  0.6    6  3.9
I  0  14.8  2  3.2   8  37  0.4  10    0  4.5    3
D  0   7.4  4    3  12   3  0.6  60  0.6    7  0.7
```

```
D  0   4.8  3  2.3  44  41  1.9  60  0.2    3  3.1
A  0   4.5  0  0.2   4  48  1.7  80  0.8    9  4.2
D  0   6.9  6  3.3  14  92  0.5  40  0.4  7.5    5
B  0   4.7  4  0.9  14  99  2.4  80    1  0.5  0.7
I  1   7.5  4  2.1  20  79  0.4  40  0.4  2.5  0.7
C  0   6.1  0  1.4  38  18  2.3  60  0.8  4.5  0.7
C  0  18.3  1    1  26  98  2.7  20    1  8.5  0.5
F  0  16.4  7  1.2  32  94  2.9  40  0.4  5.5  2.1
I  0   9.4  2  2.3  32  42  0.2  70  0.4  8.5  0.3
F  1  17.9  4  1.3  32  42    2  40  0.2    1  5.4
H  0  14.9  3  1.6  36  74  2.6  60  0.2    1  2.3
C  0  12.7  0  2.6   0  88  1.1  80  0.8  0.5  2.1
F  0   5.4  4  1.5   2   1  1.8  70  0.4  5.5  3.6
J  1  12.1  4  1.8  20  59  1.3  60  0.4    3  3.8
;
```

The following statements examine the same data as in the section "Getting Started: LOGSELECT Procedure" on page 964, but they request model selection via the forward selection technique. Effects that provide the best improvement to the selection criterion, SBC, are added until no more effects can improve the selection criterion. The DETAILS=ALL option in the SELECTION statement produces all tables that are related to model selection. The PLOTS=ALL option produces graphics to help you interpret the selection process. ODS Graphics must be enabled before you can request plots. For more information about ODS Graphics, see the section "ODS Graphics" on page 1046.

```
ods graphics on;

proc logselect data=mylib.getStarted;
   class C;
   model y = C x1-x10;
   selection method=forward details=all plots=all;
run;

ods graphics off;
```

The model selection tables are shown in Output 18.1.1 through Output 18.1.3. Results from the selected model are shown in Output 18.1.4 and Output 18.1.5. Selection graphics that the PLOTS= option produces are displayed in Output 18.1.6 and Output 18.1.7.

The "Selection Information" table in Output 18.1.1 summarizes the settings for the model selection. Effects are added to the model only if they produce a significant improvement, which is determined by comparing the values of their SBCs. The forward selection stops three steps after the smallest SBC is obtained or when all effects have been added to the model.

Output 18.1.1 Selection Information

Example 1: Model Selection

The LOGSELECT Procedure

Selection Information	
Selection Method	Forward
Select Criterion	SBC
Stop Criterion	SBC
Effect Hierarchy Enforced	None
Stop Horizon	3

For each step of the selection process, the DETAILS=ALL option displays the candidate effects for entering the model along with their SELECT= criterion. Output 18.1.2 displays this table for the first step; the other steps are not shown here.

Output 18.1.2 Step1 Entry Candidates

Best 10 Entry Candidates		
Rank	**Effect**	**SBC**
1	**x8**	128.7318
2	**x2**	129.0423
3	**x4**	129.3708
4	**x9**	131.0854
5	**x1**	131.8215
6	**x10**	132.4430
7	**x5**	132.9424
8	**x3**	132.9667
9	**x7**	133.0021
10	**x6**	133.0283

The DETAILS=ALL option also displays the dimensions, global test, fit statistics, and parameter estimates at each step of the selection process; these details are not shown here.

When the selection procedure is complete, the "Selection Summary" table in Output 18.1.3 shows the effects that were added to the model and the value of their selection criterion (and the choose and stop criteria, if they are specified). Step 0 refers to the null model that contains only an intercept. In the next step, effect x8 made the most significant contribution to the model among the candidate effects, according to the SBC statistic. In step 2, x2 made the most significant contribution when an effect was added to a model that contains the intercept and x8. In the three subsequent steps, no effect could be added to the model that would reduce the SBC, so variable selection stopped because the stop horizon (see Output 18.1.1) indicates that at most three steps beyond the minimum SBC value are taken.

In Output 18.1.3, the "Selection Summary" table is followed by three small tables that summarize why the process stopped and which model is selected.

Output 18.1.3 Selection Summary Information

Selection Summary			
Step	Effect Entered	Number Effects In	SBC
0	Intercept	1	128.4253
1	x8	2	128.7318
2	x2	3	128.2892*
3	x4	4	130.0901
4	x9	5	131.9534
5	x1	6	134.7945
* Optimal Value Of Criterion			

Selection stopped at a local minimum of the STOP criterion.

The model at step 2 is selected.

Selected Effects: Intercept x2 x8

Output 18.1.4 displays information about the selected model. Notice that the –2 log-likelihood value in the "Fit Statistics" table is larger than the value for the full model in Figure 18.9. This is expected because the selected model contains only a subset of the parameters. Because the selected model is more parsimonious than the full model, the discrepancy between the –2 log likelihood and the information criteria is less severe than previously noted.

Output 18.1.4 Fit Statistics and Null Test

Dimensions	
Columns in Design	3
Number of Effects	3
Max Effect Columns	1
Rank of Design	3
Parameters in Optimization	3

Testing Global Null Hypothesis: BETA=0			
Test	DF	Chi-Square	Pr > ChiSq
Likelihood Ratio	2	9.4237	0.0090

Fit Statistics	
-2 Log Likelihood	114.39646
AIC (smaller is better)	120.39646
AICC (smaller is better)	120.64646
SBC (smaller is better)	128.21197

The parameter estimates of the selected model are shown in Output 18.1.5. Notice that the effects are listed in the "Parameter Estimates" table in the order in which they were specified in the MODEL statement and not in the order in which they were added to the model.

Output 18.1.5 Parameter Estimates

Parameter Estimates

Parameter	DF	Estimate	Standard Error	Chi-Square	Pr > ChiSq
Intercept	1	0.858377	0.550302	2.4331	0.1188
x2	1	-0.250240	0.114578	4.7700	0.0290
x8	1	1.783956	0.790845	5.0885	0.0241

You can construct the prediction equation for this model from the parameter estimates as follows. The estimated linear predictor for an observation is

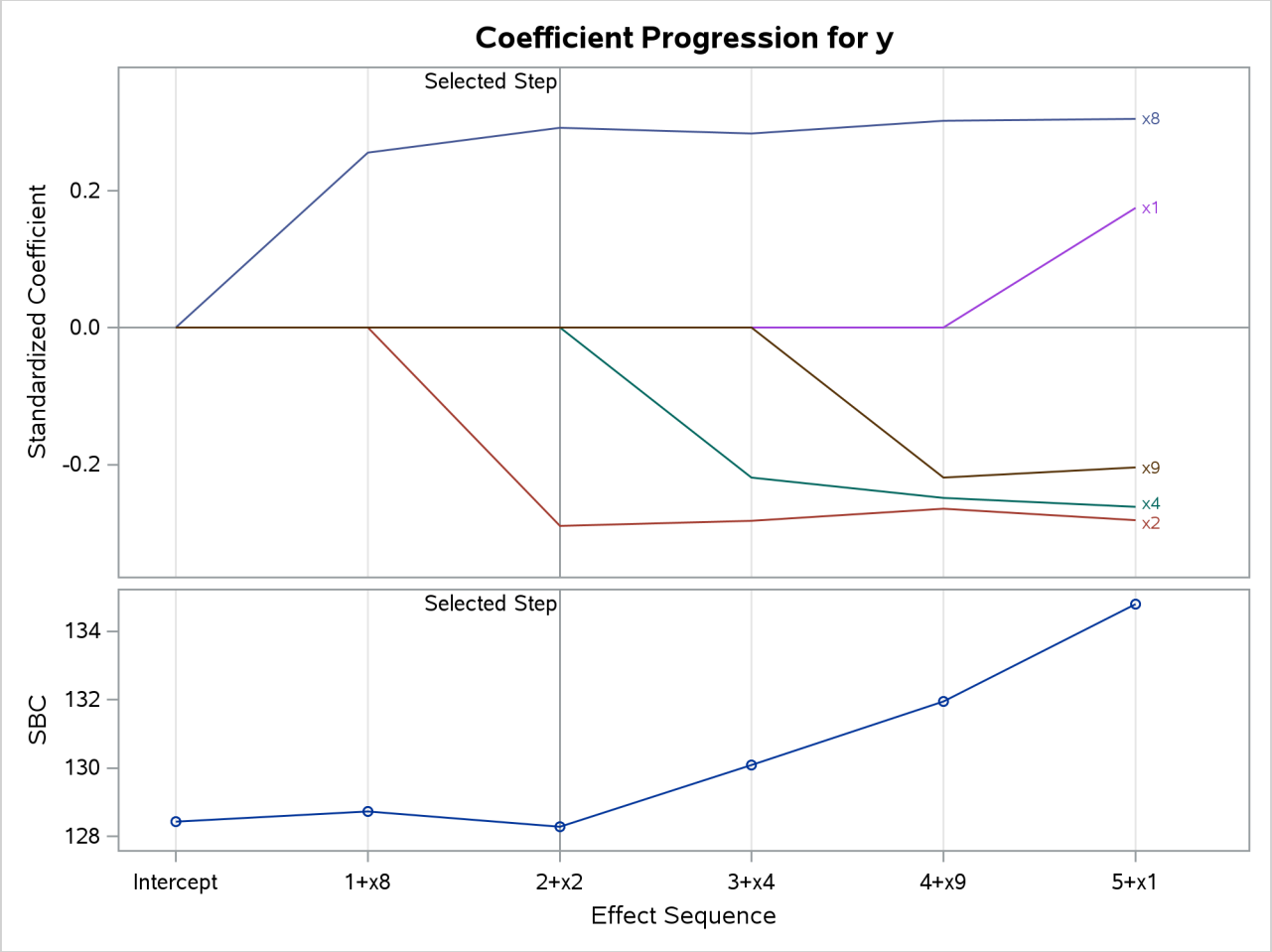
$$\widehat{\eta} = 0.8584 - 0.2502 \times x_2 + 1.7840 \times x_8$$

The predicted probability that variable y takes the value 0 is

$$\widehat{\Pr}(Y = 0) = \frac{1}{1 + \exp\{-\widehat{\eta}\}}$$

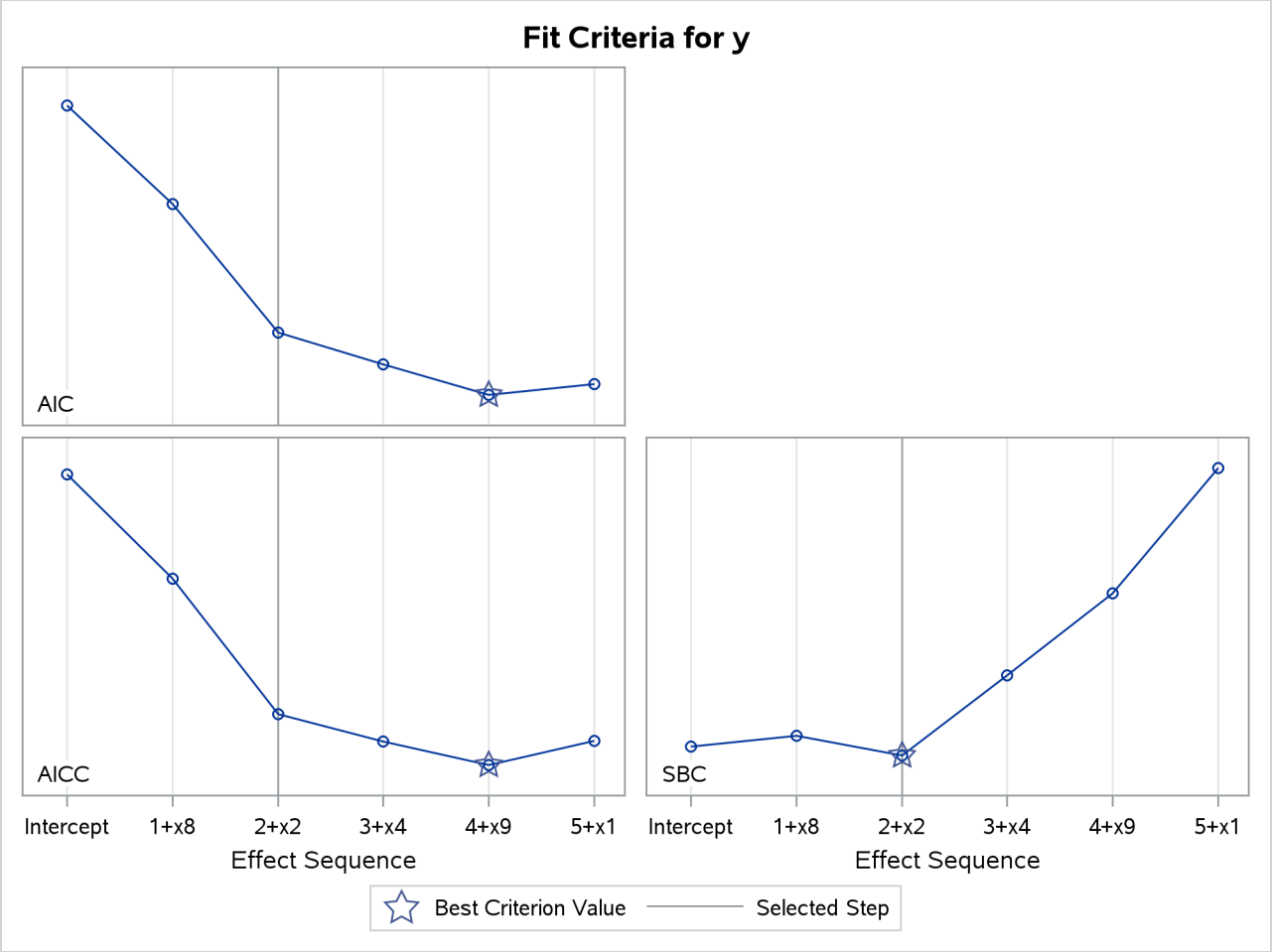
The coefficient panel in Figure 18.1.6 enables you to visualize the selection process. In this plot, standardized coefficients of all the effects that are selected at some step of the stepwise method are plotted as a function of the step number. This enables you to assess the relative importance of the effects that are selected at any step of the selection process and to know when effects entered the model. The lower plot in the panel shows how the criterion that is used to choose the selected model changes as effects enter or leave the model.

Output 18.1.6 Coefficient Progression



The criterion panel in Figure 18.1.7 provides a graphical view of the progression of the fit criteria as the selection process evolves. Notice at Step 0 that the SBC criterion seems to be at its minimum. Because the stop horizon value is 3 (see Output 18.1.1), three more steps are taken in case Step 0 is a local optimum. In this example, the global optimum is found two steps later.

Output 18.1.7 Criterion Panel



Example 18.2: Modeling Binomial Data

(View the complete code for this example (logex2.sas) in the example repository.)

If $Y_1, \ldots, Y_n$ are independent binary (Bernoulli) random variables with common success probability π, then their sum is a binomial random variable. In other words, a binomial random variable with parameters n and π can be generated as the sum of n Bernoulli(π) random experiments. The LOGSELECT procedure uses a special syntax, the events/trials syntax, to express data in binomial form.

Consider the following data, taken from Cox and Snell (1989, pp. 10–11), of the number of ingots not ready for rolling (r) out of n tested for a number of combinations of heating time and soaking time. If each test is carried out independently and if for a particular combination of heating and soaking time there is a constant probability that the tested ingot is not ready for rolling, then the random variable r follows a binomial(n, π) distribution, where the success probability π is a function of heating and soaking time.

```
data Ingots;
   input Heat Soak r n @@;
   Obsnum= _n_;
   datalines;
7 1.0 0 10  14 1.0 0 31  27 1.0 1 56  51 1.0 3 13
7 1.7 0 17  14 1.7 0 43  27 1.7 4 44  51 1.7 0  1
7 2.2 0  7  14 2.2 2 33  27 2.2 0 21  51 2.2 0  1
7 2.8 0 12  14 2.8 0 31  27 2.8 1 22  51 4.0 0  1
7 4.0 0  9  14 4.0 0 19  27 4.0 1 16
;
data mylib.Ingots;
   set Ingots;
run;
```

The following statements show the use of the events/trials syntax to model the binomial response. The events variable in this situation is *r* (the number of ingots not ready for rolling), and the trials variable is *n* (the number of ingots that are tested). The dependency of the probability of not being ready for rolling is modeled as a function of heating time, soaking time, and their interaction. The ASSOCIATION option displays ordinal measures of association between the observed responses and the predicted probabilities. The CTABLE option saves statistics to the mylib.Roc data table that are used to evaluate the predictive power of the model. The OUTPUT statement stores the linear predictors and the predicted probabilities in the mylib.Out data table along with the other variables from the input data table.

```
proc logselect data=mylib.Ingots association ctable(out=mylib.Roc nocounts tpf fpf);
   model r/n = Heat Soak Heat*Soak;
   output out=mylib.Out xbeta predicted=Pred copyvars=(Heat Soak);
run;
```

The "Model Information" table shows that the data are modeled as binomially distributed with a logit link function (Output 18.2.1). This is the default link function in the LOGSELECT procedure for binary and binomial data. The procedure estimates the parameters of the model by a Newton-Raphson algorithm.

Output 18.2.1 Model Information and Number of Observations

The LOGSELECT Procedure

Model Information	
Data Source	INGOTS
Response Variable (Events)	r
Response Variable (Trials)	n
Distribution	Binomial
Link Function	Logit
Optimization Technique	Newton-Raphson with Ridging

Number of Observations Read	19
Number of Observations Used	19

Response Profile		
Ordered Value	**Binary Outcome**	**Total Frequency**
1	**Event**	12
2	**Nonevent**	375

The second table in Output 18.2.1 shows that all 19 observations in the input data table were used in the analysis and the "Response Profile" table shows that the total number of events and nonevents equals 12 and 387, respectively.

Output 18.2.2 displays the convergence status table for this run. The LOGSELECT procedure satisfies the GCONV= convergence criterion.

Output 18.2.2 Convergence Status

Convergence criterion (GCONV=1E-8) satisfied.

Output 18.2.3 displays the "Dimensions" table for the model. The design matrix of the model (the **X** matrix) has four columns, which correspond to the intercept, the Heat effect, the Soak effect, and the interaction of the Heat and Soak effects. The model is nonsingular because the rank of the crossproducts matrix equals the number of columns in **X**. All parameters are estimable, and all participate in the optimization.

Output 18.2.3 Dimensions in Binomial Logistic Regression

Dimensions	
Columns in Design	4
Number of Effects	4
Max Effect Columns	1
Rank of Design	4
Parameters in Optimization	4

Output 18.2.4 displays the "Fit Statistics" table for this run. Evaluated at the converged estimates, –2 times the value of the log-likelihood function equals 27.9569. Further fit statistics are also displayed, all of them in "smaller is better" form. The AIC, AICC, and SBC criteria are used to compare non-nested models and to penalize the model fit for the number of observations and parameters. You can use the –2 log-likelihood value to compare nested models by way of a likelihood ratio test.

Output 18.2.4 Fit Statistics

Fit Statistics	
-2 Log Likelihood	27.95689
AIC (smaller is better)	35.95689
AICC (smaller is better)	38.81403
SBC (smaller is better)	39.73464

Output 18.2.5 shows the test of the global hypothesis that the effects jointly do not affect the probability of ingot readiness. You can obtain the chi-square test statistic by comparing the –2 log-likelihood value of the model with covariates to the value in the intercept-only model. The test is significant, with a p-value of 0.0082. One or more of the effects in the model have a significant impact on the probability of ingot readiness.

Output 18.2.5 Null Test

Testing Global Null Hypothesis: BETA=0			
Test	**DF**	**Chi-Square**	**Pr > ChiSq**
Likelihood Ratio	3	11.7663	0.0082

The "Parameter Estimates" table in Output 18.2.6 displays the estimates and standard errors of the model effects.

Output 18.2.6 Parameter Estimates

Parameter Estimates					
Parameter	DF	Estimate	Standard Error	Chi-Square	Pr > ChiSq
Intercept	1	-5.990191	1.666622	12.9183	0.0003
Heat	1	0.096339	0.047067	4.1896	0.0407
Soak	1	0.299574	0.755068	0.1574	0.6916
Heat * Soak	1	-0.008840	0.025319	0.1219	0.7270

Output 18.2.7 displays the "Association Statistics" table, which is produced when you specify the ASSOCIATION option in the PROC LOGSELECT statement. The table contains four measures of association for assessing the predictive ability of a model. For more information, see the section "Association Statistics" on page 1024.

Output 18.2.7 Association of Observed Responses and Predicted Probabilities

Association of Predicted Probabilities and Observed Responses	
Concordance Index (AUC)	0.7706
Somers' D	0.5411
Gamma	0.5858
Tau-a	0.0326
Pairs	4500
Percent Concordant	73.2444
Percent Discordant	19.1333
Percent Tied	7.6222

You can construct the prediction equation of the model from the "Parameter Estimates" table. For example, an observation with Heat equal to 14 and Soak equal to 1.7 has the linear predictor

$$\widehat{\eta} = -5.9902 + 0.09634 \times 14 + 0.2996 \times 1.7 - 0.00884 \times 14 \times 1.7 = -4.3425$$

The probability that an ingot with these characteristics is not ready for rolling is

$$\widehat{\pi} = \frac{1}{1 + \exp\{-(-4.3425)\}} = 0.01284$$

PROC LOGSELECT computes these linear predictors and probabilities and stores them in the mylib.Out data table. The following PROC PRINT statement produces Output 18.2.8, which shows the probability that an ingot with Heat equal to 14 and Soak equal to 1.7 is not ready for rolling:

```
proc print data=mylib.Out;
   where Heat=14 & Soak=1.7;
run;
```

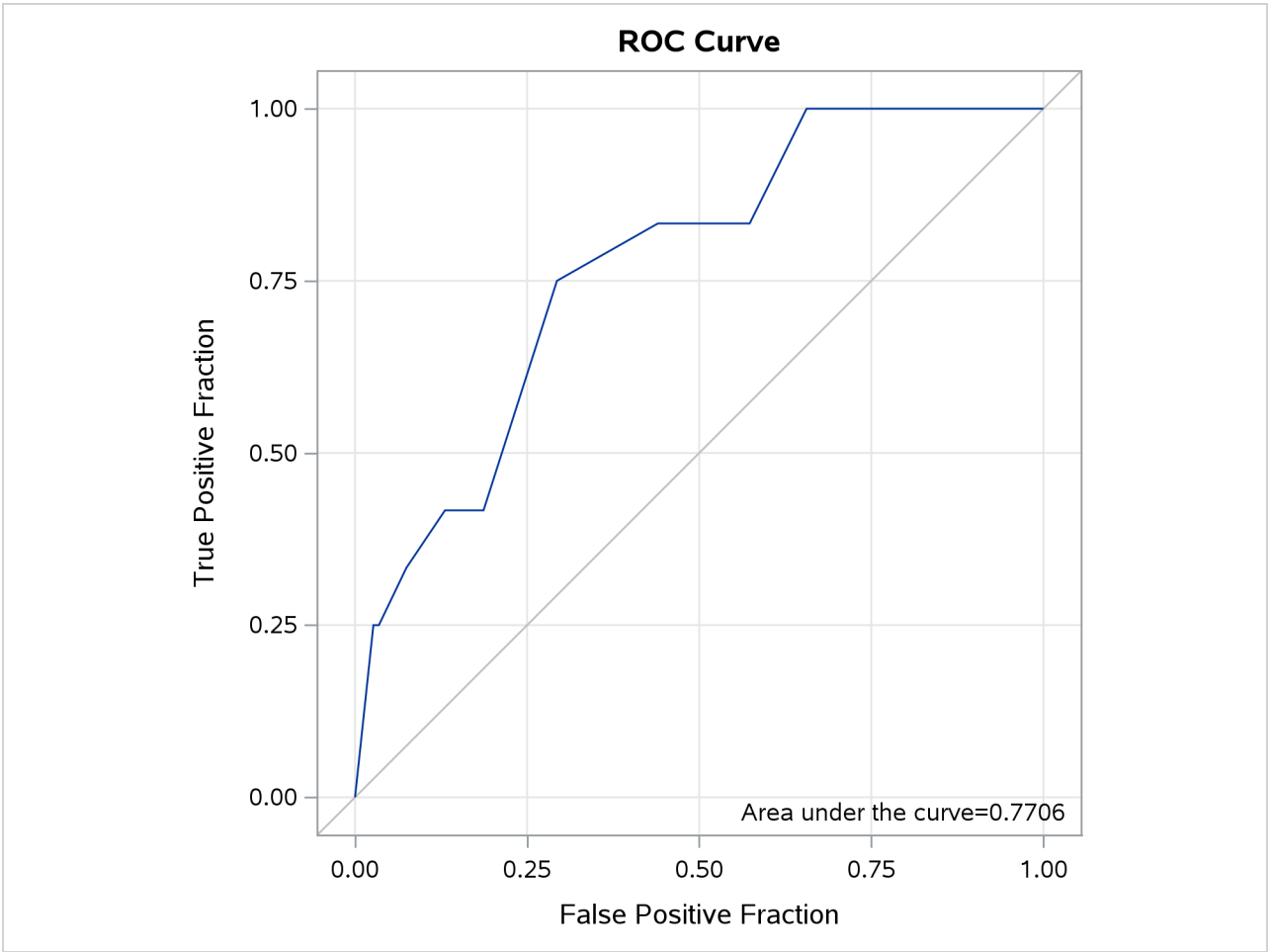
Output 18.2.8 Predicted Probability for Heat=14 and Soak=1.7

Obs	Pred	_XBETA_	Heat	Soak
1	0.012836	-4.34256	14	1.7

The CTABLE option computes statistics for binary response models (on the basis of classifying observations according to whether their predicted probabilities exceed certain values) and stores the results in the mylib.Roc data table. For more information, see the section "Classification Table and ROC Curves" on page 1022. You can use this data table to display the ROC curve by using the SGPLOT procedure as follows:

```
ods graphics on;
proc sgplot data=mylib.Roc aspect=1 noautolegend;
   title 'ROC Curve';
   xaxis values=(0 to 1 by 0.25) grid offsetmin=.05 offsetmax=.05;
   yaxis values=(0 to 1 by 0.25) grid offsetmin=.05 offsetmax=.05;
   lineparm x=0 y=0 slope=1 / lineattrs=(color=ligr);
   series x=FPF y=TPF;
   inset 'Area under the curve=0.7706' / position=bottomright;
run;
```

Output 18.2.9 Receiver Operating Characteristics Curve



Multinomial-Trial Syntax for Grouped Binary Data

You can perform the same analysis by using the multinomial-trial syntax. First you need to compute the number of ingots that are ready for rolling, *a*:

```
data mylib.Ingots2;
   set Ingots;
   a = n - r;
run;
```

Then you specify the event variable, followed by the nonevent variable, in the MODEL statement:

```
proc logselect data=mylib.Ingots2 association ctable(out=mylib.Roc nocounts tpf fpf);
   model r a = Heat Soak Heat*Soak;
   output out=mylib.Out xbeta predicted=Pred copyvars=(Heat Soak);
run;
```

The only difference from the preceding results is that in this case the "Model Information" table shown in Output 18.2.10 displays the event and nonevent variables.

Output 18.2.10 Model Information

The LOGSELECT Procedure

Model Information	
Data Source	INGOTS2
Response Variable (Events)	r
Response Variable (Nonevents)	a
Distribution	Binomial
Link Function	Logit
Optimization Technique	Newton-Raphson with Ridging

Single-Trial Syntax for Binary Data

Binomial data are a form of grouped binary data in which "successes" in the underlying Bernoulli trials are totaled. You can thus unwind data for which you use the events/trials syntax and fit these data by using techniques for binary data.

The following DATA step expands the Ingots data table, which contains 12 events in 387 trials, into a data table that has 387 observations:

```
data mylib.Ingots_binary;
   set Ingots;
   do i=1 to n;
     if i <= r then y=1; else y = 0;
     output;
   end;
run;
```

The following statements fit the model that contains the Heat effect, the Soak effect, and their interaction to the data table. The `event='1'` response-variable option in the MODEL statement ensures that the LOGSELECT procedure models the probability that the variable y takes the value 1.

```
proc logselect data=mylib.Ingots_binary;
   model y(event='1') = Heat Soak Heat*Soak;
run;
```

Output 18.2.11 displays the "Model Information," "Number of Observations," and "Response Profile" tables. The data are now modeled as binary (Bernoulli distributed) with a logit link function. The "Response Profile" table shows that the binary response breaks down into 375 observations where y=0 and 12 observations where y=1.

Output 18.2.11 Model Information in Binary Model

The LOGSELECT Procedure

Model Information	
Data Source	INGOTS_BINARY
Response Variable	y
Distribution	Binary
Link Function	Logit
Optimization Technique	Newton-Raphson with Ridging

Number of Observations Read	387
Number of Observations Used	387

Response Profile

Ordered Value	y	Total Frequency
1	**0**	375
2	**1**	12

Probability modeled is y = 1.

Output 18.2.12 displays the result of the test of the global null hypothesis and the parameter estimates. These results match those in Output 18.2.5 and Output 18.2.6.

Output 18.2.12 Null Test and Parameter Estimates

Testing Global Null Hypothesis: BETA=0

Test	DF	Chi-Square	Pr > ChiSq
Likelihood Ratio	3	11.7663	0.0082

Parameter Estimates

Parameter	DF	Estimate	Standard Error	Chi-Square	Pr > ChiSq
Intercept	1	-5.990191	1.666622	12.9183	0.0003
Heat	1	0.096339	0.047067	4.1896	0.0407
Soak	1	0.299574	0.755068	0.1574	0.6916
Heat * Soak	1	-0.008840	0.025319	0.1219	0.7270

Example 18.3: Partitioning Data

(View the complete code for this example (logex3.sas) in the example repository.)

The Sashelp.JunkMail data set comes from a study that classifies whether an email is junk email (coded as 1) or not (coded as 0). The data were collected by Hewlett-Packard Labs and donated by George Forman. The data set, which is specified in the following DATA step, contains 4,601 observations, with 2 binary variables and 57 continuous explanatory variables. The response variable, Class, is a binary indicator of whether an email is considered spam or not. The partitioning variable, Test, is a binary indicator that is used to divide the data into training and testing sets. The 57 explanatory variables are continuous variables that represent frequencies of some common words and characters and lengths of uninterrupted sequences of capital letters in emails.

```
data mylib.JunkMail;
   set Sashelp.JunkMail;
run;
```

In the following program, the PARTITION statement divides the data into two parts. The training data have a Test value of 0 and contain about two-thirds of the data; the rest of the data are used to evaluate the fit. A forward selection method selects the best model based on the training observations.

```
proc logselect data=mylib.JunkMail;
   model Class(event='1')=Make Address All _3d Our Over Remove Internet Order
         Mail Receive Will People Report Addresses Free Business Email You
         Credit Your Font _000 Money HP HPL George _650 Lab Labs Telnet _857
         Data _415 _85 Technology _1999 Parts PM Direct CS Meeting Original
         Project RE Edu Table Conference Semicolon Paren Bracket Exclamation
         Dollar Pound CapAvg CapLong CapTotal;
   partition rolevar=Test(train='0' test='1');
   selection method=forward;
run;
```

Selected results from the analysis are shown in Output 18.3.1 and Output 18.3.2.

The "Number of Observations" and "Response Profile" tables in Output 18.3.1 are divided into training and testing columns.

Output 18.3.1 Partitioned Counts

The LOGSELECT Procedure

Number of Observations

Description	Total	Training	Testing
Number of Observations Read	4601	3065	1536
Number of Observations Used	4601	3065	1536

Response Profile

Ordered Value	Class	Total Frequency	Training	Testing
1	0	2788	1847	941
2	1	1813	1218	595

Probability modeled is Class = 1.

The likelihood-based fit statistics for the selected model are displayed in the "Fit Statistics" table in Output 18.3.2, which has columns for the training and testing subsets.

Output 18.3.2 Partitioned Fit Statistics

Fit Statistics		
Description	**Training**	**Testing**
-2 Log Likelihood	1242.59491	823.68742
AIC (smaller is better)	1292.59491	873.68742
AICC (smaller is better)	1293.02268	874.54835
SBC (smaller is better)	1443.28998	1007.11085
Average Square Error	0.05659	0.06353
-2 Log L (Intercept-only)	4118.98701	2050.73514
R-Square	0.60877	0.55016
Max-rescaled R-Square	0.82359	0.74661
McFadden's R-Square	0.69833	0.59835
Misclassification Rate	0.07471	0.07813
Difference of Means	0.75122	0.73431

These statistics are computed for both the training and testing data. The statistics include the likelihood-based R-square statistics as well as several prediction-based statistics that are described in the section "Model Fit and Assessment Statistics" on page 1020. The ASE, the misclassification rate, and the difference of means should be very similar between the two groups when the training data are representative of the testing data; for this model, the values of these statistics seem similar between the two disjoint subsets.

If you want to display all the fit statistics without partitioning your data table, you can specify the PARTFIT option as follows:

```
proc logselect data=mylib.JunkMail(where Test=0) partfit;
   model Class(event='1')= Our Over Remove Internet Order Will
         Free Business You Your Font _000 Money HP George Parts
         Meeting RE Edu Semicolon Exclamation Dollar CapAvg
         CapLong;
run;
```

The resulting "Fit Statistics" table is identical to the Training column in Output 18.3.2.

Example 18.4: Ordinal Logistic Regression

(View the complete code for this example (logex4.sas) in the example repository.)

Consider a study of the effects of various cheese additives on taste. Researchers tested four cheese additives and obtained 52 response ratings for each additive. Each response was measured on a scale of nine categories ranging from strong dislike (1) to excellent taste (9). The data, given in McCullagh and Nelder (1989, p. 175) in the form of a two-way frequency table of additive by rating, are saved in the data table mylib.Cheese by using the following program. The variable y contains the response rating. The variable Additive specifies the cheese additive (1, 2, 3, or 4). The variable freq gives the frequency with which each additive received each rating.

```
data mylib.Cheese;
   do Additive = 1 to 4;
      do y = 1 to 9;
         input freq @@;
         output;
      end;
   end;
   label y='Taste Rating';
   datalines;
0  0  1  7  8  8 19  8  1
6  9 12 11  7  6  1  0  0
1  1  6  8 23  7  5  1  0
0  0  0  1  3  7 14 16 11
;
```

The response variable y is ordinally scaled. A cumulative logit model is used to investigate the effects of the cheese additives on taste. The following statements invoke PROC LOGSELECT to fit this model with y as the response variable and three indicator variables as explanatory variables, with the fourth additive as the reference level. With this parameterization, each Additive parameter compares an additive to the fourth additive.

```
proc logselect data=mylib.Cheese;
   freq freq;
   class Additive(ref='4') / param=ref;
   model y=Additive;
run;
```

Results from the logistic analysis are shown in Output 18.4.1 through Output 18.4.3.

The "Response Profile" table in Output 18.4.1 shows that the strong dislike (y=1) end of the rating scale is associated with lower OrderedValue values in the "Response Profile" table; hence the probability of disliking the additives is modeled.

Output 18.4.1 Proportional Odds Model Regression Analysis

The LOGSELECT Procedure

Model Information	
Data Source	CHEESE
Response Variable	y
Number of Response Levels	9
Frequency Variable	freq
Distribution	Multinomial
Link Type	Cumulative
Link Function	Logit
Optimization Technique	Newton-Raphson with Ridging

Number of Observations Read	36
Number of Observations Used	28
Sum of Frequencies Read	208
Sum of Frequencies Used	208

Output 18.4.1 *continued*

Response Profile		
Ordered Value	y	Total Frequency
1	1	7
2	2	10
3	3	19
4	4	27
5	5	41
6	6	28
7	7	39
8	8	25
9	9	12

Probabilities modeled are cumulated over the lower Ordered Values.

Class Level Information		
Class	Levels	Values
Additive	4	1 2 3 4

Output 18.4.2 Proportional Odds Model Regression Analysis

Convergence criterion (GCONV=1E-8) satisfied.

Dimensions	
Columns in Design	11
Number of Effects	2
Max Effect Columns	8
Rank of Design	11
Parameters in Optimization	11

Testing Global Null Hypothesis: BETA=0			
Test	DF	Chi-Square	Pr > ChiSq
Likelihood Ratio	3	148.4539	<.0001

Fit Statistics	
-2 Log Likelihood	711.34790
AIC (smaller is better)	733.34790
AICC (smaller is better)	734.69484
SBC (smaller is better)	770.06082

The positive value (1.6128) for the parameter estimate for Additive=1 in Output 18.4.3 indicates a tendency toward the lower-numbered categories of the first cheese additive relative to the fourth. In other words, the fourth additive tastes better than the first additive. Similarly, the second and third additives are both less favorable than the fourth additive. The relative magnitudes of these slope estimates imply the preference ordering: fourth, first, third, second.

Output 18.4.3 Proportional Odds Model Regression Analysis

Parameter Estimates

Parameter	y	DF	Estimate	Standard Error	Chi-Square	Pr > ChiSq
Intercept	1	1	-7.080166	0.564010	157.5844	<.0001
Intercept	2	1	-6.024980	0.476431	159.9230	<.0001
Intercept	3	1	-4.925416	0.425651	133.8992	<.0001
Intercept	4	1	-3.856801	0.388022	98.7968	<.0001
Intercept	5	1	-2.520552	0.345268	53.2940	<.0001
Intercept	6	1	-1.568538	0.312208	25.2408	<.0001
Intercept	7	1	-0.066875	0.273819	0.0596	0.8071
Intercept	8	1	1.492974	0.335696	19.7794	<.0001
Additive 1		1	1.612791	0.380544	17.9617	<.0001
Additive 2		1	4.964640	0.476721	108.4546	<.0001
Additive 3		1	3.322683	0.421830	62.0444	<.0001

Multinomial-Trial Syntax for Grouped Data

These data can be grouped according to the value of the Additive variable in the following DATA step:

```
data mylib.Cheese;
   do Additive = 1 to 4;
      input y1-y9;
      output;
   end;
   datalines;
0  0  1  7  8  8 19  8  1
6  9 12 11  7  6  1  0  0
1  1  6  8 23  7  5  1  0
0  0  0  1  3  7 14 16 11
;
```

The analysis proceeds as before with multinomial-trial syntax:

```
proc logselect data=mylib.Cheese;
   class Additive(ref='4') / param=ref;
   model y1-y9=Additive;
run;
```

Parameter estimates and tests are the same as for the preceding analysis and are not displayed here. The tables displayed in Output 18.4.4 and Output 18.4.5 are different from those produced by the preceding analysis: the "Model Information" and "Response Profile" tables include the nine response variables, there are only four observations in the data, and the fit statistics are modified by the inclusion of the constant term in the log likelihood.

Output 18.4.4 Information Tables with Multinomial-Trial Syntax

The LOGSELECT Procedure

Model Information	
Data Source	CHEESE
Response Variable	y1
Response Variable	y2
Response Variable	y3
Response Variable	y4
Response Variable	y5
Response Variable	y6
Response Variable	y7
Response Variable	y8
Response Variable	y9
Number of Response Levels	9
Distribution	Multinomial
Link Type	Cumulative
Link Function	Logit
Optimization Technique	Newton-Raphson with Ridging

Number of Observations Read	4
Number of Observations Used	4

Response Profile		
Ordered Value	**Response**	**Total Frequency**
1	**y1**	7
2	**y2**	10
3	**y3**	19
4	**y4**	27
5	**y5**	41
6	**y6**	28
7	**y7**	39
8	**y8**	25
9	**y9**	12

Probabilities modeled are cumulated over the lower Ordered Values.

Output 18.4.5 Fit Statistics with Multinomial-Trial Syntax

Fit Statistics	
-2 Log Likelihood	95.33994
AIC (smaller is better)	117.33994
AICC (smaller is better)	118.68687
SBC (smaller is better)	154.05285

Example 18.5: Odds Ratios

(View the complete code for this example (logex5.sas) in the example repository.)

```
title 'Example 5: Odds Ratios';
data mylib.getStarted;
   input C$ y x1-x10;
   datalines;
D  0  10.2  6  1.6  38  15  2.4  20  0.8  8.5  3.9
F  1  12.2  6  2.6  42  61  1.5  10  0.6  8.5  0.7
D  1   7.7  1  2.1  38  61    1  90  0.6  7.5  5.2
J  1  10.9  7  3.5  46  42  0.3   0  0.2    6  3.6
E  0  17.3  6  3.8  26  47  0.9  10  0.4  1.5  4.7
A  0  18.7  4  1.8   2  34  1.7  80    1  9.5  2.2
B  0   7.2  1  0.3  48  61  1.1  10  0.8  3.5    4
D  0   0.1  3  2.4   0  65  1.6  70  0.8  3.5  0.7
H  1   2.4  4  0.7  38  22  0.2  20    0    3  4.2
J  0  15.6  7  1.4   0  98  0.3   0    1    5  5.2
J  0  11.1  3  2.4  42  55  2.2  60  0.6  4.5  0.7
F  0     4  6  0.9   4  36  2.1  30  0.8    9  4.6
A  0   6.2  2  1.8  14  79  1.1  70  0.2    0  5.1
H  0   3.7  3  0.8  12  66  1.3  40  0.4  0.5  3.3
A  1   9.2  3  2.3  48  51  2.3  50    0    6  5.4
G  0    14  3    2  18  12  2.2   0    0    3  3.4
E  1  19.5  6  3.7  26  81  0.1  30  0.6    5  4.8
C  0    11  3  2.8  38   9  1.7  50  0.8  6.5  0.9
I  0  15.3  7  2.2  20  98  2.7 100  0.4    7  0.8
H  1   7.4  4  0.5  28  65  1.3  60  0.2  9.5  5.4
F  0  11.4  2  1.4  42  12  2.4  10  0.4    1  4.5
C  1  19.4  1  0.4  42   4  2.4  10    0  6.5  0.1
G  0   5.9  4  2.6  12  57  0.8  50  0.4    2  5.8
G  1  15.8  6  3.7  34   8  1.3  90  0.6  2.5  5.7
I  0    10  3  1.9  16  80    3  90  0.4  9.5  1.9
E  0  15.7  1  2.7  32  25  1.7  20  0.2  8.5    6
G  0    11  5  2.9  48  53  0.1  50    1  3.5  1.2
J  1  16.8  0  0.9  14  86  1.4  40  0.8    9    5
D  1    11  4  3.2  48  63  2.8  90  0.6    0  2.2
J  1   4.8  7  3.6  24   1  2.2  20    1  8.5  0.5
J  1  10.4  5    2  42  56    1  20    0  3.5  4.2
G  0  12.7  7  3.6   8  56  2.1  70    1  4.5  1.5
G  0   6.8  1  3.2  30  27  0.6   0  0.8    2  5.6
E  0   8.8  0  3.2   2  67  0.7  10  0.4    1    5
I  1   0.2  0  2.9  10  41  2.3  60  0.2    9  0.3
J  1   4.6  7  3.9  50  61  2.1  50  0.4    3  4.9
J  1   2.3  2  3.2  36  98  0.1  40  0.6  4.5  4.3
I  0  10.8  3  2.7  28  58  0.8  80  0.8    3    6
B  0   9.3  2  3.3  44  44  0.3  50  0.8  5.5  0.4
F  0   9.2  6  0.6   4  64  0.1   0  0.6  4.5  3.9
D  0   7.4  0  2.9  14   0  0.2  30  0.8  7.5  4.5
G  0  18.3  3  3.1   8  60  0.3  60  0.2    7  1.9
F  0   5.3  4  0.2  48  63  2.3  80  0.2    8  5.2
C  0   2.6  5  2.2  24   4  1.3  20    0    2  1.4
F  0  13.8  4  3.6   4   7  1.1  10  0.4  3.5  1.9
```

```
B  1  12.4  6  1.7  30  44  1.1  60  0.2    6  1.5
I  0   1.3  1  1.3   8  53  1.1  70  0.6    7  0.8
F  0  18.2  7  1.7  26  92  2.2  30    1  8.5  4.8
J  0   5.2  2  2.2  18  12  1.4  90  0.8    4  4.9
G  1   9.4  2  0.8  22  86  0.4  30  0.4    1  5.9
J  1  10.4  2  1.7  26  31  2.4  10  0.2    7  1.6
J  0    13  1  1.8  14  11  2.3  50  0.6  5.5  2.6
A  0  17.9  4  3.1  46  58  2.6  90  0.6  1.5  3.2
D  1  19.4  6    3  20  50  2.8 100  0.2    9  1.2
I  0  19.6  3  3.6  22  19  1.2   0  0.6    5  4.1
I  1     6  2  1.5  30  30  2.2  20  0.4  8.5  5.3
G  0  13.8  1  2.7   0  52  2.4  20  0.8    6    2
B  0  14.3  4  2.9  30  11  0.6  90  0.6  0.5  4.9
E  0  15.6  0  0.4  38  79  0.4  80  0.4    1  3.3
D  0    14  2    1  22  61    3  90  0.6    2  0.1
C  1   9.4  5  0.4  12  53  1.7  40    0    3  1.1
H  0  13.2  1  1.6  40  15  0.7  40  0.2    9  5.5
A  0  13.5  5  2.4  18  89  1.6  20  0.4  9.5  4.7
E  0   2.6  4  2.3  38   6  0.8  20  0.4    5  5.3
E  0  12.4  3  1.3  26   8  2.8  10  0.8    6  5.8
D  0   7.6  2  0.9  44  89  1.3  50  0.8    6  0.4
I  0  12.7  1  2.3  42   6  2.4  10  0.4    1    3
C  1  10.7  4  3.2  28  23  2.2  90  0.8  5.5  2.8
H  0  10.1  2  2.3  10  62  0.9  50  0.4  2.5  3.7
C  1  16.6  1  0.5  12  88  0.1  20  0.6  5.5  1.8
I  1   0.2  3  2.2   8  71  1.7  80  0.4  0.5  5.5
C  0  10.8  4  3.5  30  70  2.3  60  0.4  4.5  5.9
F  0   7.1  4    3  14  63  2.4  70    0    7  3.1
D  0  16.5  1  3.3  30  80  1.6  40    0  3.5  2.7
H  0  17.1  7  2.1  30  45  1.5  60  0.6  0.5  2.8
D  0   4.3  1  1.5  24  44    0  70    0    5  0.5
H  0    15  2  0.2  14  87  1.8  50    0  4.5  4.7
G  0  19.7  3  1.9  36  99  1.5  10  0.6    3  1.7
H  1   2.8  6  0.6  34  21    2  60    1    9  4.7
G  0  16.6  3  3.3  46   1  1.4  70  0.6  1.5  5.3
E  0  11.7  5  2.7  48   4  0.9  60  0.8  4.5  1.6
F  0  15.6  3  0.2   4  79  0.5   0  0.8  1.5  2.9
C  1   5.3  6  1.4   8  64    2  80  0.4    9  4.2
B  1   8.1  7  1.7  40  36  1.4  60  0.6    6  3.9
I  0  14.8  2  3.2   8  37  0.4  10    0  4.5    3
D  0   7.4  4    3  12   3  0.6  60  0.6    7  0.7
D  0   4.8  3  2.3  44  41  1.9  60  0.2    3  3.1
A  0   4.5  0  0.2   4  48  1.7  80  0.8    9  4.2
D  0   6.9  6  3.3  14  92  0.5  40  0.4  7.5    5
B  0   4.7  4  0.9  14  99  2.4  80    1  0.5  0.7
I  1   7.5  4  2.1  20  79  0.4  40  0.4  2.5  0.7
C  0   6.1  0  1.4  38  18  2.3  60  0.8  4.5  0.7
C  0  18.3  1    1  26  98  2.7  20    1  8.5  0.5
F  0  16.4  7  1.2  32  94  2.9  40  0.4  5.5  2.1
I  0   9.4  2  2.3  32  42  0.2  70  0.4  8.5  0.3
F  1  17.9  4  1.3  32  42    2  40  0.2    1  5.4
H  0  14.9  3  1.6  36  74  2.6  60  0.2    1  2.3
C  0  12.7  0  2.6   0  88  1.1  80  0.8  0.5  2.1
F  0   5.4  4  1.5   2   1  1.8  70  0.4  5.5  3.6
```

```
J  1  12.1  4  1.8  20  59  1.3  60  0.4    3  3.8
;

data mylib.Cheese;
   do Additive = 1 to 4;
      input y1-y9;
      output;
   end;
   datalines;
0  0  1  7  8  8 19  8  1
6  9 12 11  7  6  1  0  0
1  1  6  8 23  7  5  1  0
0  0  0  1  3  7 14 16 11
;
```

This example uses data that are examined in the section "Getting Started: LOGSELECT Procedure" on page 964 and in Example 18.4 to demonstrate how you can use the ODDSRATIO statement to produce and customize odds ratios.

In the following code, the ODDSRATIO statement is specified with no options to produce odds ratios for all the main effects as shown in Output 18.5.1:

```
proc logselect data=mylib.getStarted;
   model y(event='1') = x2 x5;
   oddsratio;
run;
```

Output 18.5.1 Odds Ratios or Main-Effects Model

Example 5: Odds Ratios

The LOGSELECT Procedure

Odds Ratios					
Description	Estimate	95% Wald Confidence Limits		p-Value	Percent Change
x2	1.233	0.999	1.521	0.0507	23.31
x5	1.002	0.987	1.016	0.8381	0.15

From Output 18.5.1, a subject whose X2 value is one unit larger (than its current value) has 1.233 times the odds of having a Y = 1 response, and a subject whose X5 value is one unit larger has 1.002 times the odds.

If your model has interactions, then the odds ratio computations depend on the values of the interacting variables. The following code produces odds ratios for the main effects, but because of interactions, these odds ratios depend on the values of the interacting variables as displayed in Output 18.5.2:

```
proc logselect data=mylib.getStarted;
   model y(event='1') = x2|x5;
   oddsratio;
run;
```

Output 18.5.2 Odds Ratios with Interactions

Example 5: Odds Ratios

The LOGSELECT Procedure

Odds Ratios					
Description	Estimate	95% Wald Confidence Limits		p-Value	Percent Change
x2 at x5=49.6	1.302	1.033	1.641	0.0252	30.24
x5 at x2=3.41	1.004	0.989	1.020	0.5843	0.44

The odds ratios shown in Output 18.5.2 are computed at the mean values of the interacting variables as computed from the data used to fit the model. You can change these values by specifying an AT option for each individual odds ratio, as follows. The output is shown in Output 18.5.3.

```
proc logselect data=mylib.getStarted;
   model y(event='1') = x2|x5;
   oddsratio x2(at(x5=1)) x5(at(x2=3));
run;
```

Output 18.5.3 Odds Ratios with the AT Option

Example 5: Odds Ratios

The LOGSELECT Procedure

Odds Ratios					
Description	Estimate	95% Wald Confidence Limits		p-Value	Percent Change
x2 at x5=1	1.925	1.160	3.193	0.0112	92.48
x5 at x2=3	1.008	0.991	1.025	0.3648	0.77

Alternatively, you can obtain the same results shown in Output 18.5.3 by specifying the following AT global option:

```
proc logselect data=mylib.getStarted;
   model y(event='1') = x2|x5;
   oddsratio x2 x5 / at(x5=1 x2=3);
run;
```

When odds ratios are computed for continuous variables, by default the odds ratio compares the odds computed at the mean of the continuous variable to the odds computed one unit higher than the mean. However, it might be more interesting to compare larger differences; for example, if you are comparing ages, a 10-year difference might be more interesting than a 1-year difference. You can customize the odds ratios accordingly by specifying the desired unit difference as an individual option or as a global option. In the following code, the global options set the units for X2 to 2 and 3 and for X5 to 4. However, the individual odds ratio unit option for X2 overrides the global option value and sets the units for X2 to 5.

```
proc logselect data=mylib.getStarted;
   model y(event='1') = x2|x5;
   oddsratio x2(unit=5) x5 / unit(x2=2 3 x5=4);
run;
```

Output 18.5.4 Odds Ratios with the UNIT Option

Example 5: Odds Ratios

The LOGSELECT Procedure

Odds Ratios					
Description	Estimate	95% Wald Confidence Limits		p-Value	Percent Change
x2 units=5 at x5=49.6	3.747	1.179	11.911	0.0252	274.72
x5 units=4 at x2=3.41	1.018	0.956	1.084	0.5843	1.77

When odds ratios are computed for classification variables, by default the odds ratio compares the odds computed at each level of the classification variable to the odds computed at the reference level. For example, consider the ordinal response data from Example 18.4. The following program displays the default odds ratios in Output 18.5.5:

```
proc logselect data=mylib.Cheese;
   class Additive(ref='4') / param=ref;
   model y1-y9=Additive;
   oddsratio;
run;
```

Output 18.5.5 Odds Ratios for Ordinal Response Model

Example 5: Odds Ratios

The LOGSELECT Procedure

Odds Ratios					
Description	Estimate	95% Wald Confidence Limits		p-Value	Percent Change
Additive 1 vs 4	5.017	2.380	10.577	<.0001	401.68
Additive 2 vs 4	143.257	56.277	364.671	<.0001	14225.70
Additive 3 vs 4	27.735	12.133	63.399	<.0001	2673.47

From the first line in Output 18.5.5, subjects who eat cheese with Additive = 1 have 5.017 times the odds to dislike it (Y1 end of the scale) than to like it (Y9 end of the scale) than subjects who eat cheese with Additive = 4.

By default, the odds ratios shown in Output 18.5.5 compare each level of Additive to its reference level, Additive = 4. Output 18.5.6 compares all pairs of Additive levels by specifying the DIFF=ALL global option in the following program:

```
proc logselect data=mylib.Cheese;
   class Additive(ref='4') / param=ref;
   model y1-y9=Additive;
   oddsratio / diff=all;
run;
```

Output 18.5.6 Odds Ratios with the DIFF=ALL Option

Example 5: Odds Ratios

The LOGSELECT Procedure

Odds Ratios					
Description	Estimate	95% Wald Confidence Limits		p-Value	Percent Change
Additive 1 vs 2	0.035	0.015	0.081	<.0001	-96.50
Additive 1 vs 3	0.181	0.087	0.375	<.0001	-81.91
Additive 2 vs 3	5.165	2.475	10.778	<.0001	416.53
Additive 1 vs 4	5.017	2.380	10.577	<.0001	401.68
Additive 2 vs 4	143.257	56.277	364.671	<.0001	14225.70
Additive 3 vs 4	27.735	12.133	63.399	<.0001	2673.47

Example 18.6: Binary Model with Repeated Measures

(View the complete code for this example (logex6.sas) in the example repository.)

This example uses data from the "Six Cities" study of the effects of air pollution on health (Ware et al. 1984). The analysis concerns the 16 selected cases in Lipsitz et al. (1994). The response is binary and indicates the wheezing status of 16 children. The children were assessed at ages 9, 10, 11, and 12 years. A value of 1 for wheezing status indicates the occurrence of wheezing at that age. The following statements create the data set:

```
data mylib.six;
     input case city$ @@;
   do i=1 to 4;
      input age smoke wheeze @@;
      output;
   end;
   datalines;
 1 portage   9 0 1  10 0 1  11 0 1  12 0 0
 2 kingston  9 1 1  10 2 1  11 2 0  12 2 0
 3 kingston  9 0 1  10 0 0  11 1 0  12 1 0
 4 portage   9 0 0  10 0 1  11 0 1  12 1 0
 5 kingston  9 0 0  10 1 0  11 1 0  12 1 0
 6 portage   9 0 0  10 1 0  11 1 0  12 1 0
 7 kingston  9 1 0  10 1 0  11 0 0  12 0 0
 8 portage   9 1 0  10 1 0  11 1 0  12 2 0
 9 portage   9 2 1  10 2 0  11 1 0  12 1 0
10 kingston  9 0 0  10 0 0  11 0 0  12 1 0
11 kingston  9 1 1  10 0 0  11 0 1  12 0 1
12 portage   9 1 0  10 0 0  11 0 0  12 0 0
13 kingston  9 1 0  10 0 1  11 1 1  12 1 1
14 portage   9 1 0  10 2 0  11 1 0  12 2 1
15 kingston  9 1 0  10 1 0  11 1 0  12 2 1
16 portage   9 1 1  10 1 1  11 2 0  12 1 0
;
```

The mean response is modeled by using a logistic regression model. The explanatory variables are city of residence (city), age in years (age), and maternal smoking status at each age (smoke). The responses for each child are assumed to be equally correlated, indicating that an exchangeable correlation structure is appropriate. The following statements fit the model by using generalized estimating equations (GEEs) and the chosen correlation structure:

```
proc logselect data=mylib.six;
   class case city;
   model wheeze(event='1') = city age smoke;
   repeated subject=case / type=exch covb corrw printmle;
run;
```

The MODEL statement specifies the mean model for the wheeze response variable as a logistic regression that uses city, age, and smoke as explanatory variables. This is the same specification that you use for an ordinary logistic regression.

The REPEATED statement uses GEE for parameter estimation, specifies the working correlation structure, and controls the output that is displayed. The SUBJECT=CASE option specifies that the case variable identifies the individual subjects in the data set. The case variable must be listed in the CLASS statement. The TYPE=EXCH option specifies that the procedure use the exchangeable working correlation structure. The exchangeable correlation structure does not rely on an order in the repeated measurements, so you do not need to specify a within-subject effect in the WITHIN= option. The COVB option displays the parameter estimates covariance matrix, and the CORRW option displays the final working correlation matrix. The PRINTMLE option displays a table of the initial parameter estimates that are produced by maximum likelihood estimation.

The initial parameter estimates for the GEE fit are computed by an ordinary logistic regression, which assumes no dependence between the repeated measurements.

Figure 18.6.1 displays general information about the GEE model fit.

Output 18.6.1 GEE Model Information

The LOGSELECT Procedure

GEE Model Information	
Correlation Structure	Exchangeable
Subject Effect	case (16 levels)
Number of Clusters	16
Correlation Matrix Dimension	4
Maximum Cluster Size	4
Minimum Cluster Size	4

Figure 18.6.2 displays the estimated parameter covariance matrices. The COVB option produces both the model-based and empirical covariance matrices.

Output 18.6.2 GEE Parameter Estimate Covariance Matrices

GEE Parameter Estimates
Model-Based Covariance Matrix

	Intercept	city kingston	age	smoke
Intercept	5.7495	-0.2226	-0.5347	0.01655
city kingston	-0.2226	0.4548	-0.00241	0.01876
age	-0.5347	-0.00241	0.05300	-0.01658
smoke	0.01655	0.01876	-0.01658	0.1910

GEE Parameter Estimates
Empirical Covariance Matrix

	Intercept	city kingston	age	smoke
Intercept	9.3399	-0.8510	-0.8325	-0.1653
city kingston	-0.8510	0.4737	0.05736	0.04023
age	-0.8325	0.05736	0.07778	-0.00236
smoke	-0.1653	0.04023	-0.00236	0.1305

The working correlation matrix that is specified by the CORRW option is displayed in Figure 18.6.3.

Output 18.6.3 GEE Working Correlation Matrix

Working Correlation Matrix

	Col1	Col2	Col3	Col4
Row1	1.0000	0.1648	0.1648	0.1648
Row2	0.1648	1.0000	0.1648	0.1648
Row3	0.1648	0.1648	1.0000	0.1648
Row4	0.1648	0.1648	0.1648	1.0000

The exchangeable working correlation structure is characterized by a single parameter. Figure 18.6.4 shows the table that contains this information.

Output 18.6.4 GEE Exchangeable Correlation

GEE Exchangeable Correlation

Correlation	0.164816

Figure 18.6.5 displays the "GEE Parameter Estimates" table. This table shows the parameter estimates, standard errors, chi-square values, and *p*-values for the parameter estimates. The table uses the empirical standard error estimates. You can also produce a table that uses the model-based standard error estimates by specifying the MODELSE option in the REPEATED statement.

Output 18.6.5 GEE Parameter Estimates Table

Analysis of GEE Parameter Estimates
Using Empirical Standard Error Estimates

Parameter	DF	Estimate	Standard Error	Chi-Square	Pr > ChiSq
Intercept	1	1.2751	3.0561	0.1741	0.6765
city kingston	1	0.1223	0.6882	0.0316	0.8589
city portage	0	0	.	.	.
age	1	-0.2036	0.2789	0.5327	0.4655
smoke	1	-0.0935	0.3613	0.0670	0.7957

You can compare the GEE parameter estimates to the parameter estimates from the initial model. The parameter estimates from the initial model are displayed in Figure 18.6.6.

Output 18.6.6 Initial Parameter Estimates Table

Analysis of Initial Parameter Estimates

Parameter	DF	Estimate	Standard Error	Chi-Square	Pr > ChiSq
Intercept	1	1.259707	2.610423	0.2329	0.6294
city kingston	1	0.139098	0.552686	0.0633	0.8013
city portage	0	0	.	.	.
age	1	-0.200315	0.250815	0.6379	0.4245
smoke	1	-0.128364	0.410239	0.0979	0.7544

The GEE parameter estimates are not substantially different from the parameter estimates in the initial model.

The GEE parameter estimates in Figure 18.6.5 and the empirical covariance matrix in Figure 18.6.2 are used for subsequent processing by the ASSOCIATION, CTABLE, and LACKFIT options and the ODDSRATIO and OUTPUT statements.

Example 18.7: Restoring a Stored Model

(View the complete code for this example (logex7.sas) in the example repository.)

The Sashelp.JunkMail data set from Example 18.3 is used here to show how to store a model and later how to restore that same model for further processing. The following program fits a model by using a SELECTION statement, then stores the model by using a STORE statement. The NOTE= option in the STORE statement stores a few notes along with the model; you might also want to store the submitted program. The DISPLAY statement suppresses all output.

```
data mylib.JunkMail;
   set Sashelp.JunkMail;
run;

proc logselect data=mylib.JunkMail;
   model Class(event='1')=Make Address All _3d Our Over Remove Internet Order
         Mail Receive Will People Report Addresses Free Business Email You
         Credit Your Font _000 Money HP HPL George _650 Lab Labs Telnet _857
```

```
         Data _415 _85 Technology _1999 Parts PM Direct CS Meeting Original
         Project RE Edu Table Conference Semicolon Paren Bracket Exclamation
         Dollar Pound CapAvg CapLong CapTotal;
   display / excludeall;
   selection method=forward(maxstep=5);
   store mylib.mymodel /
         note="Generated with stepwise selection.";
run;
```

The following program replays the standard tables from the fitted model. You can also specify the STB, CLB, CORRB, COVB, and TYPE3 options and the CODE and ODDSRATIO statements, because they do not require access to data.

```
proc logselect restore=mylib.mymodel stb clb corrb covb type3;
   code;
   oddsratio;
run;
```

Information about the item store is displayed in Output 18.7.1. The Replay section is displayed in Output 18.7.2 through Output 18.7.5.

Output 18.7.1 Store Information

Store Information	
Name	MYMODEL
Data Source	JUNKMAIL
Data Libname	<<your library>>
Procedure	LOGSELECT
Action	regression.logistic
Action Version	1.0.6
Date	19May2023:14:02:33

Generated with stepwise selection.

The standard tables from PROC LOGSELECT are displayed next, beginning with the "Model Information" table in Output 18.7.2.

Output 18.7.2 Replay Section

The LOGSELECT Procedure

Replay

Model Information	
Response Variable	Class
Distribution	Binary
Link Function	Logit
Optimization Technique	Newton-Raphson with Ridging

Number of Observations Read	4601
Number of Observations Used	4601

Output 18.7.2 *continued*

Response Profile		
Ordered Value	Class	Total Frequency
1	0	2788
2	1	1813

Probability modeled is Class = 1.

Output 18.7.3 Replay Section (*continued*)

Convergence criterion (ABSGCONV=1E-7) satisfied.

Dimensions	
Columns in Design	6
Number of Effects	6
Max Effect Columns	1
Rank of Design	6
Parameters in Optimization	6

Testing Global Null Hypothesis: BETA=0			
Test	DF	Chi-Square	Pr > ChiSq
Likelihood Ratio	5	2677.9370	<.0001

The fit statistics shown in Output 18.7.4 are those that are computed when the original model is fit. If you specify the PARTFIT option or a PARTITION statement when the original model is fit, then this table always includes the additional statistics. However, specifying a PARTFIT option when you restore the model does not include the additional statistics in this table in the Replay section. Instead, the same table that appears in the Restore section is expanded; see Output 18.7.8.

Output 18.7.4 Replay Section (*continued*)

Fit Statistics	
-2 Log Likelihood	3492.21579
AIC (smaller is better)	3504.21579
AICC (smaller is better)	3504.23408
SBC (smaller is better)	3542.81997

Parameter Estimates								
Parameter	DF	Estimate	Standard Error	Chi-Square	Pr > ChiSq	95% Confidence Limits		Standardized Estimate
Intercept	1	-2.007045	0.057444	1220.7512	<.0001	-2.11963	-1.89446	
Remove	1	4.674731	0.371458	158.3777	<.0001	3.94669	5.40278	1.008867
Free	1	1.057249	0.093489	127.8902	<.0001	0.87401	1.24048	0.481347
Your	1	0.465818	0.035907	168.2990	<.0001	0.39544	0.53619	0.308391
_000	1	4.582674	0.461809	98.4719	<.0001	3.67754	5.48780	0.885020
Dollar	1	8.208586	0.631423	169.0034	<.0001	6.97102	9.44615	1.112771

Output 18.7.5 Replay Section (*continued*)

Parameter Estimates Covariance Matrix

Parameter	Intercept	Remove	Free	Your	_000	Dollar
Intercept	0.003300	-0.00318	-0.00128	-0.00094	-0.00300	-0.00847
Remove	-0.00318	0.1380	-0.00136	-0.00089	-0.00279	-0.00730
Free	-0.00128	-0.00136	0.008740	-0.00024	0.000223	-0.00156
Your	-0.00094	-0.00089	-0.00024	0.001289	-0.00025	-0.00225
_000	-0.00300	-0.00279	0.000223	-0.00025	0.2133	-0.05709
Dollar	-0.00847	-0.00730	-0.00156	-0.00225	-0.05709	0.3987

Parameter Estimates Correlation Matrix

Parameter	Intercept	Remove	Free	Your	_000	Dollar
Intercept	1.0000	-0.1488	-0.2393	-0.4550	-0.1131	-0.2335
Remove	-0.1488	1.0000	-0.0393	-0.0667	-0.0163	-0.0311
Free	-0.2393	-0.0393	1.0000	-0.0706	0.0052	-0.0264
Your	-0.4550	-0.0667	-0.0706	1.0000	-0.0149	-0.0994
_000	-0.1131	-0.0163	0.0052	-0.0149	1.0000	-0.1958
Dollar	-0.2335	-0.0311	-0.0264	-0.0994	-0.1958	1.0000

Output 18.7.6 Replay Section (*continued*)

Model Analysis of Variance (Type III)

Effect	DF	Chi-Square	Pr > ChiSq
Remove	1	158.3777	<.0001
Free	1	127.8902	<.0001
Your	1	168.2990	<.0001
_000	1	98.4719	<.0001
Dollar	1	169.0034	<.0001

Odds Ratios

Description	Estimate	95% Wald Confidence Limits		p-Value	Percent Change
_000	97.776	39.549	241.726	<.0001	9677.55
Dollar	>999.999	>999.999	>999.999	<.0001	367134.6
Free	2.878	2.397	3.457	<.0001	187.84
Remove	107.204	51.764	222.022	<.0001	10620.38
Your	1.593	1.485	1.709	<.0001	59.33

Output 18.7.7 Replay Timing

Task Timing		
Task	**Seconds**	**Percent**
Model Restoration	0.00	0.30%
Setup and Parsing	0.10	37.06%
Producing Score Code	0.00	1.27%
Type III Sums of Squares	0.00	0.99%
Display	0.00	1.70%
Cleanup	0.00	0.00%
Total	0.27	100.00%

If you specify the CTABLE, LACKFIT, ASSOCIATION, or PARTFIT option or the OUTPUT statement, then you must also specify an input data table. The following program requests these tables; the Replay section tables are produced but not displayed here.

```
proc logselect restore=mylib.mymodel data=mylib.JunkMail
   association ctable(cutpt=0.1 to 0.9 by 0.1) lackfit partfit;
   output out=mylib.out p cbar;
proc print data=mylib.out(obs=1);
run;
```

The first table that is displayed in the Restore section (Output 18.7.8) is the "Postfit Information" table, which contains the name of the input data table and information usually displayed in an "NObs" table. The next two tables are the "Response Profile" and "Fit Statistics" tables, which are computed from the JunkMail data.

Output 18.7.8 Restore Section

Results for JUNKMAIL

Postfit Information	
Postfit Data Source	JUNKMAIL
Number of Observations Read	4601
Number of Observations Used	4601

Response Profile		
Ordered Value	**Class**	**Total Frequency**
1	**0**	2788
2	**1**	1813

Probability modeled is Class = 1.

Output 18.7.8 *continued*

Fit Statistics	
-2 Log Likelihood	3492.21579
AIC (smaller is better)	3504.21579
AICC (smaller is better)	3504.23408
SBC (smaller is better)	3542.81997
Average Square Error	0.10819
-2 Log L (Intercept-only)	6170.15284
R-Square	0.44124
Max-rescaled R-Square	0.59754
McFadden's R-Square	0.43401
Misclassification Rate	0.14606
Difference of Means	0.52695

The "Association" and "Classification" tables in Output 18.7.9 also provide information that is computed from the input data table.

Output 18.7.9 Restore Section (*continued*)

Association of Predicted Probabilities and Observed Responses	
Concordance Index (AUC)	0.9125
Somers' D	0.8250
Gamma	0.8589
Tau-a	0.3940
Pairs	5054644
Percent Concordant	89.2764
Percent Discordant	6.7789
Percent Tied	3.9447

Classification Table				
	Correct		Incorrect	
Probability Level	**Event**	**Nonevent**	**Event**	**Nonevent**
0.1000	1813	0	2788	0
0.2000	1613	2230	558	200
0.3000	1483	2491	297	330
0.4000	1363	2598	190	450
0.5000	1289	2640	148	524
0.6000	1213	2675	113	600
0.7000	1132	2696	92	681
0.8000	1029	2722	66	784
0.9000	858	2750	38	955

The two Hosmer-Lemeshow tables whose content is computed from the input data are shown in Output 18.7.10.

Output 18.7.10 Restore Section (*continued*)

Partition for the Hosmer and Lemeshow Test					
Group	Total	Observed 1	Expected 1	Observed 0	Expected 0
1	1763	121	208.85	1642	1554.15
2	461	45	68.63	416	392.37
3	460	110	95.92	350	364.08
4	460	235	158.10	225	301.90
5	461	355	315.79	106	145.21
6	460	429	431.12	31	28.88
7	536	518	534.59	18	1.41

Hosmer and Lemeshow Goodness-of-Fit Test		
Chi-Square	DF	Pr > ChiSq
322.2997	5	<.0001

The first line of the output data table is displayed in Output 18.7.11. Note that the CBAR statistic is not produced because the computation requires the original input data table.

Output 18.7.11 First Observation from Output Data Table

Obs	_PRED_
1	0.22767

You can specify the FITDATA option to indicate that the DATA= table is the original input table so that computation of the CBAR statistic is valid. The first observation in the output data table is shown in Output 18.7.12 and displays the CBAR statistic.

```
proc logselect restore=mylib.mymodel data=mylib.JunkMail fitdata;
   output out=mylib.out p cbar;
proc print data=mylib.out(obs=1);
run;
```

Output 18.7.12 First Observation from Output Data Table with FITDATA Option

Obs	_PRED_	_CBAR_
1	0.22767	.001559475

Example 18.8: Modeling Microarray Data with Elastic Net

(View the complete code for this example (logex8.sas) in the example repository.)

This example uses data from a cancer study that attempts to determine whether a patient has type 1 leukemia (acute lymphoblastic leukemia) or type 2 leukemia (acute myeloid leukemia) on the basis of the patient's gene expression measurements (Golub et al. 1999; Zou and Hastie 2005). The data sets, which are specified in the following DATA steps, contain 38 observations for training and 34 observations for testing prediction accuracy. Each observation contains 7,129 continuous variables that are gene expression measurements. The response variable is a binary indicator that is coded as 1 for type 1 leukemia and –1 for type 2 leukemia. An additional role variable is added to the training data set so that a subset of training observations are used for computing the validation error.

```
data LeuTrain;
   set Sashelp.LeuTrain;
   call streamInit(123);
   if rand('UNIFORM')<0.6 then role='train'; else role='valid';
run;
data mylib.LeuTrain;
   set LeuTrain;
run;
data mylib.LeuTest;
   set Sashelp.LeuTest;
run;
```

This study tries to find genes that can help identify the leukemia type most effectively. Because the number of variables is much greater than the number of observations, and because many measurements are highly correlated, model selection methods that are based on maximum likelihood estimation are not suitable choices. In this case, penalized selection methods provide solutions that can achieve a better trade-off between bias and variance. In particular, elastic net selection inherits advantages from both LASSO selection, which can return sparse models, and ridge regression, which can handle wide data sets that contain highly correlated variables. The following program builds a predictive model by using the SELECTION statement with the elastic net method, then stores the model by using the STORE statement. The SELECTION statement uses the average square error that is computed from the validation data as the criterion to determine the best candidate. The PARTITION statement uses the role variable to divide the training data so that approximately 40% of the observations are used for validation.

```
proc logselect data=mylib.LeuTrain partfit;
   model y=x1-x7129;
   selection method=elasticnet(choose=validate);
   partition role=role(train='train' validate='valid');
   store mylib.ElasticNetModel / note="Microarray data with Elastic Net.";
run;
```

The "Selection Information" table in Output 18.8.1 displays information about the selection process. The maximum lambda value is the upper bound of the regularization parameter for the LASSO penalty such that there can be at least one nonzero regression parameter estimate before lambda reaches that value and all estimates reduce to zero when lambda increases further. There are a total of 20 regularization parameters that elastic net selection uses to fit candidate models. By default, elastic net uses a balanced LASSO and ridge penalty.

Output 18.8.1 Selection Information

Selection Information	
Selection Method	Elastic Net
Choose Criterion	Validation ASE
Maximum Lambda	0.763464
Lambda Steps	20
Regularization Balance	0.5

The "Selected Effects" table in Output 18.8.2 shows the genes that are selected from the total of 7,129 genes by the elastic net selection method.

Output 18.8.2 Selected Genes

Selected Effects:	Intercept x50 x136 x494 x1092 x1095 x1247 x1394 x1745 x1779 x1834 x1882 x2020 x2043 x2186 x2242 x2288 x2402 x3012 x3123 x3258 x3320 x3847 x4052 x4196 x4197 x4229 x4279 x4289 x4377 x4407 x4461 x4847 x5039 x5122 x5231 x5794 x5954 x6185 x6201 x6215 x6362 x6405 x6539 x6677 x6797 x6803 x6806 x6919 x6989

Because the data for assessing test error are in a separate data set, you can compute the assessment statistics by using the stored model in several ways. One way is to use the RESTORE option in the PROC LOGSELECT statement to load the selected model and compute values from the new data. The following statements restore the selected model mylib.ElasticNetModel and compute the test error from the test data set mylib.LeuTest. Output 18.8.3 displays the fit statistics, including the misclassification rate. Compared to other selection methods—including LASSO, which returns models that have the number of selected variables only up to the training sample size—elastic net selection can select more variables and thus provide better prediction accuracy.

```
proc logselect restore=mylib.ElasticNetModel data=mylib.LeuTest partfit;
run;
```

Output 18.8.3 Fit Statistics

Results for LEUTEST

Fit Statistics	
-2 Log Likelihood	12.83258
AIC (smaller is better)	112.83258
AICC (smaller is better)	5212.83258
SBC (smaller is better)	189.15060
Average Square Error	0.04512
-2 Log L (Intercept-only)	46.06962
R-Square	0.62377
Max-rescaled R-Square	0.84061
McFadden's R-Square	0.72145
Misclassification Rate	0.02941
Difference of Means	0.77283

Example 18.9: Risks

(View the complete code for this example (logex9.sas) in the example repository.)

The following data set comes from an environmental study of workers in a cotton factory. A similar data set is analyzed in Ibrahim, Chen, and Lipsitz (2001). A contingency table is created from the study, where 912 workers are checked for dyspnea (difficult or labored respiration) and their exposure to cotton dust is recorded. The variable y indicates whether the worker has developed dyspnea, expd indicates whether the worker was exposed to cotton dust, and count is the number of workers with that profile.

```
data mylib.d3;
   input y expd count;
   datalines;
0 0 349
0 1 284
1 0 118
1 1 161
;
```

The syntax for computing risk statistics is very similar to that for the ODDSRATIO statement, except that you can specify only global options; there are no options for modifying individual risks. In the following program, both the ODDSRATIO and RISK statements are specified with no risk variables, so by default the expd variable is analyzed. The E option displays the profiles that are used in the computations of the model-predicted probabilities, and the SHOWPROBS option displays those probabilities.

```
proc logselect data=mylib.d3;
   freq count;
   class expd(ref='0');
   model y(event='1')=expd;
   oddsratio;
   risk / e showprobs;
run;
```

The population profiles that you are comparing are displayed in Output 18.9.1. The rows of this table are in the same order as the rows in the "Parameter Estimates" table. Using the notation from the section "Risk Differences and Relative Risks" on page 1034, the profile $L_1 = (1, 1, 0)$ is multiplied by the parameter estimates $\boldsymbol{\beta} = (-1.084387, 0.516817, 0)$, and applying the inverse-logit function gives $p_1 = 0.361798$, as displayed in the P1 column of the "Risks" table in Output 18.9.2.

Output 18.9.1 Risk Coefficients

Coefficients for Risk Computations		
Parameter	**L1_1**	**L1_2**
Parm1	1	1
Parm2	1	
Parm3		1

In Output 18.9.2, both the odds ratio results and the risk results show that exposure to cotton dust is a significant factor in a worker's having dyspnea.

Output 18.9.2 Odds Ratio and Risk Difference

Odds Ratios

Description	Estimate	95% Wald Confidence Limits		p-Value	Percent Change
expd 1 vs 0	1.677	1.262	2.229	0.0004	67.67

Risk Differences

Description	Estimate	Standard Error	Chi-Square	Pr > ChiSq	95% Confidence Limits		P1	P2
expd 1 vs 0	0.109121	0.030385	12.898	0.0003	0.049568	0.168674	0.361798	0.252677

References

Agresti, A. (2013). *Categorical Data Analysis*. 3rd ed. Hoboken, NJ: John Wiley & Sons.

Akaike, H. (1974). "A New Look at the Statistical Model Identification." *IEEE Transactions on Automatic Control* AC-19:716–723.

Albert, A., and Anderson, J. A. (1984). "On the Existence of Maximum Likelihood Estimates in Logistic Regression Models." *Biometrika* 71:1–10.

Allison, P. D. (2012). *Logistic Regression Using SAS: Theory and Application*. 2nd ed. Cary, NC: SAS Institute Inc.

Brier, G. W. (1950). "Verification of Forecasts Expressed in Terms of Probability." *Monthly Weather Review* 78:1–3.

Burnham, K. P., and Anderson, D. R. (1998). *Model Selection and Inference: A Practical Information-Theoretic Approach*. New York: Springer-Verlag.

Collett, D. (2003). *Modelling Binary Data*. 2nd ed. London: Chapman & Hall.

Cook, R. D., and Weisberg, S. (1982). *Residuals and Influence in Regression*. New York: Chapman & Hall.

Cox, D. R., and Snell, E. J. (1989). *The Analysis of Binary Data*. 2nd ed. London: Chapman & Hall.

Diggle, P. J., Liang, K.-Y., and Zeger, S. L. (1994). *Analysis of Longitudinal Data*. Oxford: Clarendon Press.

Fagerland, M. W., and Hosmer, D. W. (2013). "A Goodness-of-Fit Test for the Proportional Odds Regression Model." *Statistics in Medicine* 32:2235–2249.

Fagerland, M. W., and Hosmer, D. W. (2016). "Tests for Goodness of Fit in Ordinal Logistic Regression Models." *Journal of Statistical Computation and Simulation* 86:3398–3418.

Fagerland, M. W., Hosmer, D. W., and Bofin, A. M. (2008). "Multinomial Goodness-of-Fit Tests for Logistic Regression Models." *Statistics in Medicine* 27:4238–4253.

Fleiss, J. L., Levin, B., and Paik, M. C. (2003). *Statistical Methods for Rates and Proportions*. 3rd ed. Hoboken, NJ: John Wiley & Sons.

Golub, T. R., Slonim, D. K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J. P., Coller, H., et al. (1999). "Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression." *Science* 286:531–537.

Gupta, A. K., Nguyen, T., and Pardo, L. (2008). "Residuals for Polytomous Logistic Regression Models Based on φ-Divergences Test Statistics." *Statistics* 42:495–514.

Hardin, J. W., and Hilbe, J. M. (2003). *Generalized Estimating Equations*. Boca Raton, FL: Chapman & Hall/CRC.

Hastie, T. J., Tibshirani, R. J., and Friedman, J. H. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd ed. New York: Springer-Verlag.

Hosmer, D. W., Jr., and Lemeshow, S. (2000). *Applied Logistic Regression*. 2nd ed. New York: John Wiley & Sons.

Hosmer, D. W., Jr., and Lemeshow, S. (2013). *Applied Logistic Regression*. 3rd ed. New York: John Wiley & Sons.

Hurvich, C. M., and Tsai, C.-L. (1989). "Regression and Time Series Model Selection in Small Samples." *Biometrika* 76:297–307.

Ibrahim, J. G., Chen, M.-H., and Lipsitz, S. R. (2001). "Missing Responses in Generalised Linear Mixed Models When the Missing Data Mechanism Is Nonignorable." *Biometrika* 88:551–564.

Lawless, J. F., and Singhal, K. (1978). "Efficient Screening of Nonnormal Regression Models." *Biometrics* 34:318–327.

Lesaffre, E., and Albert, A. (1989). "Multiple-Group Logistic Regression Diagnostics." *Journal of the Royal Statistical Society, Series C* 38:425–440.

Liang, K.-Y., and Zeger, S. L. (1986). "Longitudinal Data Analysis Using Generalized Linear Models." *Biometrika* 73:13–22.

Lipsitz, S. R., Fitzmaurice, G. M., Orav, E. J., and Laird, N. M. (1994). "Performance of Generalized Estimating Equations in Practical Situations." *Biometrics* 50:270–278.

Littell, R. C., Freund, R. J., and Spector, P. C. (1991). *SAS System for Linear Models*. 3rd ed. Cary, NC: SAS Institute Inc.

Magee, L. (1990). "R^2 Measures Based on Wald and Likelihood Ratio Joint Significant Tests." *American Statistician* 44:250–253.

Martín, N. (2015). "Using Cook's Distance in Polytomous Logistic Regression." *British Journal of Mathematical and Statistical Psychology* 68:84–115.

McCullagh, P., and Nelder, J. A. (1989). *Generalized Linear Models*. 2nd ed. London: Chapman & Hall.

McFadden, D. (1974). "Conditional Logit Analysis of Qualitative Choice Behavior." In *Frontiers in Econometrics*, edited by P. Zarembka, 105–142. New York: Academic Press.

McNicol, D. (2005). *A Primer of Signal Detection Theory*. Mahwah, NJ: Lawrence Erlbaum Associates.

Muller, K. E., and Fetterman, B. A. (2002). *Regression and ANOVA: An Integrated Approach Using SAS Software*. Cary, NC: SAS Institute Inc.

Murphy, A. H. (1973). "A New Vector Partition of the Probability Score." *Journal of Applied Meteorology* 12:595–600.

Nagelkerke, N. J. D. (1991). "A Note on a General Definition of the Coefficient of Determination." *Biometrika* 78:691–692.

Nelder, J. A., and Wedderburn, R. W. M. (1972). "Generalized Linear Models." *Journal of the Royal Statistical Society, Series A* 135:370–384.

Pan, W. (2001). "Akaike's Information Criterion in Generalized Estimating Equations." *Biometrics* 57:120–125.

Paul, P., Pennell, M. L., and Lemeshow, S. (2013). "Standardizing the Power of the Hosmer-Lemeshow Goodness of Fit Test in Large Data Sets." *Statistics in Medicine* 32:67–80.

Pepe, M. S. (2003). *The Statistical Evaluation of Medical Tests for Classification and Prediction*. New York: Oxford University Press.

Pregibon, D. (1981). "Logistic Regression Diagnostics." *Annals of Statistics* 9:705–724.

Preisser, J. S., and Qaqish, B. F. (1996). "Deletion Diagnostics for Generalised Estimating Equations." *Biometrika* 83:551–562.

Royall, R. M. (1986). "Model Robust Inference Using Maximum Likelihood Estimators." *International Statistical Review* 54:221–226.

Santner, T. J., and Duffy, D. E. (1986). "A Note on A. Albert and J. A. Anderson's Conditions for the Existence of Maximum Likelihood Estimates in Logistic Regression Models." *Biometrika* 73:755–758.

Schwarz, G. (1978). "Estimating the Dimension of a Model." *Annals of Statistics* 6:461–464.

Stokes, M. E., Davis, C. S., and Koch, G. G. (2012). *Categorical Data Analysis Using SAS*. 3rd ed. Cary, NC: SAS Institute Inc.

Tjur, T. (2009). "Coefficients of Determination in Logistic Regression Models—a New Proposal: The Coefficient of Discrimination." *American Statistician* 63:366–372.

Venzon, D. J., and Moolgavkar, S. H. (1988). "A Method for Computing Profile-Likelihood-Based Confidence Intervals." *Journal of the Royal Statistical Society, Series C* 37:87–94.

Ware, J. H., Dockery, S. A., III, Speizer, F. E., and Ferris, B. G., Jr. (1984). "Passive Smoking, Gas Cooking, and Respiratory Health of Children Living in Six Cities." *American Review of Respiratory Diseases* 129:366–374.

White, H. (1982). "Maximum Likelihood Estimation of Misspecified Models." *Econometrica* 50:1–25.

Williams, D. A. (1987). "Generalized Linear Model Diagnostics Using the Deviance and Single Case Deletions." *Journal of the Royal Statistical Society, Series C* 36:181–191.

Zeger, S. L., Liang, K.-Y., and Albert, P. S. (1988). "Models for Longitudinal Data: A Generalized Estimating Equation Approach." *Biometrics* 44:1049–1060.

Zou, H., and Hastie, T. (2005). "Regularization and Variable Selection via the Elastic Net." *Journal of the Royal Statistical Society, Series B* 67:301–320.

Chapter 19
MBC Procedure

Contents

Overview: MBC Procedure **1092**
Motivation for Model-Based Clustering 1092
PROC MBC Compared with Other Procedures 1093
Using CAS Sessions and CAS Engine Librefs 1093
Getting Started: MBC Procedure **1094**
Old Faithful Data 1094
Syntax: MBC Procedure **1099**
PROC MBC Statement 1099
BY Statement 1104
DISPLAY Statement 1104
DISPLAYOUT Statement 1105
INIT Statement 1106
OUTPUT Statement 1106
STORE Statement 1108
VAR Statement 1108
Details: MBC Procedure **1108**
Log-Likelihood Definitions 1108
Posterior Probabilities and Clustering 1109
Expectation-Maximization (EM) Algorithm 1109
Singular Covariance Matrices 1110
Covariance Structures 1110
Noise Component 1113
Missing Data 1113
Model Selection 1113
Number of Parameters 1114
Displayed Output 1115
ODS Table Names 1116
Examples: MBC Procedure **1117**
Example 19.1: Storing and Scoring 1117
Example 19.2: Adjusting Starting Values 1121
References **1127**

Overview: MBC Procedure

The MBC procedure fits mixtures of multivariate Gaussian and uniform distributions to achieve unsupervised and semisupervised clustering of data. It treats the cluster memberships as missing data and uses the expectation-maximization (EM) algorithm to maximize the likelihood. The procedure can produce output data tables that contain cluster membership weights and component log likelihoods. PROC MBC can also store the fitted model for scoring subsequent input data.

PROC MBC assumes that the data can be described by a mixture of multivariate Gaussian components and an optional noise component. The procedure also assumes that the data are continuous, and it does not support CLASS variables.

PROC MBC fits Gaussian mixtures by using two types of covariance structures:

- Nine covariance structures that have specific constraints in the covariance matrices across components. These are referred to as Gaussian mixture models.
- Eight covariance structures that are derived from factor models, with specific constraints in the factor loading matrices and unique error variance matrices across components. These are referred to as parsimonious Gaussian mixture models, or PGMMs (McNicholas and Murphy 2008). The term "parsimonious" indicates that using factor models can reduce the dimension of the data during modeling.

Motivation for Model-Based Clustering

The traditional goal of clustering has been to identify groups of observations that are similar by some measurement. Standard clustering methods achieve this primarily through heuristic methods, by using a measurement such as the distance to a center or a boundary. These methods appeal to intuition, but they cannot answer fundamental questions about the number of clusters, the suitability of certain cluster structures, or the handling of outliers. In addition, standard methods do not have well-defined statistical properties and so do not support formal inference.

In contrast, model-based clustering (MBC) models the observations by using a mixture of specific distributions. In this framework, the qualities of these clusters and cluster membership are parameters to be estimated by formal statistical methods.

The MBC procedure implements model-based clustering by using mixtures of multivariate Gaussian distributions. PROC MBC allows for a noise component and automatic model selection through the use of information criteria. In addition, it provides posterior scoring of new input data.

PROC MBC Compared with Other Procedures

SAS/STAT and SAS Viya offer several other clustering and mixture-modeling procedures.

PROC MBC is based on the model likelihood, whereas the clustering methods in the FASTCLUS, CLUSTER, KCLUS, MODECLUS, and FASTKNN procedures use a distance-based clustering criterion. The likelihood-based approach allows direct comparison of models through the use of information criteria.

This likelihood-based approach means that the MBC procedure can produce "soft" clusterings, which are based on posterior probabilities. These clusterings indicate relative strengths of affinity that each observation has for each cluster in the model. In contrast, the other clustering procedures produce "hard" clusterings, which associate each observation with a single cluster.

Like PROC MBC, the FMM and HPFMM procedures use a likelihood-based approach. PROC FMM and PROC HPFMM can fit several different univariate finite mixture models, but they cannot fit multivariate models. PROC MBC uses the expectation-maximization (EM) technique (Dempster, Laird, and Rubin 1977) to find parameter estimates, whereas the FMM and HPFMM procedures use traditional maximum likelihood and Bayesian techniques.

Using CAS Sessions and CAS Engine Librefs

SAS Cloud Analytic Services (CAS) is the analytic server and associated cloud services in SAS Viya. This section describes how to create a CAS session and set up a CAS engine libref that you can use to connect to the CAS session. It assumes that you have a CAS server already available; contact your system administrator if you need help starting and terminating a server. This CAS server is identified by specifying the host on which it runs and the port on which it listens for communications. To simplify your interactions with this CAS server, the host information and port information for the server are stored as SAS option values that are retrieved automatically whenever this CAS server needs to be accessed. You can examine the host and port values for the server at your site by using the following statements:

```
proc options option=(CASHOST CASPORT);
run;
```

In addition to starting a CAS server, your system administrator might also have created a CAS session and a CAS engine libref for your use. You can define your own sessions and CAS engine librefs that connect to the CAS server as shown in the following statements:

```
cas mysess;
libname mylib cas sessref=mysess;
```

The CAS statement creates the CAS session named mysess, and the LIBNAME statement creates the mylib CAS engine libref that you use to connect to this session. It is not necessary to explicitly name the CASHOST and CASPORT of the CAS server in the CAS statement, because these values are retrieved from the corresponding SAS option values.

If you have created the mysess session, you can terminate it by using the TERMINATE option in the CAS statement as follows:

```
cas mysess terminate;
```

For more information about the CAS and LIBNAME statements, see the section "Introduction to Shared Concepts" on page 6 in Chapter 2, "Shared Concepts."

Getting Started: MBC Procedure

Old Faithful Data

(View the complete code for this example at https://github.com/sascommunities/doc-supplement-casstat/tree/main/Examples/mbcgs1.sas.)

This example uses the Old Faithful data on geyser eruptions (Cook and Weisberg 1982) to illustrate a mixture of multivariate Gaussian components. This data set contains time before eruption and duration of eruption for Old Faithful geyser in Yellowstone National Park in the United States. The following DATA step creates the data table mylib.getStarted, which consists of 272 observations on the duration of each eruption and the wait time until the next eruption. Both duration and wait time are measured in minutes.

```
data mylib.getStarted;
  input Duration Wait @@;
  datalines;
3.600 79  1.800 54  3.333 74  2.283 62  4.533 85  2.883 55  4.700 88  3.600 85
1.950 51  4.350 85  1.833 54  3.917 84  4.200 78  1.750 47  4.700 83  2.167 52
1.750 62  4.800 84  1.600 52  4.250 79  1.800 51  1.750 47  3.450 78  3.067 69
4.533 74  3.600 83  1.967 55  4.083 76  3.850 78  4.433 79  4.300 73  4.467 77
3.367 66  4.033 80  3.833 74  2.017 52  1.867 48  4.833 80  1.833 59  4.783 90
4.350 80  1.883 58  4.567 84  1.750 58  4.533 73  3.317 83  3.833 64  2.100 53
4.633 82  2.000 59  4.800 75  4.716 90  1.833 54  4.833 80  1.733 54  4.883 83
3.717 71  1.667 64  4.567 77  4.317 81  2.233 59  4.500 84  1.750 48  4.800 82
1.817 60  4.400 92  4.167 78  4.700 78  2.067 65  4.700 73  4.033 82  1.967 56
4.500 79  4.000 71  1.983 62  5.067 76  2.017 60  4.567 78  3.883 76  3.600 83
4.133 75  4.333 82  4.100 70  2.633 65  4.067 73  4.933 88  3.950 76  4.517 80
2.167 48  4.000 86  2.200 60  4.333 90  1.867 50  4.817 78  1.833 63  4.300 72
4.667 84  3.750 75  1.867 51  4.900 82  2.483 62  4.367 88  2.100 49  4.500 83
4.050 81  1.867 47  4.700 84  1.783 52  4.850 86  3.683 81  4.733 75  2.300 59
4.900 89  4.417 79  1.700 59  4.633 81  2.317 50  4.600 85  1.817 59  4.417 87
2.617 53  4.067 69  4.250 77  1.967 56  4.600 88  3.767 81  1.917 45  4.500 82
2.267 55  4.650 90  1.867 45  4.167 83  2.800 56  4.333 89  1.833 46  4.383 82
1.883 51  4.933 86  2.033 53  3.733 79  4.233 81  2.233 60  4.533 82  4.817 77
4.333 76  1.983 59  4.633 80  2.017 49  5.100 96  1.800 53  5.033 77  4.000 77
2.400 65  4.600 81  3.567 71  4.000 70  4.500 81  4.083 93  1.800 53  3.967 89
2.200 45  4.150 86  2.000 58  3.833 78  3.500 66  4.583 76  2.367 63  5.000 88
1.933 52  4.617 93  1.917 49  2.083 57  4.583 77  3.333 68  4.167 81  4.333 81
4.500 73  2.417 50  4.000 85  4.167 74  1.883 55  4.583 77  4.250 83  3.767 83
2.033 51  4.433 78  4.083 84  1.833 46  4.417 83  2.183 55  4.800 81  1.833 57
4.800 76  4.100 84  3.966 77  4.233 81  3.500 87  4.366 77  2.250 51  4.667 78
2.100 60  4.350 82  4.133 91  1.867 53  4.600 78  1.783 46  4.367 77  3.850 84
1.933 49  4.500 83  2.383 71  4.700 80  1.867 49  3.833 75  3.417 64  4.233 76
```

```
2.400 53  4.800 94  2.000 55  4.150 76  1.867 50  4.267 82  1.750 54  4.483 75
4.000 78  4.117 79  4.083 78  4.267 78  3.917 70  4.550 79  4.083 70  2.417 54
4.183 86  2.217 50  4.450 90  1.883 54  1.850 54  4.283 77  3.950 79  2.333 64
4.150 75  2.350 47  4.933 86  2.900 63  4.583 85  3.833 82  2.083 57  4.367 82
2.133 67  4.350 74  2.200 54  4.450 83  3.567 73  4.500 73  4.150 88  3.817 80
3.917 71  4.450 83  2.000 56  4.283 79  4.767 78  4.533 84  1.850 58  4.250 83
1.983 43  2.250 60  4.750 75  4.117 81  2.150 46  4.417 90  1.817 46  4.467 74
;
```

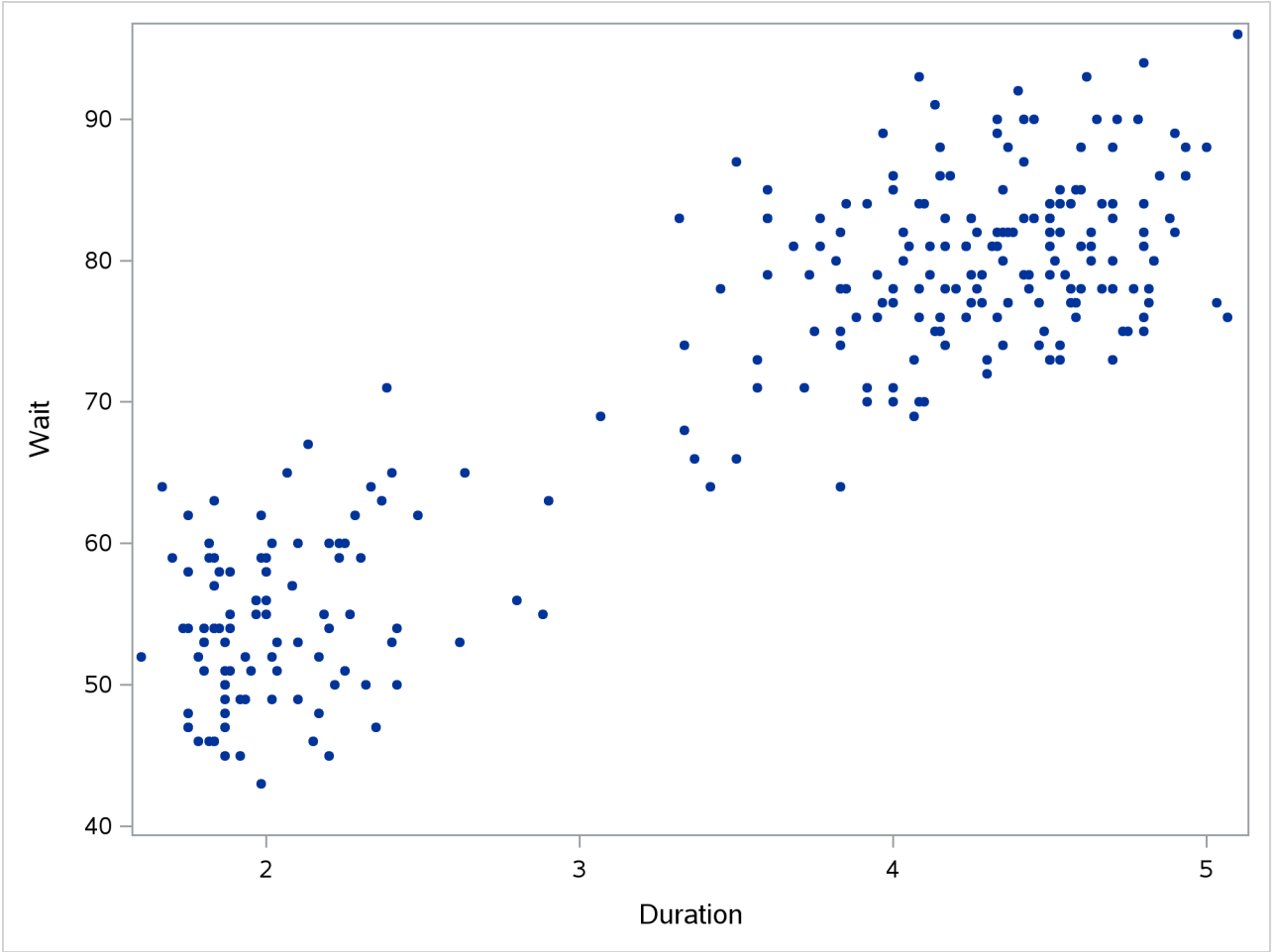
These statements assume that your libref is named mylib, but you can substitute any appropriately defined libref.

The following statements produce a scatter plot of the duration and wait times:

```
proc sgplot data=mylib.getStarted;
  scatter x=duration y=wait / markerattrs=(symbol=circlefilled size=4);
run; quit;
```

It is clear from the scatter plot in Figure 19.1 that there are two clusters of eruptions. The first consists of short, frequent eruptions, and the second consists of long, infrequent eruptions. However, the shape of these clusters is not spherical. The MBC procedure can help you explore the shapes.

Figure 19.1 Scatter Plot of Old Faithful Data



The following statements fit several different mixture models to these data:

```
proc mbc data=mylib.getStarted nclusters=(1 to 5)
                               noise=(YES NO)
                               covstruct=(EEV EII VVV)
                               seed=1418410433;
  var duration wait;
  output out=mylib.scores copyvars=(duration wait);
run;
```

The NCLUSTERS= option list fits separate models by using one to five Gaussian clusters. The NOISE= option list fits these models with and without noise. The COVSTRUCT= option list fits each of these models by using three different covariance structures. Finally, the SEED= option specifies a starting random seed to generate initial cluster weights. In total, PROC MBC fits 30 different models, corresponding to the unique combinations of each element in the NCLUSTERS=, NOISE=, and COVSTRUCT= option lists.

The VAR statement identifies the two variables, Duration and Wait, that define the points to be clustered.

The OUTPUT statement specifies a data table to contain the posterior cluster weights for each observation. The COPYVARS= option includes the two analysis variables, Duration and Wait, in the output data table.

After fitting all 30 models, PROC MBC chooses the model that has the best Bayesian information criterion (BIC) value. BIC is a quantity composed of the log likelihood and a penalty term. Smaller BIC values indicate models with better fit. The "Fit Summary" table in Figure 19.2 shows model information for the 10 models that have the best (smallest) BIC values, sorted so that the best model appears first. The selected model has two nonspherical clusters, as suggested by the plot in Figure 19.1.

Figure 19.2 Summary of Fit Statistics

The MBC Procedure

Model Selection Summary

Covariance Structure	Number of Clusters	Noise Component	Number of Parameters	-2 Log L	AIC	AICC	BIC
VVV	2	N	11	2260.52818	2282.52818	2283.54356	2322.19200
VVV	2	Y	13	2260.61415	2286.61415	2288.02500	2333.48957
VVV	3	N	17	2238.48109	2272.48109	2274.89054	2333.77973
EEV	2	Y	11	2278.66416	2300.66416	2301.67955	2340.32798
VVV	3	Y	19	2238.48207	2276.48207	2279.49794	2344.99231
VVV	4	N	23	2216.08379	2262.08379	2266.53541	2345.01724
EEV	4	Y	19	2241.44464	2279.44464	2282.46051	2347.95488
EEV	3	Y	15	2266.00364	2296.00364	2297.87864	2350.09067
EEV	5	N	21	2239.56249	2281.56249	2285.25849	2357.28433
VVV	5	N	29	2208.49803	2266.49803	2273.68811	2371.06629

Figure 19.3 shows the "Model Information" table. This table summarizes the characteristics of the selected model and of the modeling technique that was used.

Figure 19.3 Model Information for Selected Model

The MBC Procedure

Model Information	
Number of Gaussian Clusters	2
Covariance Structure	VVV
Noise Cluster Present	No
Expectation Technique	EM
Model Selection Criterion	BIC
Initialization Method	Random
Convergence Test	Log Likelihood
EM Convergence Criterion	1e-05
Singularity Criterion	1e-08
Parameter Criterion	1e-08
Random Seed	1418410433

The "Parameter Estimates" table in Figure 19.4 contains the estimates of the centers of the two clusters and the estimates of the covariance for the multivariate Gaussian that is represented by each cluster.

Figure 19.4 Mean and Covariance Estimates for Selected Model

Parameter Estimates for Selected Model				
			Covariance	
Cluster	**Variable**	**Mean**	**Duration**	**Wait**
1	**Duration**	4.28979	0.16980	0.93852
	Wait	79.96968		36.02280
2	**Duration**	2.03654	0.06928	0.43639
	Wait	54.48000		33.70574

If you compare the mean estimates to the plot in Figure 19.1, you can see that the estimates correspond well to the data.

Figure 19.1 suggests that one cluster is slightly heavier than the other. The "Mixing Estimates" table in Figure 19.5 supports this impression.

Figure 19.5 Mixing Weight Estimates for Selected Model

Mixing Probability Estimates for Selected Model	
Mixing Component	**Mixing Probability**
1	0.64407
2	0.35593

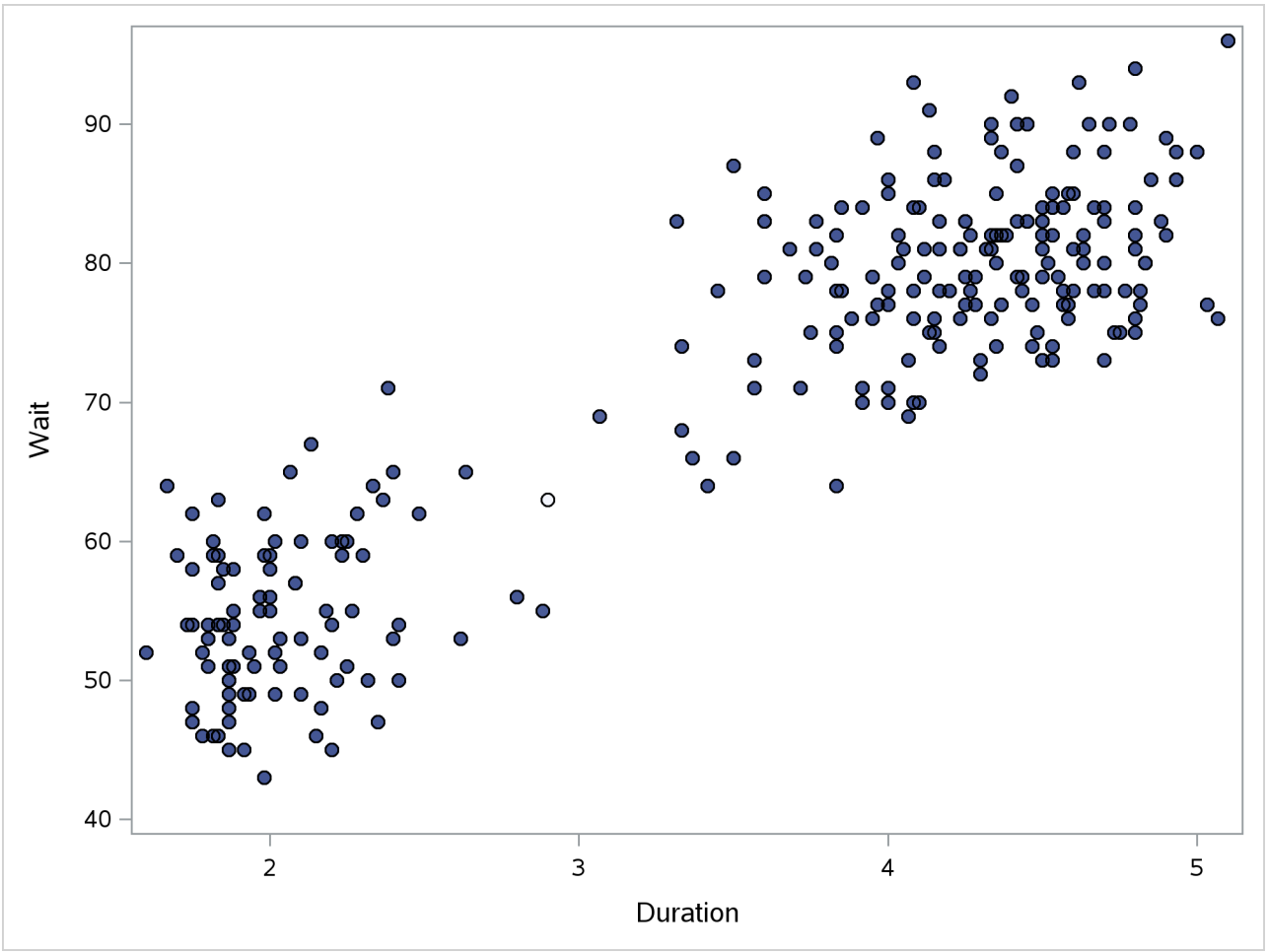
The following statements use the output table mylib.scores to produce a plot of the maximum posterior weights for each observation:

```
data scores;
  set mylib.scores;
  maxwt = max(of next:);
run;

proc sgplot data=scores noautolegend;
  * ---- shaded corresponding to maximum posterior weight --------- ;
  scatter x=duration y=wait   / markerattrs=(symbol=circlefilled)
                                colorresponse=maxwt
                                colormodel=TwoColorRamp;
  scatter x=duration y=wait   / markerattrs=(symbol=circle color=black);
run;
quit;
```

Figure 19.6 plots the observations with shading to indicate the strength of the affinity that each point has for the cluster that it is more strongly associated with. Only one point has light shading, indicating that it is in a region where the larger posterior cluster weight is not as large as it is in other regions. This indicates a region where the posterior weights for the two clusters are closer in value.

Figure 19.6 Old Faithful Data and Selected Model



Syntax: MBC Procedure

The following statements are available in the MBC procedure:

PROC MBC < *options* > **;**
BY *variables* **;**
VAR *variables* **;**
INIT *variables* **;**
OUTPUT OUT=*libref.data-table* < *keyword* < **=***name* > >. . . < *keyword* < **=***name* > > < *options* > **;**
DISPLAY < *table-list* > < / *options* > **;**
DISPLAYOUT *table-spec-list* < / *options* > **;**
STORE < **OUT=** >*libref.data-table* **;**

The PROC MBC statement is required.

PROC MBC Statement

PROC MBC < *options* > **;**

The PROC MBC statement invokes the procedure. Table 19.1 summarizes the available *options* in the PROC MBC statement by function. They are then described fully in alphabetical order.

Table 19.1 PROC MBC Statement Options

Option	Description
Basic Options	
DATA=	Specifies the input data table
Options Related to Model Fitting	
CONVTEST=	Specifies the convergence test to use for model fitting
COVSTRUCT=	Specifies the covariance structure
CRITERION=	Specifies the fit statistic to use for model selection
EMCRITERION=	Tunes the expectation-maximization (EM) convergence criterion
FACTORDETAILS	Includes factor loadings and unique error variances in the ParameterEstimates table
INIT=	Specifies the initialization method
ITHIST=	Displays the iteration history
MAXITER=	Specifies the maximum number of EM iterations
NCLUSTERS=	Specifies the number of Gaussian components
NFACTORS=	Specifies the number of latent factors
NOISE=	Specifies the presence of noise
SEED=	Specifies the pseudorandom number seed
SINGPARM=	Tunes the parameter singularity criterion
SINGULAR=	Tunes the singularity criterion
TECHNIQUE=	Specifies the EM technique
TOPMODELS=	Specifies the number of models to display in summary

The following options are available in the PROC MBC statement:

CONVTEST=AITKEN | LOGL
specifies the convergence test to use. You can specify the following values:

AITKEN uses Aitken's acceleration (McLachlan and Krishnan 1997) with the stopping criterion described in Böhning et al. (1994) for computing the test value in the following steps:

1. $a^{(k)} = \frac{\ell^{(k+1)} - \ell^{(k)}}{\ell^{(k)} - \ell^{(k-1)}}$, where $\ell^{(k)}$ is the log-likelihood value at the kth iteration.
2. $\ell_{\infty}^{(k+1)} = \ell^{(k)} + \frac{1}{1-a^{(k)}} \left(\ell^{(k+1)} - \ell^{(k)} \right)$
3. Compute $\left| \ell_{\infty}^{(k+1)} - \ell_{\infty}^{(k)} \right|$ and compare this value to the convergence criterion value.

LOGL uses the relative log-likelihood change test: $|\ell^{(k+1)} - \ell^{(k)}| / |\ell^{(k)}|$, where $\ell^{(k)}$ is the model log likelihood after the kth iteration.

By default, CONVTEST=LOGL. If the convergence test value is less than or equal to the value specified in the EMCRITERION= option, the process is considered to have converged.

COVSTRUCT=*name* | (*name* < *name* ... *name* >) | ALLGMIX | ALLPGMIX | ALL
specifies the covariance structure for the Gaussian components. The value of *name* can be one of the covariance structures for Gaussian mixture modeling or parsimonious Gaussian mixture modeling.

Nine covariance structures are available for fitting Gaussian mixture models (GMMs):
COVSTRUCT=EEE | EEI | EEV | EII | EVI | EVV | VII | VVI | VVV.

Eight covariance structures are available for fitting parsimonious Gaussian mixture models (PGMMs):
COVSTRUCT=CCC | CCU | CUC | CUU | UCC | UCU | UUC | UUU.

By default, COVSTRUCT=VVV.

The GMM structures impose one or more constraints on the covariance matrices in the separate Gaussian components, whereas the PGMM structures constrain the factor loading matrices and the unique error variance matrices. For a complete discussion of the different covariance structures, see the section "Covariance Structures" on page 1110.

If you specify a list of covariance structures, PROC MBC fits a separate model for each value in the list and for each value of the NCLUSTERS= and NOISE= options. For any covariance structure that belongs to the set of PGMM covariance structures, the procedure also fits a separate model for each value of the NFACTORS= option.

If you specify the ALLGMIX value, the procedure fits models by using each Gaussian mixture covariance structure. If you specify the ALLPGMIX value, the procedure fits models by using each parsimonious Gaussian mixture covariance structure. Specifying the ALL value is equivalent to specifying both ALLGMIX and ALLPGMIX.

If you specify one of the parsimonious Gaussian mixture covariance structures, the ALLPGMIX value, or the ALL value, you can also specify the number of latent factors by using the NFACTORS= option. The procedure identifies the model that has the best fit, where the fit is computed according to the value of the CRITERION= option.

CRITERION=AIC | AICC | BIC | LOGL
specifies the model fit criterion to use in model selection. You can specify the following values:

AIC	specifies Akaike's information criterion.
AICC	specifies Akaike's information criterion (corrected).
BIC	specifies the Bayesian information criterion.
LOGL	specifies the log likelihood.

By default, CRITERION=BIC. Each criterion is defined so that a model with a smaller value for the criterion is judged to have a better fit than a model with a larger value for the same criterion.

DATA=*libref.data-table*
names the input data table for PROC MBC to use. The default is the most recently created data table. *libref.data-table* is a two-level name, where

libref	refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to the data, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 1093.
data-table	specifies the name of the input data table.

EMCRITERION=*number*
specifies the convergence criterion for the expectation-maximization (EM) technique. The EM process terminates when the value of the test that you specify in the CONVTEST= option is less than or equal to *number*. The value of *number* must be greater than or equal to 0. The default value depends on the value of the CONVTEST= option. If you specify CONVTEST=LOGL or do not specify a value for CONVTEST, then by default EMCRITERION=1E–5. If you specify CONVTEST=AITKEN, then by default EMCRITERION=1E–3.

FACTORDETAILS
includes the factor loadings and the unique error variances in the "Parameter Estimates" table. These estimates provide a fuller picture of the model. In general, the estimated factor loadings are subject to rotational indeterminacy, and thus they are not indicative of the associations between the factors and the observed variables. This option has no effect on models that do not use one of the parsimonious Gaussian mixture model covariance structures. For a complete description of the different types of covariance structures, see the COVSTRUCT= option.

INIT=KMEANS | RANDOM
specifies which method to use to initialize cluster membership. This option is ignored if you specify initial weights in the INIT statement. You can specify one of the following values:

KMEANS	uses *k*-means clustering to generate starting values. BY-group processing is not available when you specify this method.
RANDOM	specifies the random weight method.

By default, INIT=RANDOM.

If you use the INIT statement to specify initialization variables, the procedure uses those variables for initialization and ignores any INIT= option.

ITHIST < =DETAILS | NONE | SUMMARY >
produces the "Iteration History" table. You can specify one of the following levels:

DETAILS	produces a table that contains expanded information about each iteration.
NONE	suppresses the production of the table.
SUMMARY	produces a table that contains basic information about each iteration.

If you specify the ITHIST option but do not specify one of the levels, the procedure uses the SUMMARY level.

MAXITER=*number*
specifies the maximum number of expectation-maximization (EM) iterations before termination. The EM process terminates after the specified *number* of iterations, even if the process has not converged. By default, MAXITER=500.

NCLUSTERS=*number* | (*number* < *number* **...** *number* >) | (*number* **to** *number* < **by** *number* >)
specifies the number of Gaussian components in the model. If you specify a list of numbers or a range of numbers, PROC MBC fits a separate model for each value in the list or range and for all values of the NOISE= and COVSTRUCT= options. For any covariance structure that belongs to the set of PGMM covariance structures, the procedure also fits a separate model for each value of the NFACTORS= option. The procedure identifies the model that has the best fit, where the fit is computed according to the value of the CRITERION= option. By default, NCLUSTERS=2.

If you use the (*smallest* to *largest* by *increment*) syntax, the list contains the values (*smallest*, *smallest*+*increment*, *smallest*+2**increment*, . . ., *smallest*+*p***increment*), where each value in the list is less than or equal to *largest*. If the resulting list contains any noninteger values or values less than 1, the procedure terminates with a syntax error.

NFACTORS=*number* | (*number* < *number* **...** *number* >) | (*number* **to** *number* < **by** *number* >)
specifies the number of latent factors to use with PGMM covariance structures. All specified values must be less than or equal to the number of variables that you specify in the VAR statement. If you specify a list of numbers or a range of numbers, PROC MBC fits a separate model for each value in the list or range, for all values of the NOISE= option, for all values of the NCLUSTERS= option, and for values of the COVSTRUCT= option that belong to the set of PGMM covariance structures. The procedure identifies the model that has the best fit, where the fit is computed according to the value of the CRITERION= option. By default, NFACTORS=1.

If you use the (*smallest* to *largest* by *increment*) syntax, the list contains the values (*smallest*, *smallest*+*increment*, *smallest*+2**increment*, . . ., *smallest*+*p***increment*), where each value in the list is less than or equal to *largest*. If the resulting list contains any noninteger values or values less than 1, the procedure terminates with a syntax error.

NOISE=YES | NO | (*value* < *value* . . . *value* >)
specifies whether to include a noise component. If you specify a list of values, PROC MBC fits a separate model for each distinct value in the list and for all values of the COVSTRUCT= and NCLUSTERS= options. For any covariance structure that belongs to the set of PGMM covariance structures, the procedure also fits a separate model for each value of the NFACTORS= option. The procedure identifies the model that has the best fit, where the fit is computed according to the value of the CRITERION= option. By default, NOISE=NO.

SEED=*integer*
specifies the pseudorandom number seed for generating random starting weights. Random starting weights are used only when you do not specify starting weight variables in the INIT statement and when you do not specify INIT=RANDOM. If you do not specify a seed, then the seed is derived from the default random number generator on the controller.

SINGPARM=*number*
specifies the parameter singularity criterion. If the mixing estimate for a model component is less than or equal to *number*, it is considered to be zero. In models that have noise components, if the computed noise volume is less than or equal to *number*, the procedure will not fit the model. The value of *number* must be greater than zero. By default, SINGPARM=1E–8.

SINGULAR=*number*
specifies the general singularity criterion to use in matrix operations. The value of *number* must be greater than or equal to zero. By default, SINGULAR=1E–8.

TECHNIQUE=EM | CEM
specifies the expectation-maximization (EM) technique to use. You can specify the following values:

EM specifies the traditional EM technique.

CEM specifies the classification EM technique.

By default, TECHNIQUE=EM.

TOPMODELS=*number*
specifies the maximum number of models to include in the FitSummary table. The value of *number* must be an integer greater than or equal to 1. This option is applicable only when at least one of the NCLUSTERS=, NOISE=, or COVSTRUCT= options has multiple values. The FitSummary table displays summary information for *number* of models or for as many models as were fit, whichever is smaller. By default, TOPMODELS=10.

BY Statement

BY *variables* **;**

You can specify a BY statement in PROC MBC to obtain separate analyses of observations in groups that are defined by the values of the BY variables. If you specify more than one BY statement, only the last one specified is used. For more information, see the discussion of BY-group processing in the "Grouping Data" section of *SAS Programmers Guide: Essentials*.

DISPLAY Statement

DISPLAY < *table-list* > < / *options* > **;**

The DISPLAY statement enables you to specify a list of display tables to display or exclude. This statement is similar to the ODS SELECT, ODS EXCLUDE, and ODS TRACE statements. However, the DISPLAY statement can improve performance when a large number of tables could be generated (such as in BY-group processing). The procedure processes the DISPLAY statement on a CAS server and thus sends only a subset of ODS tables to the SAS client. Because ODS statements are processed on a SAS client, first all the generated display tables are sent to the client, and then the client creates a subset.

If you use both DISPLAY and ODS statements together, the DISPLAY statement takes precedence over the ODS statements. Note that the ODS EXCLUDE statement processes tables that are sent to the client after they have been filtered by the DISPLAY statement. In some cases, it might appear that the ODS EXCLUDE statement is taking precedence because it can further filter the tables. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

You can specify the *table-list* as a list of table names, paths, partial pathnames, and regular expressions.

The table names that you can specify are listed in the section "ODS Table Names" on page 1116. A path is a table name that is prefixed with dot-separated grouping information. For example, a SelectionSummary table that a procedure produces during a selection routine might have the path *Bygroup1.Summary.SelectionSummary*. A partial pathname does not include all groups; for example, *SelectionSummary* and *Summary.SelectionSummary* are partial pathnames for *Bygroup1.Summary.SelectionSummary*.

When you specify a table name or partial pathname, all display tables whose paths end in the specified name are selected for display or exclusion. For example, both *SelectionSummary* and *Summary.SelectionSummary* select *Bygroup1.Summary.SelectionSummary*.

A regular expression is enclosed in forward slashes (/). For example, specifying "/tions/" selects all pathnames that contain the substring "tions"; in particular, the *Bygroup1.Summary.SelectionSummary* table is selected. Specifying "!/tions/" selects all pathnames that do not contain the substring "tions"; in particular, the *Bygroup1.Summary.SelectionSummary* table is not selected.

You can specify the following *options* after a slash (/):

CASESENSITIVE
performs a case-sensitive comparison of table names in the *table-list* to display table names when tables are subsetted for display. To preserve case, you must enclose table names in the *table-list* in quotation marks.

EXCLUDE
displays all display tables except those that you specify in the *table-list*.

EXCLUDEALL
suppresses display of all tables. This option takes precedence over the other options.

TRACE
displays the display table names, labels, and paths.

DISPLAYOUT Statement

DISPLAYOUT *table-spec-list* < / *options* > **;**

The DISPLAYOUT statement enables you to create output tables from your displayed output. This statement is similar to the ODS OUTPUT statement. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

The *table-spec-list* specifies a list of output tables to create. Each entry in the list has either a *key*=*value* format or a *key* format:

key=*value*	specifies *key* as the ODS table name, path, or partial pathname, and specifies *value* as the output table name.
key	specifies *key* as the ODS table name and also as the output table name.

The ODS table names that you can specify are listed in the section "ODS Table Names" on page 1116. You cannot specify the ODS table named OutputCasTables in the *table-spec-list*.

Table names and partial pathnames are discussed under the DISPLAY statement. The DISPLAYOUT statement does not support regular expressions.

You can specify the following *options* after a slash (/):

INCLUDEALL
creates output tables for all display tables. The name of the created output table is the same as the name of the corresponding display table. If you specify this option, the *table-spec-list* specification is ignored.

NOREPLACE
does not replace any existing output table of the same name.

REPEATED
replicates all output tables on all nodes.

INIT Statement

INIT *variables* **;**

The INIT statement specifies the variables that contain initial weights for the mixture components. By convention, if the model includes a noise component, the first *variable* in the INIT statement list defines the observation's initial weight for the noise component. If you are fitting many different models by using the NCLUSTERS=, NOISE=, or COVSTRUCT= option, you must supply enough INIT *variables* for the model combination that has the most components. The INIT statement is not required. You can specify at most one INIT statement.

If any of the variables have negative values or if the sum of the variables is less than the value specified in the SINGULAR= option, the procedure will use equal initial weights on each cluster for that observation.

Without an INIT statement, PROC MBC uses the default initialization method.

OUTPUT Statement

OUTPUT OUT=*libref.data-table* < *keyword* < **=***name* > >. . . < *keyword* < **=***name* > > < *options* > **;**

The OUTPUT statement creates a data table that contains observationwise statistics that PROC MBC computes after fitting the model. The variables in the input data table are *not* included in the output data table, in order to avoid data duplication for large data tables; however, variables that you specify in the COPYVAR= option are included. The output statistics are computed on the basis of the final parameter estimates. If no model converges, then the output data table is not created.

You must specify the following option:

OUT=*libref.data-table*
names the output data table for PROC MBC to use. You must specify this option before any other options. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to where the data table is to be stored, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 1093.

data-table specifies the name of the output data table.

You can also specify the following options:

ALL

ALLSTAT
adds all available statistics to the output data table.

COPYVAR=*variable*

COPYVARS=(*variables***)**
transfers one or more *variables* from the input data table to the output data table.

keyword **<=***name***>**
specifies a statistic to include in the output data table and optionally names the variable *name*. If you provide the *keyword* but do not provide a *name*, PROC MBC assigns a default name that is based on the type of statistic requested. PROC MBC produces the NEXTCLUS statistics even if you do not provide the corresponding keyword.

Table 19.2 summarizes the *keywords* available in the OUTPUT statement.

Table 19.2 OUTPUT Statement Keywords

Keyword	Description	Default Name
Statistic Options		
CURRCLUS	Specifies the cluster weights that produce the final parameter estimates	CURR
LOGLIK	Specifies the log likelihood produced by the final parameter estimates	LOGLIK
MAXPOST	Specifies the index of the component with the maximum posterior probability	MAXPOST
NEXTCLUS	Specifies the cluster weights produced by the final parameter estimates	NEXT

The following list describes these *keywords*. For more information, see the section "Posterior Probabilities and Clustering" on page 1109.

CURRCLUS <=*name***>**
specifies the cluster weights that produce the final parameter estimates. The default name is CURR*n*, where *n* indicates the cluster index. You can modify the name by specifying the *name* option.

LOGLIK <=*name***>**
specifies the component and total log likelihood by using the final parameter estimates. The default name for the component log likelihoods is LOGLIK*n*, where *n* indicates the cluster index. The default name for the total log likelihood is LOGLIK. You can modify this name by specifying the *name* option.

MAXPOST <=*name***>**
specifies the identification of the cluster that has the maximum posterior weight by using the final parameter estimates. The default name for this variable in the output data set is MAXPOST. You can modify this name by specifying the *name* option. During scoring, if an observation has the same posterior weight for multiple clusters, the value of the output variable *name* for that observation is set to the lowest cluster number among those clusters.

NEXTCLUS <=*name***>**
specifies the cluster weights that the final parameter estimates produce. The default name is NEXT*n*, where *n* indicates the cluster index. You can modify the name by specifying the *name* option.

STORE Statement

STORE <OUT=>*libref.data-table* **;**

The STORE statement enables you to save the fitted model to an analytic store. The contents of the analytic store can be processed by the ASTORE procedure and by the **`mbcScore`** action in the **`mbc`** action set.

libref.data-table is a two-level name, where *libref* refers to the library, and *data-table* specifies the name of the input data table. For more information about this two-level name, see the DATA= option and the section "Using CAS Sessions and CAS Engine Librefs" on page 1093. You can specify at most one STORE statement.

VAR Statement

VAR *variables* **;**

The VAR statement specifies the variables that you want to cluster. This statement is required, and you can specify at most one VAR statement.

Details: MBC Procedure

Log-Likelihood Definitions

The MBC procedure uses two definitions of the log likelihood to assess convergence. Which definition the procedure uses depends on the value of the TECHNIQUE= option.

If you do not specify the TECHNIQUE= option or if you specify TECHNIQUE=EM, PROC MBC uses the mixture log likelihood ℓ_{mix}, defined as

$$\ell_{\text{mix}} = \sum_{i=1}^{N} \log \left[\sum_{k=0}^{G} \tau_k f\left(\mathbf{y}_i | \boldsymbol{\theta}_k\right) \right]$$

where τ_k are the mixture weights and $\boldsymbol{\theta}_k$ are the mixture component parameters.

If you specify TECHNIQUE=CEM, the procedure uses the classification complete data log likelihood ℓ_{CDL}, defined as

$$\ell_{\text{CDL}} = \sum_{i=1}^{N} \sum_{k=0}^{G} \tilde{z}_{ik} \log \left[\tau_k f\left(\mathbf{y}_i | \boldsymbol{\theta}_k\right)\right]$$

In this definition, $\tilde{z}_{ik}$ is an indicator variable.

Posterior Probabilities and Clustering

In the OUTPUT statement, the CURRCLUS and NEXTCLUS options produce posterior probabilities (weights) for each observation. The weights that are computed for the CURRCLUS option are the observation weights that produce the final parameter estimates. In contrast, the weights that are produced for the NEXTCLUS option are the weights that the final parameter estimates produce for each observation.

The MAXPOST option outputs the index of the cluster that has the maximum posterior probability for each observation. This maximum is taken over the posterior probabilities that are produced by the final parameter estimates.

Expectation-Maximization (EM) Algorithm

Traditional EM

The expectation-maximization (EM) algorithm (Dempster, Laird, and Rubin 1977) is a tool that addresses problems of missing data. The EM algorithm proceeds by finding the conditional expectation of the missing data, given the observed data, treating that conditional expectation as observed data, and iterating until the model converges. In the Gaussian mixture models that PROC MBC fits, the missing data are the identity of the cluster to which each observation belongs. The steps of the EM algorithm follow:

1. Initialize the cluster membership for each observation through the specified technique (INIT= or INIT statement).
2. Update the cluster-specific parameters by using the current cluster memberships. Retain these new parameters as $\hat{\boldsymbol{\theta}}^{(t+1)}$.
3. Given the current parameters $\hat{\boldsymbol{\theta}}^{(t+1)}$, compute the conditional expected values of cluster membership for each observation. These are estimates $\hat{z}_{ik}$ such that

$$\hat{z}_{ik} = \text{estimated probability that observation } i \text{ belongs to cluster } k; 0 \leq z_{ik} \leq 1 \; \forall i,k; \; \sum_{k=1}^{G} \hat{z}_{ik} = 1$$

4. Evaluate the current value of the convergence criterion. You specify the convergence criterion by using the CONVTEST= option. If the convergence criterion is less than or equal to the specified value of the EMCRITERION= option, terminate the algorithm. Otherwise, go to step 2.

Classification EM

Classification EM (Celeux and Govaert 1995) proceeds in the same manner as traditional EM, but it uses the classification log likelihood to assess convergence in step 4 and replaces the "fuzzy" $\hat{z}_{ik}$ in step 3 with hard classifications: $\tilde{z}_{ik} = 1$ if $\hat{z}_{ik} = \max_j\{\hat{z}_{ij}\}$, $\tilde{z}_{ik} = 0$ otherwise.

Alternating Expectation-Conditional Maximization

PROC MBC uses the alternating expectation-conditional maximization (AECM) algorithm (Meng and Van Dyk 1997) in the estimation process for the parsimonious Gaussian mixture models (PGMMs). The PGMMs rely on latent factors, which are not observed. The AECM algorithm provides a framework for estimation in the presence of this additional source of missing data. AECM adds a step to the EM algorithm process in order to update the covariance parameters.

Singular Covariance Matrices

In some circumstances, the covariance matrix for a Gaussian component can become singular during the progress of the EM algorithm. This terminates the model fitting process. This can happen if the cluster represents a single observation or a set of effectively identical observations. When this happens, PROC MBC issues a note that indicates the existence of the singularity. You might be able to use a different initialization method or random seed to alter the progress of the algorithm and so avoid the singularity, but this is not possible in all cases.

Covariance Structures

PROC MBC supports two types of Gaussian mixtures. The two types are distinguished by their use of latent factors in the model. The models that do not use latent factors are referred to as Gaussian mixture models (GMMs), and the models that use latent factors are referred to as parsimonious Gaussian mixture models (PGMMs). These models use different sets of covariance structures. The covariance structures of the Gaussian mixture models are directly specified and constrained. In contrast, the covariance structures of the PGMMs are derived from the underlying factor models. The constraints for PGMMs are set through the factor loading and unique error variance matrices.

Gaussian Mixtures

The covariance structures available in PROC MBC follow the notation of Banfield and Raftery (1993). In this treatment, the three aspects of each component's covariance (shape, volume, and orientation) can be left arbitrary or can be forced to be equal across clusters. In addition, the shape can be forced to be spherical, and the orientation can be forced to align with the coordinate axes.

The eigenvalue decomposition of the covariance matrix for the kth Gaussian mixture component is written as

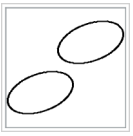
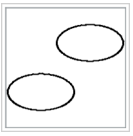
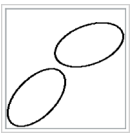
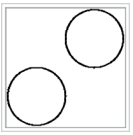
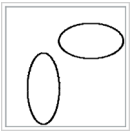
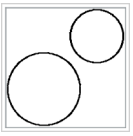
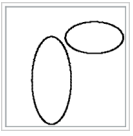
$$\mathbf{\Sigma}_k = \lambda_k \mathbf{D}_k \mathbf{A}_k \mathbf{D}_k'$$

where λ_k is the volume factor for $\mathbf{\Sigma}_k$, $\mathbf{A}_k$ is a diagonal matrix with the normalized eigenvalues of $\mathbf{\Sigma}_k$ on the diagonal, and $\mathbf{D}_k$ is the matrix of eigenvectors of $\mathbf{\Sigma}_k$. In a geometric description of the covariance matrix $\mathbf{\Sigma}_k$, λ_k characterizes the volume, $\mathbf{A}_k$ characterizes the shape, and $\mathbf{D}_k$ characterizes the orientation.

There are several ways to restrict the covariance structures across the Gaussian clusters. You can hold the volume constant by setting each λ_k equal to λ. To leave the shape arbitrary but common across the clusters, you can require each $\mathbf{A}_k$ to equal $\mathbf{A}$. To require spherical covariance matrices, you can restrict each $\mathbf{A}_k$ to equal $\mathbf{I}$, the identity matrix. If you hold each $\mathbf{D}_k$ equal to $\mathbf{D}$, then all covariance matrices will have the same orientation. You can also require all covariance matrices to be oriented to the coordinate axes by specifying $\mathbf{D}_k$ equal to $\mathbf{I}$.

PROC MBC accommodates a subset of the possible covariance structure restrictions. Table 19.3 describes the structures that the procedure supports. The Name column indicates how you specify each structure when invoking the MBC procedure. The Profile column shows a two-dimensional visualization of the covariance structure in a model that contains two clusters.

Table 19.3 Covariance Structures for Gaussian Mixtures

Name	Structure	Volume	Shape	Orientation	Profile
EEE	$\lambda\mathbf{DAD}'$	Equal	Equal	Equal	
EEI	$\lambda\mathbf{A}$	Equal	Equal	Coordinate axes	
EEV	$\lambda\mathbf{D}_k\mathbf{AD}_k'$	Equal	Equal	Variable	
EII	$\lambda\mathbf{I}$	Equal	Spherical	Coordinate axes	
EVI	$\lambda\mathbf{A}_k$	Equal	Variable	Coordinate axes	
EVV	$\lambda\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k'$	Equal	Variable	Variable	
VII	$\lambda_k\mathbf{I}$	Variable	Spherical	Coordinate axes	
VVI	$\lambda_k\mathbf{A}_k$	Variable	Variable	Coordinate axes	
VVV	$\lambda_k\mathbf{D}_k\mathbf{A}_k\mathbf{D}_k'$	Variable	Variable	Variable	

Parsimonious Gaussian Mixtures

The PGMM covariance structures assume that the observed values $\boldsymbol{y}_i$ are related to latent (unobserved) quantities $\boldsymbol{u}_i$ by a set of factor loadings and that the conditional variance of the $\boldsymbol{y}_i$ is described by a diagonal matrix, called the *unique error variance* matrix. In the mixture context, the PGMMs use a set of constraints that are defined by the invariance of factor loadings and unique error variances across mixture components. In the PGMMs, the distribution of $\boldsymbol{y}_i$ in the kth Gaussian component is based on the sampling model

$$\boldsymbol{y}_i|\boldsymbol{u}_{ik} \sim N(\boldsymbol{\mu}_k + \boldsymbol{\Lambda}_k \boldsymbol{u}_{ik}, \boldsymbol{\Psi}_k)$$

where the latent factor $\boldsymbol{u}_{ik} \sim N(\mathbf{0}, \mathbf{I}_q)$ and q is the number of latent factors. $\boldsymbol{\Lambda}_k$ is a matrix of factor loadings for the kth component, and $\boldsymbol{\Psi}_k$ is a diagonal matrix of unique error variances for the kth component. The marginal distribution of the $\boldsymbol{y}_i$ in the kth component after integrating out the $\boldsymbol{u}_i$ is then

$$\boldsymbol{y}_i \sim N(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k \boldsymbol{\Lambda}_k' + \boldsymbol{\Psi}_k)$$

The general form of the covariance matrix in the kth component is $\boldsymbol{\Lambda}_k \boldsymbol{\Lambda}_k' + \boldsymbol{\Psi}_k$. The eight covariance structures for the PGMMs are derived according to whether the factor loading matrices $\boldsymbol{\Lambda}_k$ are constrained to be equal across components ($\boldsymbol{\Lambda}_k = \boldsymbol{\Lambda}$), whether the error covariance matrices $\boldsymbol{\Psi}_k$ are constrained to be equal across components ($\boldsymbol{\Psi}_k = \boldsymbol{\Psi}$), and whether the error covariance matrices are isotropic ($\boldsymbol{\Psi}_k = \psi_k \mathbf{I}_p$). The eight covariance structures are summarized in Table 19.4. The UUU structure corresponds to the general mixture of factor analyzers model (Ghahramani and Hinton 1997; Yung 1997).

Table 19.4 Covariance Structures for Parsimonious Gaussian Mixtures

Name	$\boldsymbol{\Sigma}_k$ Structure	Factor Loadings	Error Variance	Isotropic Errors
CCC	$\boldsymbol{\Lambda}\boldsymbol{\Lambda}' + \psi \mathbf{I}_d$	Constrained	Constrained	Constrained
CCU	$\boldsymbol{\Lambda}\boldsymbol{\Lambda}' + \boldsymbol{\Psi}$	Constrained	Constrained	Unconstrained
CUC	$\boldsymbol{\Lambda}\boldsymbol{\Lambda}' + \psi_k \mathbf{I}_d$	Constrained	Unconstrained	Constrained
CUU	$\boldsymbol{\Lambda}\boldsymbol{\Lambda}' + \boldsymbol{\Psi}_k$	Constrained	Unconstrained	Unconstrained
UCC	$\boldsymbol{\Lambda}_k \boldsymbol{\Lambda}_k' + \psi \mathbf{I}_d$	Unconstrained	Constrained	Constrained
UCU	$\boldsymbol{\Lambda}_k \boldsymbol{\Lambda}_k' + \boldsymbol{\Psi}$	Unconstrained	Constrained	Unconstrained
UUC	$\boldsymbol{\Lambda}_k \boldsymbol{\Lambda}_k' + \psi_k \mathbf{I}_d$	Unconstrained	Unconstrained	Constrained
UUU	$\boldsymbol{\Lambda}_k \boldsymbol{\Lambda}_k' + \boldsymbol{\Psi}_k$	Unconstrained	Unconstrained	Unconstrained

For example, in the CUC structure, the first letter in the structure name indicates that the $\boldsymbol{\Lambda}_k$ are constrained to be equal, so that $\boldsymbol{\Lambda}$ is the common factor loading matrix for all components in the mixture. The second letter in the name indicates that the unique error variance matrices, $\boldsymbol{\Psi}_k$, are not constrained to be equal across components, so there is a different $\boldsymbol{\Psi}_k$ for each component. The final letter in the name indicates that within each Gaussian component, the unique error variance matrix $\boldsymbol{\Psi}_k$ is constrained to be isotropic, so that $\boldsymbol{\Psi}_k = \psi_k \mathbf{I}_d$.

If you choose a number of latent factors q that is smaller than the number of variables d, you can reduce the dimension of the data. This reduction in dimension is the motivation for the use of the term "parsimonious" in the name of these models. When the variability in the data is adequately explained in a lower-dimensional space, the model can achieve good performance without the complexity of a model in the original space. The choice of $q < d$ can also substantially reduce the number of parameters in the model. For more information

about the effect that the choice of q and specific constraints has on the number of parameters, see the section "Number of Parameters" on page 1114.

Noise Component

A noise component is uniform noise. This can be useful for modeling outliers. PROC MBC estimates the density of this cluster as the inverse of the volume of the space that contains the observations. The procedure computes the volume as the product of the range of the data in each dimension.

Missing Data

PROC MBC excludes from the analysis observations that have missing values for any variables in the VAR list.

If you use the INIT statement to specify initial weights and any of the variables in the INIT list have missing values, PROC MBC uses a small positive weight for each variable that has a missing value.

Model Selection

You can evaluate several different models by specifying lists of numbers of components, covariance structures, or noise components in the PROC MBC statement. The CRITERION= option in the PROC MBC statement specifies the model fit criterion that the procedure uses to compare different models. Each criterion uses the log likelihood of the model, given the training data, with a criterion-specific penalty term that favors smaller models.

Table 19.5 summarizes the computation of the available criteria.

Table 19.5 Model Selection Criteria

Name	Definition
AIC	$-2LL + 2p$
AICC	$-2LL + \frac{2pn}{n-p-1}$
BIC	$-2LL + p\log(n)$
LOGL	$-2\sum_{i=1}^{n}\log\left[\sum_{k=1}^{K}\tau_k f_k(y_i\|\theta_k)\right]$

Number of Parameters

The section "Model Selection" on page 1113 section defines each information criterion as a function of the number of observations n and the number of parameters p. The number of parameters depends on the number and type of components in the model, as well as the covariance structure that is chosen for the Gaussian components. Table 19.6 summarizes the methods that PROC MBC uses to determine the number of parameters in a particular model. In this table, G refers to the number of Gaussian components in the mixture model, d refers to the number of variables in the model, and q refers to the number of factors in the model. In the entries for each Gaussian mixture covariance structure, the total number of parameters is presented as the sum of contributions from volume, shape, and orientation. In the entries for each parsimonious Gaussian mixture covariance structure, the total number of parameters is presented as the sum of contributions from the factor loading matrices $\mathbf{\Lambda}_k$ and the unique error variance matrices $\mathbf{\Psi}_k$. The contribution from the $\mathbf{\Lambda}_k$ reflects that $q(q-1)/2$ constraints are necessary in order for $\mathbf{\Lambda}_k$ to be uniquely identified in the component factor models.

Table 19.6 Number of Parameters Computation

Source	Number of Parameters		
Mixing probabilities τ_k	$G-1$ without noise; G with noise		
Cluster means $\boldsymbol{\mu}_k$	Gd		
Noise volume	1		
Covariance Matrix $\mathbf{\Sigma}_k$: Gaussian Mixtures			
EEE	$1+(d-1)+(d-1)d/2$		
EEI	$1+(d-1)$		
EEV	$1+(d-1)+G(d-1)d/2$		
EII	1		
EVI	$1+G(d-1)$		
EVV	$1+G(d-1)+G(d-1)d/2$		
VII	G		
VVI	$G+G(d-1)$		
VVV	$G+G(d-1)+G(d-1)d/2$		
Covariance Matrix $\mathbf{\Sigma}_k$: Parsimonious Gaussian Mixtures			
	$\mathbf{\Lambda}$		**$\mathbf{\Psi}$**
CCC	$dq-q(q-1)/2$	+	1
CCU	$dq-q(q-1)/2$	+	d
CUC	$dq-q(q-1)/2$	+	G
CUU	$dq-q(q-1)/2$	+	Gd
UCC	$G\,[dq-q(q-1)/2]$	+	1
UCU	$G\,[dq-q(q-1)/2]$	+	d
UUC	$G\,[dq-q(q-1)/2]$	+	G
UUU	$G\,[dq-q(q-1)/2]$	+	Gd

Displayed Output

The following sections describe the output that PROC MBC produces. The output is organized into various tables, which are discussed in their order of appearance.

Model Information

The "Model Information" table displays basic information about the model, including the number of Gaussian clusters, the covariance structure of those clusters, and whether a noise component is included in the model. The table also includes information about the method that is used to estimate the parameters in the model and the tolerances for covariance matrices and parameters. If you specify multiple models, the table displays information about the selected model and also includes the model selection criterion that is used.

Number of Observations

The "Number of Observations" table displays the number of observations that are read from the input data table and the number of observations that are used in the analysis.

Iteration History

The "Iteration History" table displays the progress of the estimation algorithm for each model that is fitted. By default, this table includes information about the model, the iteration number, the log likelihood at that iteration, the change in the log likelihood, and the computed convergence criterion value. If you specify the ITHIST=DETAILS option, the table also includes the values of the model parameters at each iteration.

Convergence Status

The "Convergence Status" table displays the convergence status of each model. The table includes information about the model and a message that indicates whether the model converged or encountered an error.

Fit Statistics

The "Fit Statistics" table displays several likelihood-based measures of fit for the specified model. If you specify more than one model, these values correspond to the selected model. The table also includes the number of parameters that are used in the computation of the fit statistics. For more information about the computation of the fit statistics, see Table 19.5.

Parameter Estimates

The "Parameter Estimates" table displays the mean and covariance estimates for the specified model. If you specify more than one model, these values correspond to the selected model.

Mixing Estimates

The "Mixing Estimates" table displays the mixing probability estimates for the specified model. These values correspond to the τ_k that is presented in the section "Log-Likelihood Definitions" on page 1108. If you specify more than one model, these values correspond to the selected model.

Fit Summary

The "Fit Summary" table displays the fit statistics for a number of the fitted models. By default, this table shows the 10 models that have the best (smallest) value of the model selection criterion that is specified in the CRITERION= option. You can modify the number of models that the table includes by using the TOPMODELS= option. This table is displayed only if you specify more than one model.

Timing

The "Timing" table displays the amount of time (in seconds) that PROC MBC required to perform different tasks in the analysis.

OutputCasTables Table

The OutputCasTables table is a special table that has information about each CAS table that is created during a CAS action execution. The information for each CAS table consists of the CAS table name, the caslib in which the table resides, and the number of columns and rows in the CAS table. Because this table is not a typical ODS table that contains analytical results, you cannot include it in the *table-spec-list* in the DISPLAYOUT statement.

ODS Table Names

Each table that the MBC procedure creates has a name associated with it. You must use this name to refer to the table when you use the DISPLAY statement, the DISPLAYOUT statement, or ODS statements. These names are listed in Table 19.7.

Table 19.7 ODS Tables Produced by PROC MBC

Table Name	Description	Statement	Option
ConvergenceStatus	Convergence status of each model	PROC MBC	Default
FitStatistics	Fit statistics for specified or selected model	PROC MBC	Default
FitSummary	Comparison of fit statistics for all models	PROC MBC	Multiple values in COVSTRUCT=, NOISE=, or NCLUSTERS=
IterHistory	Iteration history from the EM algorithm	PROC MBC	ITHIST
MixingEstimates	Estimates for mixing proportions	PROC MBC	Default
ModelInfo	Information about the fitted model	PROC MBC	Default

Table 19.7 *continued*

Table Name	Description	Statement	Option
NObs	Information about observations	PROC MBC	Default
OutputCasTables	See the section "OutputCasTables Table" on page 1116	OUTPUT DISPLAYOUT	OUT=
ParameterEstimates	Mean and covariance estimates	PROC MBC	Default
Timing	Time used for different tasks	PROC MBC	Default

Examples: MBC Procedure

The examples in this chapter are available in the GitHub repository located at https://github.com/sascommunities/doc-supplement-casstat.

The examples in this section provide further applications of the MBC procedure and illustrate other aspects of its syntax.

Example 19.1: Storing and Scoring

(View the complete code for this example (mbcex1.sas) in the example repository.)

When you use PROC MBC to fit a model, you can use the STORE statement to save the model for later application to a new data table. For example, you can score new observations to find their posterior probabilities of cluster membership for each cluster in the model. This example uses PROC CAS to run the **mbcScore** action to do this subsequent scoring.

The following statements generate a data table that represents a scenario in product marketing. The different market segments are distinguished by different levels of certain financial characteristics.

Segment 1 is a generally younger group with a higher debt-to-income ratio, and it might be attracted by newer, more luxurious goods. When marketing to this group, you would like to highlight newer lines with higher price points. In contrast, segment 2, with a low debt-to-income ratio and an average age between that of segments 1 and 3, might represent a group of bargain hunters who would prefer a product that offers the best value. The customers in segment 3 have a higher average age and a moderate debt-to-income ratio, and they represent a group that is not interested in luxury items and not strongly motivated by price. Finally, segment 4 represents customers who do not fit into these groups and so might respond to a variety of offers.

These statements assume that your libref is named mylib, but you can substitute any appropriately defined libref.

```
data mylib.marketdata;
  label moninc  = 'Monthly Income'
        mondebt = 'Monthly Debt'
        tenancy = 'Months at Current Residence'
        ageyrs  = 'Age';
```

```
  call streaminit(869884);

  * -- Segment 1 : Young Spenders ------------ ;
  do j = 1 to 1000;
    moninc  = 12 + sqrt(2)    * rand('normal');
    mondebt =  6 + sqrt(1)    * rand('normal');
    tenancy = 18 + sqrt(2)    * rand('normal');
    ageyrs =  30 + sqrt(8)    * rand('normal');
    output;
  end;
  * -- Segment 2 : Middle-Aged Savers -------- ;
  do j = 1 to 3000;
    moninc  = 10 + sqrt(2.5)  * rand('normal');
    mondebt =  2 + sqrt(0.25) * rand('normal');
    tenancy = 24 + sqrt(3)    * rand('normal');
    ageyrs =  40 + sqrt(8)    * rand('normal');
    output;
  end;
  * -- Segment 3 : Comfortably Established --- ;
  do j = 1 to 6000;
    moninc  = 15 + sqrt(2)    * rand('normal');
    mondebt =  3 + sqrt(0.5)  * rand('normal');
    tenancy = 32 + sqrt(4)    * rand('normal');
    ageyrs =  50 + sqrt(5)    * rand('normal');
    output;
  end;
  * -- Segment 4: Defying Classification ----- ;
  do j = 1 to 300;
    moninc  = rand('uniform') * 15 +  5;
    mondebt = rand('uniform') *  9 +  0;
    tenancy = rand('uniform') * 30 + 10;
    ageyrs  = rand('uniform') * 40 + 20;
    output;
  end;
  drop j;
;
```

The existence of outliers is a good motivation to include a noise component in the model, because the outliers can otherwise distort the cluster structure in the model. You can use the NOISE=YES option to specify this directly. The following statements fit a range of models to these data. Each model includes a noise component.

```
proc mbc data=mylib.marketdata nclusters=(2 3 4 5) noise=YES seed=1389035719;
  var moninc mondebt tenancy ageyrs;
run;
```

Output 19.1.1 shows the "Fit Summary" table, indicating that the model with the best fit has three Gaussian clusters and a noise component.

Output 19.1.1 Mean and Covariance Estimates for Selected Model

The MBC Procedure

Model Selection Summary

Covariance Structure	Number of Clusters	Noise Component	Number of Parameters	-2 Log L	AIC	AICC	BIC
VVV	3	Y	46	170967	171059	171059	171392
VVV	4	Y	61	170965	171087	171088	171529
VVV	5	Y	76	170942	171094	171095	171644
VVV	2	Y	31	182253	182315	182315	182539

The goal of this analysis is to identify segments within the group of customers, and you can use the OUTPUT statement to label each customer with its most strongly associated segment. The MAXPOST option in the following OUTPUT statement includes the index of the component that has the highest posterior clustering probability for each observation. The indices start at zero for a model that includes a noise component:

```
proc mbc data=mylib.marketdata nclusters=(2 3 4 5) noise=YES;
  var moninc mondebt tenancy ageyrs;
  output out=mylib.mktscore copyvars=(moninc mondebt tenancy ageyrs) maxpost;
run;
```

In Output 19.1.2, the "Mixing Estimates" table indicates a mixture component with index 0; this is the noise component.

Output 19.1.2 Mixing Estimates for Selected Model

Mixing Probability Estimates for Selected Model

Mixing Component	Mixing Probability
0	0.02865
1	0.58248
2	0.09776
3	0.29110

Output 19.1.3 shows the parameter estimates for the selected model.

Output 19.1.3 Parameter Estimates for Selected Model

Parameter Estimates for Selected Model

Cluster	Variable	Mean	Covariance			
			moninc	mondebt	tenancy	ageyrs
1	moninc	15.00354	2.05211	0.01309	-0.00126	0.02988
	mondebt	3.00321		0.50819	0.05722	0.00485
	tenancy	31.99125			4.01148	-0.03269
	ageyrs	50.04533				5.03305
2	moninc	11.96829	2.12425	0.02417	0.07263	-0.09442
	mondebt	6.01248		1.06665	-0.00584	0.05854
	tenancy	17.99029			2.01793	0.03317
	ageyrs	29.99537				7.41317
3	moninc	9.99463	2.52433	-0.01639	0.05279	0.10067
	mondebt	1.99783		0.24534	0.00233	0.00226
	tenancy	24.01474			2.93710	0.05776
	ageyrs	39.91170				8.17630

The clusters that the procedure has identified align well with the segments that are generated in the example.

To market effectively to these different groups, you want to be able to assign a cluster membership score to each potential customer. You can use the STORE statement as follows to save the fitted model for later scoring:

```
proc mbc data=mylib.marketdata nclusters=(2 3 4 5) noise=YES seed=1389035719;
  var moninc mondebt tenancy ageyrs;
  store mylib.mktgroups;
run;
```

The mylib.mktgroups store contains a representation of the model that you can apply to a new set of customer data. You can use the **mbcScore** action to score new observations according to this saved model.

The following statements show a possible set of new data:

```
data mylib.newcusts;
input moninc    mondebt    tenancy     ageyrs;
datalines;
 7       5       20       35
12       6       20       30
18       4       31       52
10       4       25       37
;
```

The following statements show how to score the new set by using PROC CAS to invoke the **mbcScore** action. The **mbcScore** action call takes several parameters. You specify the new observations to score by using the **table** parameter. The saved model mylib.mktgroups is specified in the **restore** parameter. You use the **casOut** parameter to designate that the data table mylib.newScores will contain the scores for the new data. The **copyvars** parameter includes the listed input variables in the output data table. The **maxpost** parameter requests that the variable Group in the output set identify the cluster that has the highest posterior weight for each observation. Finally, you use the **nextclus** parameter to specify the root of the variable name for each cluster weight.

```
proc cas;
  action mbc.mbcScore /
    table={name='newcusts'}
    restore={name='mktgroups'}
    casOut={name='newScores', replace=true}
    copyvars={'moninc' , 'mondebt' , 'tenancy' , 'ageyrs'}
    maxpost='group'
    nextclus='cluswt';
  run;
quit;
```

When the scoring is finished, you can examine the clusterings in the data table mylib.newScores.

Output 19.1.4 shows the input data and the resulting weights and clustering.

Output 19.1.4 Weights and Scores for New Data

cluswt0	cluswt1	cluswt2	cluswt3	group	moninc	mondebt	tenancy	ageyrs
0.73350	0.00000	0.26650	0.00000	0	7	5	20	35
0.00096	0.00000	0.99904	0.00000	2	12	6	20	30
0.00176	0.99824	0.00000	0.00000	1	18	4	31	52
0.35676	0.00000	0.00001	0.64323	3	10	4	25	37

You can use the values in the Group variable to decide what action to take for each customer. You can also consider the strength of the association shown in the variables cluswt0 through cluswt3.

In this case, each observation has a strong association with its assigned cluster, but one observation has a noticeable affinity for the noise cluster as well.

Example 19.2: Adjusting Starting Values

(View the complete code for this example (mbcex2.sas) in the example repository.)

This example shows how you can use alternative starting values to influence the model fitting process.

The data in this example consist of physical measurements of 159 freshwater fish. The data set is available from Puranen (1917).

Each measurement includes species, weight, height, width, and three length variables. The values that are recorded for height and width are percentages of the third length variable.

The following DATA step creates the data table mylib.fish. These statements assume that your libref is named mylib, but you can substitute any appropriately defined libref.

```
data mylib.fish (drop=HtPct WidthPct);
   input Species Weight Length1 Length2 Length3 HtPct WidthPct @@;

   *** transform variables;
   if Weight <= 0 or Weight =. then delete;
   Weight3=Weight**(1/3);
   Height=HtPct*Length3/(Weight3*100);
   Width=WidthPct*Length3/(Weight3*100);
```

```
   Length1=Length1/Weight3;
   Length2=Length2/Weight3;
   Length3=Length3/Weight3;
   logLengthRatio=log(Length3/Length1);

   datalines;
1  242.0 23.2 25.4 30.0 38.4 13.4 1  290.0 24.0 26.3 31.2 40.0 13.8
1  340.0 23.9 26.5 31.1 39.8 15.1 1  363.0 26.3 29.0 33.5 38.0 13.3
1  430.0 26.5 29.0 34.0 36.6 15.1 1  450.0 26.8 29.7 34.7 39.2 14.2
1  500.0 26.8 29.7 34.5 41.1 15.3 1  390.0 27.6 30.0 35.0 36.2 13.4
1  450.0 27.6 30.0 35.1 39.9 13.8 1  500.0 28.5 30.7 36.2 39.3 13.7
1  475.0 28.4 31.0 36.2 39.4 14.1 1  500.0 28.7 31.0 36.2 39.7 13.3
1  500.0 29.1 31.5 36.4 37.8 12.0 1     .  29.5 32.0 37.3 37.3 13.6
1  600.0 29.4 32.0 37.2 40.2 13.9 1  600.0 29.4 32.0 37.2 41.5 15.0
1  700.0 30.4 33.0 38.3 38.8 13.8 1  700.0 30.4 33.0 38.5 38.8 13.5
1  610.0 30.9 33.5 38.6 40.5 13.3 1  650.0 31.0 33.5 38.7 37.4 14.8
1  575.0 31.3 34.0 39.5 38.3 14.1 1  685.0 31.4 34.0 39.2 40.8 13.7
1  620.0 31.5 34.5 39.7 39.1 13.3 1  680.0 31.8 35.0 40.6 38.1 15.1

   ... more lines ...

7   19.7 13.2 14.3 15.2 18.9 13.6 7   19.9 13.8 15.0 16.2 18.1 11.6
;
```

This example uses the variables height, width, and weight3 for analysis. Your goal is to find the model that best represents the clustering in these three variables.

The "Getting Started" example and "Example 19.1: Storing and Scoring" on page 1117 both show how to use PROC MBC to compare several models, but both examples rely on random starting values to begin the model fitting. PROC MBC provides two other ways to generate starting values: the *k*-means technique and predefined starting values.

When you specify INIT=KMEANS, the procedure uses the *k*-means technique to get starting values for the cluster weights, means, and covariances. This is equivalent to starting with the M-step in the EM algorithm. The *k*-means method might be a better method for starting values in some situations, particularly where the clusters are roughly spherical.

You use the INIT statement to specify variables that contain initial cluster assignment weights for each observation. The procedure uses these weights for the initial E-step in the EM algorithm. You might want to use the INIT statement in cases where you have prior knowledge of clustering and want to use that knowledge to influence the model fitting process.

The following statements use the default, or random start, method to generate initial values and to choose from several models that are distinguished by covariance structure:

```
proc mbc data=mylib.fish seed=9982346 covstruct=(eee eei eev eii evi evv vii vvi vvv)
  nclusters=3;
  var height width weight3;
run;
```

Output 19.2.1 summarizes the parameter estimates, mixing estimates, and fit statistics for the selected model, which has three clusters and uses the VVV covariance structure.

Output 19.2.1 Selected Model Using INIT=RANDOM Option

The MBC Procedure

Parameter Estimates for Selected Model

Cluster	Variable	Mean	Covariance Height	Width	Weight3
1	Height	1.21349	0.00430	0.00014005	0.03748
	Width	0.71109		0.00176	0.02778
	Weight3	6.27450			4.08257
2	Height	1.77964	0.00757	0.00124	0.08550
	Width	0.63619		0.00142	0.03318
	Weight3	7.65945			4.02232
3	Height	0.93893	0.00327	0.00015888	-0.11160
	Width	0.59277		0.00135	0.01157
	Weight3	5.88766			9.76032

Mixing Probability Estimates for Selected Model

Mixing Component	Mixing Probability
1	0.52334
2	0.28662
3	0.19003

Model Selection Summary

Covariance Structure	Number of Clusters	Noise Component	Number of Parameters	-2 Log L	AIC	AICC	BIC
EVV	3	N	27	-9.23633	44.76367	56.48460	127.28231
EEE	3	N	17	48.50814	82.50814	86.91102	134.46432
EEI	3	N	14	64.01230	92.01230	94.97004	134.79974
EVI	3	N	18	45.24593	81.24593	86.20245	136.25835
VVI	3	N	20	36.63419	76.63419	82.81066	137.75911
VVV	3	N	29	6.22721	64.22721	77.92800	152.85834
EEV	3	N	23	131.15440	177.15440	185.45515	247.44805
VII	3	N	14	1026.60551	1054.60551	1057.56325	1097.39295
EII	3	N	12	1075.29202	1099.29202	1101.45868	1135.96697

To see the effect of a different method of determining starting values, you can use the INIT=KMEANS option. The following statements use the *k*-means method of initialization:

```
proc mbc data=mylib.fish seed=9982346 init=kmeans
         covstruct=(eee eei eev eii evi evv vii vvi vvv)
  nclusters=3;
  var height width weight3;
run;
```

Output 19.2.2 summarizes the model selection process for the *k*-means method. The selected model uses the EVV covariance structure, and the fit statistics for other models are different from the result that uses the default random start method. In particular, the log-likelihood statistic for each of the top three models is a good deal worse with the *k*-means method than with the default method. This might be associated with

the fact that the three clusters are not really spherical, so the starting values that are determined using the *k*-means method might not be suitable in this situation.

Output 19.2.2 Model Selection Summary Using INIT=KMEANS Option

The MBC Procedure

Model Selection Summary

Covariance Structure	Number of Clusters	Noise Component	Number of Parameters	-2 Log L	AIC	AICC	BIC
EVV	3	N	27	-9.23633	44.76367	56.48460	127.28231
VVI	3	N	20	162.69679	202.69679	208.87326	263.82170
VVV	3	N	29	157.72594	215.72594	229.42673	304.35707
EEV	3	N	23	256.65424	302.65424	310.95500	372.94790
EEE	3	N	17	289.96205	323.96205	328.36493	375.91823
EVI	3	N	18	303.23126	339.23126	344.18779	394.24369
EEI	3	N	14	324.15413	352.15413	355.11188	394.94157
VII	3	N	14	1026.60549	1054.60549	1057.56324	1097.39293
EII	3	N	12	1075.29276	1099.29276	1101.45942	1135.96771

Output 19.2.3 shows the parameter estimates and mixing estimates for the selected model. The parameter estimates and mixing estimates are broadly similar to those for the default initialization method, but dissimilar enough to drive the difference in the selected model.

Output 19.2.3 Parameter Estimates and Mixing Estimates Using INIT=KMEANS Option

Parameter Estimates for Selected Model

Cluster	Variable	Mean	Covariance Height	Width	Weight3
1	Height	1.21348	0.00430	0.00014041	0.03749
	Width	0.71109		0.00176	0.02778
	Weight3	6.27437			4.08255
2	Height	0.93892	0.00327	0.00015841	-0.11160
	Width	0.59277		0.00135	0.01159
	Weight3	5.88795			9.76185
3	Height	1.77964	0.00757	0.00124	0.08550
	Width	0.63619		0.00142	0.03318
	Weight3	7.65945			4.02246

Mixing Probability Estimates for Selected Model

Mixing Component	Mixing Probability
1	0.52337
2	0.19001
3	0.28662

The remaining option for adjusting starting values is the INIT statement. Using the INIT statement requires you to specify variables that give initial weights for each observation.

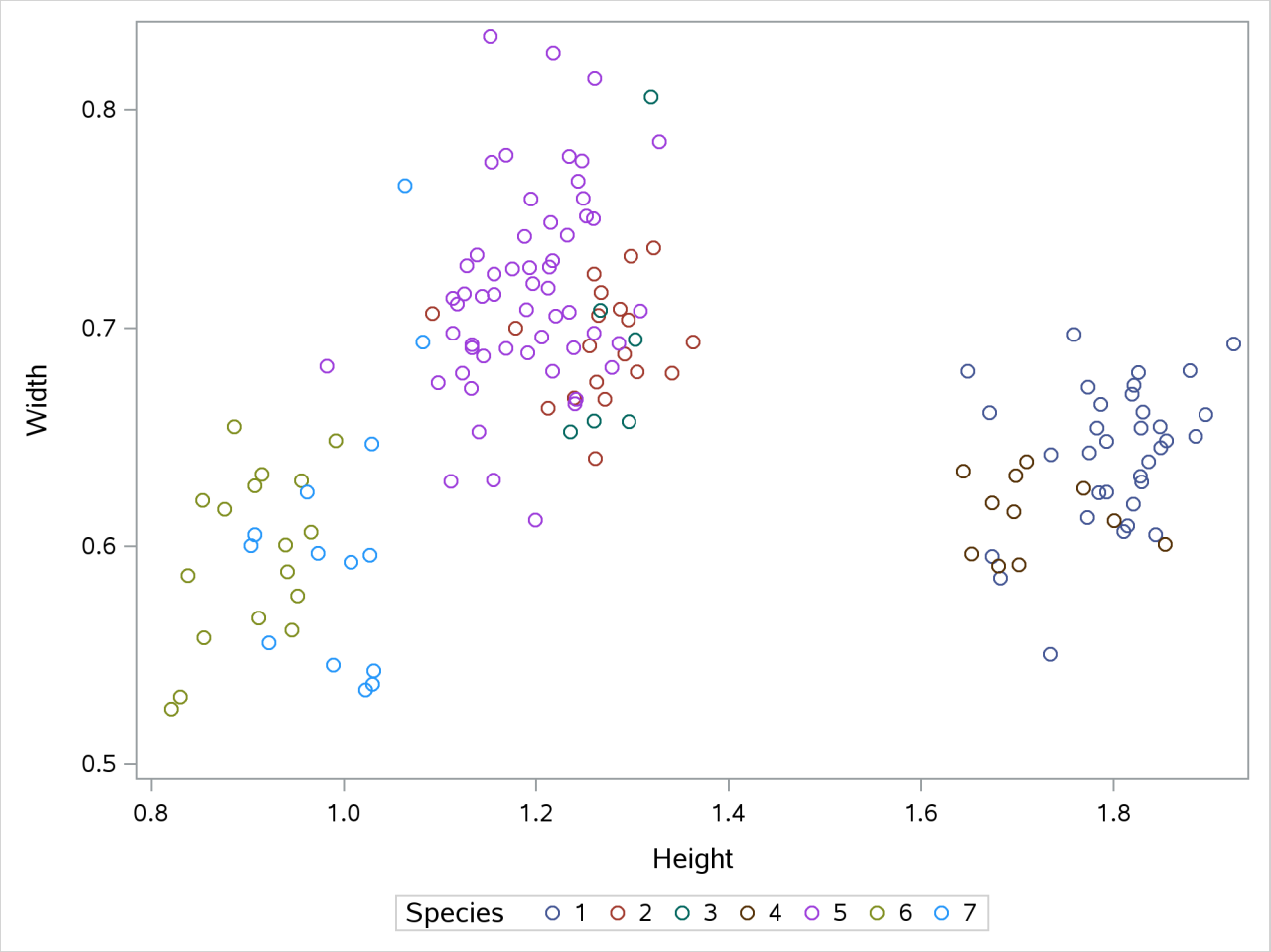
The following statements produce a scatter plot of the Height and Width variables, labeling each point with a

species number:

```
proc sgplot data=mylib.fish;
  scatter x=height y=width / group=species;
run; quit;
```

Output 19.2.4 suggests that the species encodes useful information about the clustering.

Output 19.2.4 Fish Measurement Data with Species Labels



You can use these visible clusterings to generate initialization variables. The following DATA step shows one way to generate initialization variables that reflect your observations:

```
data mylib.fishlbl;
   set mylib.fish;
  if      (species in (6,7)) then do; z1=1; z2=0; z3=0; end;
  else if (species in (1,4)) then do; z1=0; z2=1; z3=0; end;
  else                            do; z1=0; z2=0; z3=1; end;
;
```

PROC MBC uses the initialization values for each observation as a set of weights that drive the parameter estimates and mixing estimates for the first EM iteration. The following statements use these initialization values by including them in the desired order in the INIT statement:

```
proc mbc data=mylib.fishlbl
         covstruct=(eee eei eev eii evi evv vii vvi vvv)
  nclusters=3;
  var height width weight3;
  init z1 z2 z3;
run;
```

Output 19.2.5 summarizes the model selection process. The ordering of models does not match that from either of the previous results, but the selected model uses the same covariance structure (VVV) as the random start approach.

Output 19.2.5 Model Selection Summary Using INIT Statement

The MBC Procedure

Model Selection Summary

Covariance Structure	Number of Clusters	Noise Component	Number of Parameters	-2 Log L	AIC	AICC	BIC
VVV	3	N	29	-19.57464	38.42536	52.12615	127.05649
EVV	3	N	27	-9.23633	44.76367	56.48460	127.28230
EEV	3	N	23	12.65665	58.65665	66.95741	128.95031
EEE	3	N	17	48.50809	82.50809	86.91097	134.46427
EEI	3	N	14	64.01230	92.01230	94.97005	134.79974
EVI	3	N	18	45.24591	81.24591	86.20243	136.25834
VVI	3	N	20	36.63419	76.63419	82.81066	137.75911
VII	3	N	14	1026.60540	1054.60540	1057.56314	1097.39284
EII	3	N	12	1075.29045	1099.29045	1101.45712	1135.96540

Output 19.2.6 shows the parameter estimates and mixing estimates. The selected model matches the selected model from the random start approach, but with a different order for the clusters.

Output 19.2.6 Parameter Estimates and Mixing Estimates Using INIT Statement

Parameter Estimates for Selected Model

Cluster	Variable	Mean	Covariance Height	Width	Weight3
1	Height	0.93901	0.00394	0.00019642	-0.13391
	Width	0.59276		0.00162	0.01388
	Weight3	5.88904			11.69267
2	Height	1.77964	0.00529	0.00086794	0.05971
	Width	0.63619		0.00098863	0.02317
	Weight3	7.65945			2.80942
3	Height	1.21341	0.00472	0.00015780	0.04118
	Width	0.71108		0.00192	0.03042
	Weight3	6.27394			4.46530

Output 19.2.6 *continued*

Mixing Probability Estimates for Selected Model	
Mixing Component	Mixing Probability
1	0.18995
2	0.28662
3	0.52343

This approach results in different fit statistics for some of the other models that are considered. In this case, the use of initialization variables through the INIT statement does not select a model different from that selected using the default method. However, the difference in the fit statistics for the other models that are considered in the selection process shows the effect of the different initialization method.

References

Banfield, J. D., and Raftery, A. E. (1993). "Model-Based Gaussian and Non-Gaussian Clustering." *Biometrics* 49:803–821.

Böhning, D., Dietz, E., Schaub, R., Schlattmann, P., and Lindsay, B. G. (1994). "The Distribution of the Likelihood Ratio for Mixtures of Densities from the One-Parameter Exponential Family." *Annals of the Institute of Statistical Mathematics* 46:373–388.

Celeux, G., and Govaert, G. (1995). "Gaussian Parsimonious Clustering Models." *Pattern Recognition* 28:781–793.

Cook, R. D., and Weisberg, S. (1982). *Residuals and Influence in Regression*. New York: Chapman & Hall.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). "Maximum Likelihood from Incomplete Data via the EM Algorithm." *Journal of the Royal Statistical Society, Series B* 39:1–38.

Ghahramani, Z., and Hinton, G. E. (1997). *The EM Algorithm for Mixtures of Factor Analyzers*. Technical Report CRG-TR-96-1, University of Toronto.

McLachlan, G. J., and Krishnan, T. (1997). *The EM Algorithm and Extensions*. New York: John Wiley & Sons.

McNicholas, P. D. (2017). *Mixture Model-Based Classification*. Boca Raton, FL: CRC Press.

McNicholas, P. D., and Murphy, T. B. (2008). "Parsimonious Gaussian Mixture Models." *Statistics and Computing* 18:285–296.

Meng, X.-L., and Van Dyk, D. (1997). "The EM Algorithm—an Old Folk-Song Sung to a Fast New Tune." *Journal of the Royal Statistical Society, Series B* 59:511–567.

Puranen, J. (1917). "Fish Catch data set (1917)." Journal of Statistics Education Data Archive. Accessed February 17, 2022. http://jse.amstat.org/jse_data_archive.htm.

Yung, Y.-F. (1997). "Finite Mixtures in Confirmatory Factor-Analysis Models." *Psychometrika* 62:297–330.

Chapter 20
MODELMATRIX Procedure

Contents

Overview: MODELMATRIX Procedure **1130**
PROC MODELMATRIX Features 1130
PROC MODELMATRIX Compared with Other SAS Procedures 1131
Using CAS Sessions and CAS Engine Librefs 1131
Getting Started: MODELMATRIX Procedure **1132**
Syntax: MODELMATRIX Procedure **1135**
PROC MODELMATRIX Statement 1135
BY Statement 1136
CLASS Statement 1136
DISPLAY Statement 1137
DISPLAYOUT Statement 1138
EFFECT Statement 1139
FREQ Statement 1140
MODEL Statement 1141
OUTPUT Statement 1142
WEIGHT Statement 1142
Details: MODELMATRIX Procedure **1143**
Computational Method 1143
Displayed Output 1143
ODS Table Names 1144
Examples: MODELMATRIX Procedure **1145**
Example 20.1: Spline Effects and BY-Group Processing 1145
References **1150**

Overview: MODELMATRIX Procedure

The MODELMATRIX procedure creates a design matrix (matrix of covariates) that is associated with a user-specified MODEL statement and a user-specified data table in SAS Viya.

The design matrix is the fundamental component of any regression model; it is often expressed as the **X** matrix.

Each observation in the user-specified data table corresponds to a row in the design matrix. The columns of the design matrix are the parameters that are included in a regression model.

The models that PROC MODELMATRIX supports can contain main effects for continuous and classification variables and interaction effects of these variables. The models can also include constructed effects such as splines.

PROC MODELMATRIX Features

The MODELMATRIX procedure has the following main features:

- **Model specification**
  - supports multiple parameterizations for classification effects
  - supports any degree of interaction (crossed effects) and nested effects
  - supports a hierarchy among effects
  - provides a FREQ statement for grouped analysis
  - provides a WEIGHT statement for weighted analysis
- **Display and output**
  - produces an output data table that contains the design matrix
  - produces an output data table (OutDesignInfo) that names the parameters (columns) of the design matrix
  - produces output data tables that provide information about the number of observations read (NObs), model information (ModelInfo), dimensions information (Dimensions), class levelization details (ClassInfo), and effect details
  - supports custom prefixing for the names of the columns of the design matrix along with a default value (Col) if the prefix is not specified

Because the MODELMATRIX procedure runs in the CAS environment, it also does the following:

- enables you to run on a cluster of machines that distribute the data and the computations
- enables you to run in single-machine mode
- exploits all the available cores and concurrent threads

For information about how PROC MODELMATRIX uses threads, see the section "Multithreading" on page 102 in Chapter 2, "Shared Concepts."

PROC MODELMATRIX Compared with Other SAS Procedures

The MODELMATRIX procedure is the counterpart of the GLMMOD procedure (in SAS/STAT). It creates a design matrix from explanatory variables in a regression model. A number of other procedures, including the LOGISTIC and GLMSELECT procedures in SAS/STAT, enable you to create the design matrix as a by-product of other statistical analysis.

PROC MODELMATRIX produces a global design matrix across all BY groups. In other words, the columns of the design matrix represent the union of the parameters across all BY groups. The elements of any design row that do not belong to a BY group are replaced with missing values.

PROC MODELMATRIX Compared with the LOGISTIC Procedure

The functionality of the MODELMATRIX procedure closely resembles that of the LOGISTIC procedure when you specify the OUTDESIGNONLY option in the PROC LOGISTIC statement. The only difference between the results of the MODELMATRIX and LOGISTIC procedures is in the way invalid and missing variables in the FREQ and WEIGHT statements are treated. In PROC MODELMATRIX, rows that have invalid values (< 1 for the FREQ statement or ≤ 0 for the WEIGHT statement) are filled in with missing values. Observations that have missing covariates are treated in the same way. In short, all the columns of invalid observations are filled in with missing values. In contrast, PROC LOGISTIC does not treat invalid observations differently. If data are missing for a variable (including FREQ and WEIGHT variables), only the corresponding column (in the case of interval variables) or columns (in the case of categorical variables) are filled in with missing values. In addition, PROC LOGISTIC uses the EFFECT encoding for parameterization of the categorical variables by default, whereas PROC MODELMATRIX uses the GLM encoding.

Using CAS Sessions and CAS Engine Librefs

SAS Cloud Analytic Services (CAS) is the analytic server and associated cloud services in SAS Viya. This section describes how to create a CAS session and set up a CAS engine libref that you can use to connect to the CAS session. It assumes that you have a CAS server already available; contact your system administrator if you need help starting and terminating a server. This CAS server is identified by specifying the host on which it runs and the port on which it listens for communications. To simplify your interactions with this CAS server, the host information and port information for the server are stored as SAS option values that are retrieved automatically whenever this CAS server needs to be accessed. You can examine the host and port values for the server at your site by using the following statements:

```
proc options option=(CASHOST CASPORT);
run;
```

In addition to starting a CAS server, your system administrator might also have created a CAS session and a CAS engine libref for your use. You can define your own sessions and CAS engine librefs that connect to the CAS server as shown in the following statements:

```
cas mysess;
libname mylib cas sessref=mysess;
```

The CAS statement creates the CAS session named mysess, and the LIBNAME statement creates the mylib CAS engine libref that you use to connect to this session. It is not necessary to explicitly name the

CASHOST and CASPORT of the CAS server in the CAS statement, because these values are retrieved from the corresponding SAS option values.

If you have created the mysess session, you can terminate it by using the TERMINATE option in the CAS statement as follows:

```
cas mysess terminate;
```

For more information about the CAS and LIBNAME statements, see the section "Introduction to Shared Concepts" on page 6 in Chapter 2, "Shared Concepts."

Getting Started: MODELMATRIX Procedure

(View the complete code for this example at https://github.com/sascommunities/doc-supplement-casstat/tree/main/Examples/modelmatrixgs.sas.)

The Sashelp.Baseball data set contains salary and performance information for Major League Baseball players, excluding pitchers, who played at least one game in both the 1986 and 1987 seasons. The salaries (*Sports Illustrated,* April 20, 1987) are from the 1987 season, and the performance measures are from 1986 (Collier Books, *The 1987 Baseball Encyclopedia Update*). The following step displays (in Figure 20.1) the variables in the data set that are used in this example:

```
proc contents varnum data=sashelp.baseball
(KEEP = League Division nHits yrMajor);
   ods select position;
run;
```

Figure 20.1 Sashelp.Baseball Data Set

The CONTENTS Procedure

Variables in Creation Order

#	Variable	Type	Len	Label
4	nHits	Num	8	Hits in 1986
9	YrMajor	Num	8	Years in the Major Leagues
16	League	Char	8	League at the End of 1986
17	Division	Char	8	Division at the End of 1986

Assume that you want to predict the logSalary by using the categorical variables League and Division and the interaction of the nHits and YrMajor variables. The goal is to create a design matrix that represents this regression model. This example shows how you can use PROC MODELMATRIX for this purpose. Because the variation in salaries is much greater for the higher salaries, a log transformation is first applied to the salaries.

NOTE: Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections "Using CAS Sessions and CAS Engine Librefs" on page 7 and "Loading a SAS Data Set onto a CAS Server" on page 7 in Chapter 2, "Shared Concepts."

You can load the Sashelp.Baseball data set by using the following DATA step:

```
data mylib.baseball;
   set sashelp.baseball;
   index = _N_;
run;
```

These statements assume that your libref is named mylib, as in the section "Using CAS Sessions and CAS Engine Librefs" on page 1131, but you can substitute any appropriately defined libref.

The following statements define the described model, create the design matrix, and store the design matrix in the output table mylib.designMat:

```
proc modelmatrix data=mylib.baseball;
  class League Division;
  model logSalary = League Division nHits*yrMajor;
  output out = mylib.designMat copyvar=index;
run;
```

The output from this analysis is presented in Figure 20.2 through Figure 20.4.

Figure 20.2 Number of Observations, Dimensions, and Column Names of the Design Matrix

The MODELMATRIX Procedure

Number of Observations Read	322
Number of Observations Used	263

Dimensions	
Number of Effects	4
Number of Parameters	6

Column Names of the Design Matrix	
Name	**Parameter**
Col1	Intercept
Col2	League American
Col3	League National
Col4	Division East
Col5	Division West
Col6	nHits * YrMajor

Figure 20.2 displays the "Number of Observations," "Dimensions," and the "Column Names of the Design Matrix" tables. The "Number of Observations" table shows that only 263 of the 322 observations in the input data are used, because some of the observations contain missing data. When you specify effects that contain classification variables, the number of parameters is usually larger than the number of effects. The "Dimensions" table shows the number of effects and the number of parameters that are considered.

The "Column Names of the Design Matrix" table lists all the parameters that are included in the design matrix and indicates the column name that is associated with each parameter. As presented in Figure 20.2, the model includes a total of six parameters (two categorical variables with two levels each, one interaction variable, and the intercept), and therefore the design matrix has six columns. Because the OUTPUT statement does not specify a PREFIX= option, the default prefix (col) is used for the column names. For more information about how to change the column names, see the PREFIX= option and Examples.

The "Model Information" table in Figure 20.3 shows the main components of the model.

Figure 20.3 Model Information Table

The MODELMATRIX Procedure

Model Information	
Data Source	BASEBALL
Response Variable	logSalary

To be able to inspect the created design matrix, you first need to transfer the output data table (mylib.designMat) from the computing cluster to a local data table (designMat) and then use the PRINT procedure. Note that PROC MODELMATRIX does not preserve the order of the rows in the original data table. Therefore the design matrix that is produced is sorted before printing. The following statements show how you can use a DATA step along with a PROC PRINT step to display the first 15 rows of the transferred design matrix:

```
data designMat;
 set mylib.designMat;
run;

proc sort data=designMat;
 by index;
run;

proc print data = designMat(obs=15 drop=index) label;
label COL1='Intercept'
      COL2='League American'
      COL3='League National'
      COL4='Division East'
      COL5='Division West'
      COL6='nHits * YrMajor';
run;
```

Figure 20.4 First 15 Rows of the Design Matrix Stored in the designMat Data Table

Obs	Intercept	League American	League National	Division East	Division West	nHits * YrMajor
1	.	.	.	.	.	.
2	1	0	1	0	1	1134
3	1	1	0	0	1	390
4	1	0	1	1	0	1551
5	1	0	1	1	0	174
6	1	1	0	0	1	1859
7	1	0	1	1	0	74
8	1	1	0	0	1	219
9	1	0	1	0	1	162
10	1	1	0	1	0	1196
11	1	1	0	1	0	1590
12	1	0	1	0	1	477
13	1	0	1	1	0	452
14	1	1	0	1	0	360
15	1	0	1	1	0	559

Syntax: MODELMATRIX Procedure

The following statements are available in the MODELMATRIX procedure:

PROC MODELMATRIX < *options* > **;**
BY *variables* **;**
CLASS *variable* < **(***options***)** >. . . < *variable* < **(***options***)** > > < / *global-options* > **;**
DISPLAY < *table-list* > < / *options* > **;**
DISPLAYOUT *table-spec-list* < / *options* > **;**
EFFECT *name* **=** *effect-type* **(***variables* < / *options* >**) ;**
FREQ *variable* **;**
MODEL *dependent* **=** < *effects* > < / *model-options* > **;**
OUTPUT OUT=*libref.data-table* < *options* > **;**
WEIGHT *variable* **;**

The PROC MODELMATRIX statement, a single MODEL statement, and a single OUTPUT statement are required. All other statements are optional.

The CLASS statement can appear multiple times. If a CLASS statement is specified, it must precede the MODEL statement.

The rest of this section provides detailed syntax information about each of the preceding statements, beginning with the PROC MODELMATRIX statement. The remaining statements are described in alphabetical order.

PROC MODELMATRIX Statement

PROC MODELMATRIX < *options* > **;**

The PROC MODELMATRIX statement invokes the procedure. Table 20.1 summarizes the options in the PROC MODELMATRIX statement by function.

Table 20.1 PROC MODELMATRIX Statement Options

Option	Description
Basic Options	
DATA=	Specifies the input data table

You can specify the following *options*:

DATA=*libref.data-table*
names the input data table for PROC MODELMATRIX to use. The default is the most recently created data table. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to the data, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME

statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 1131.

data-table specifies the name of the input data table.

BY Statement

BY *variables* **;**

You can specify a BY statement in PROC MODELMATRIX to obtain separate analyses of observations in groups that are defined by the values of the BY variables. If you specify more than one BY statement, only the last one specified is used. For more information, see the discussion of BY-group processing in the "Grouping Data" section of *SAS Programmers Guide: Essentials*.

CLASS Statement

CLASS *variable* < **(***options***)** >. . . < *variable* < **(***options***)** > > < / *global-options* > **;**

The CLASS statement names the classification variables to be used as explanatory variables in the analysis. Table 20.2 summarizes the values that you can use for either an *option* or a *global-option*. The options are fully documented in the section "CLASS Statement" on page 9 in Chapter 2, "Shared Concepts."

Table 20.2 CLASS Statement Options

Option	Description
DESCENDING	Reverses the sort order
MISSING	Treats missing values as valid levels
ORDER=	Specifies the sort order for the levels
PARAM=	Specifies the parameterization of the variable
REF=	Specifies the reference level of the variable
SPLIT	Splits levels of CLASS variables into independent effects

DISPLAY < *table-list* > < / *options* > **;**

DISPLAY Statement

DISPLAY < *table-list* > < / *options* > **;**

The DISPLAY statement enables you to specify a list of display tables to display or exclude. This statement is similar to the ODS SELECT, ODS EXCLUDE, and ODS TRACE statements. However, the DISPLAY statement can improve performance when a large number of tables could be generated (such as in BY-group processing). The procedure processes the DISPLAY statement on a CAS server and thus sends only a subset of ODS tables to the SAS client. Because ODS statements are processed on a SAS client, first all the generated display tables are sent to the client, and then the client creates a subset.

If you use both DISPLAY and ODS statements together, the DISPLAY statement takes precedence over the ODS statements. Note that the ODS EXCLUDE statement processes tables that are sent to the client after they have been filtered by the DISPLAY statement. In some cases, it might appear that the ODS EXCLUDE statement is taking precedence because it can further filter the tables. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

You can specify the *table-list* as a list of table names, paths, partial pathnames, and regular expressions.

The table names that you can specify are listed in the section "ODS Table Names" on page 1144. A path is a table name that is prefixed with dot-separated grouping information. For example, a SelectionSummary table that a procedure produces during a selection routine might have the path *Bygroup1.Summary.SelectionSummary*. A partial pathname does not include all groups; for example, *SelectionSummary* and *Summary.SelectionSummary* are partial pathnames for *Bygroup1.Summary.SelectionSummary*.

When you specify a table name or partial pathname, all display tables whose paths end in the specified name are selected for display or exclusion. For example, both *SelectionSummary* and *Summary.SelectionSummary* select *Bygroup1.Summary.SelectionSummary*.

A regular expression is enclosed in forward slashes (/). For example, specifying "/tions/" selects all pathnames that contain the substring "tions"; in particular, the *Bygroup1.Summary.SelectionSummary* table is selected. Specifying "!/tions/" selects all pathnames that do not contain the substring "tions"; in particular, the *Bygroup1.Summary.SelectionSummary* table is not selected.

You can specify the following *options* after a slash (/):

CASESENSITIVE
performs a case-sensitive comparison of table names in the *table-list* to display table names when tables are subsetted for display. To preserve case, you must enclose table names in the *table-list* in quotation marks.

EXCLUDE
displays all display tables except those that you specify in the *table-list*.

EXCLUDEALL
suppresses display of all tables. This option takes precedence over the other options.

TRACE
displays the display table names, labels, and paths.

DISPLAYOUT Statement

DISPLAYOUT *table-spec-list* < / *options* > **;**

The DISPLAYOUT statement enables you to create output tables from your displayed output. This statement is similar to the ODS OUTPUT statement. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

The *table-spec-list* specifies a list of output tables to create. Each entry in the list has either a *key*=*value* format or a *key* format:

key=*value* specifies *key* as the ODS table name, path, or partial pathname, and specifies *value* as the output table name.

key specifies *key* as the ODS table name and also as the output table name.

The ODS table names that you can specify are listed in the section "ODS Table Names" on page 1144. You cannot specify the ODS table named OutputCasTables in the *table-spec-list*.

Table names and partial pathnames are discussed under the DISPLAY statement. The DISPLAYOUT statement does not support regular expressions.

You can specify the following *options* after a slash (/):

INCLUDEALL
creates output tables for all display tables. The name of the created output table is the same as the name of the corresponding display table. If you specify this option, the *table-spec-list* specification is ignored.

NOREPLACE
does not replace any existing output table of the same name.

REPEATED
replicates all output tables on all nodes.

EFFECT Statement

EFFECT *name*=*effect-type* **(***variables* < / *options* >**) ;**

The EFFECT statement enables you to construct special collections of columns for design matrices. These collections are referred to as *constructed effects* to distinguish them from the usual model effects that are formed from continuous or classification variables, as discussed in the section "GLM Parameterization of Classification Variables and Effects" on page 67 in Chapter 2, "Shared Concepts."

You can specify the following *effect-types*:

COLLECTION specifies a collection effect that defines one or more variables as a single effect that has multiple degrees of freedom. The variables in a collection are considered as a unit for purposes of estimation and inference.

MULTIMEMBER | MM specifies a multimember classification effect whose levels are determined by one or more variables that appear in a CLASS statement.

POLYNOMIAL | POLY specifies a multivariate polynomial effect in the specified numeric variables.

SPLINE specifies a regression spline effect whose columns are univariate spline expansions of one or more variables. A spline expansion replaces the original variable with an expanded or larger set of new variables.

Table 20.3 summarizes the *options* available in the EFFECT statement.

Table 20.3 EFFECT Statement Options

Option	Description
Collection Effects Option	
DETAILS	Displays the constituents of the collection effect
Multimember Effects Options	
DETAILS	Displays the levels of the multimember effect
NOEFFECT	Specifies that observations whose levels are all missing for the multimember variables should have values of 0 in the corresponding design matrix columns
STDIZE	Standardizes the design matrix entries so that each observation has a sum of 1
WEIGHT=	Specifies the weight variable for the contributions of each classification effect
Polynomial Effects Options	
DEGREE=	Specifies the degree of the polynomial
DETAILS	Displays details of the specified polynomial
MDEGREE=	Specifies the maximum degree of any variable in a term of the polynomial
NOSEPARATE	Treats the polynomial as a single effect that has multiple degrees of freedom
STANDARDIZE=	Specifies centering and scaling suboptions for the variables that define the polynomial

Table 20.3 *continued*

Option	Description
Spline Effects Options	
BASIS=	Specifies the type of basis (B-spline basis or truncated power function basis) for the spline effect
DATABOUNDARY	Uses the extremes of the data as boundary knots for a B-spline basis
DEGREE=	Specifies the degree of the spline effect
DETAILS	Displays the knots and locations for each spline basis function
KNOTMAX=	Requests equally spaced right-side boundary knots that start at the variables' maximum and end at the specified KNOTMAX= value
KNOTMETHOD=	Specifies how to construct the knots for the spline effect
KNOTMIN=	Requests equally spaced left-side boundary knots that start at the specified KNOTMIN= value and end at the variables' minimum value
NATURALCUBIC	Specifies a natural cubic spline basis for the spline effect
SEPARATE	Treats the spline basis for each variable as a separate effect when multiple variables are specified
SPLIT	Treats each design matrix column as a separate effect

For more information about the syntax of these *effect-types* and how columns of constructed effects are computed, see the section "EFFECT Statement" on page 18 in Chapter 2, "Shared Concepts."

FREQ Statement

FREQ *variable* **;**

The *variable* in the FREQ statement identifies a numeric variable in the data set that contains the frequency of occurrence of each observation. PROC MODELMATRIX treats each observation as if it appears f times, where f is the value of the *variable* for the observation. If f is not an integer, it is truncated to an integer. If f is less than 1 or missing, all the columns of the design matrix for that observation are filled in with missing values. When the FREQ statement is omitted, each observation is assigned a frequency of 1.

MODEL Statement

MODEL *dependent* **=** < *effects* > / < *model-options* > **;**

The MODEL statement names the dependent variable and the explanatory effects, including covariates, main effects, interactions, and nested effects. If you omit the explanatory effects, the procedure produces a single column of ones.

After the keyword MODEL, the dependent (response) variable is specified, followed by an equal sign. The explanatory effects follow the equal sign.

For information about constructing the model effects, see the section "Specification and Parameterization of Model Effects" on page 64 in Chapter 2, "Shared Concepts."

The *model-options* control other aspects of model formation and inference. Table 20.4 summarizes these options.

Table 20.4 MODEL Statement Options

Option	Description
Model Options	
INFORMATIVE	Models missing values by using extra indicator variables
NOINT	Suppresses the intercept

Model Options

You can specify the following *model-options* in the MODEL statement after a slash (/):

INFORMATIVE
models missing values by using extra model effects. These effects consist of dummy variables that take the value 1 when the value of a continuous model variable involved in the effect is missing, and take the value 0 otherwise. The missing value in the original model effect is replaced by the average value of the effect for the nonmissing values. For continuous-by-class effects, such as A*x, where A is a classification variable and x is a continuous variable, informative missingness creates multiple dummy columns and substitutes the effect mean of x that corresponds to the respective level of A. Missing values for classification variables are treated as valid levels. For more information about informative missingness, see the section "Informative Missingness" on page 99 in Chapter 2, "Shared Concepts."

NOINT
suppresses the intercept term that is otherwise included in the model.

OUTPUT Statement

OUTPUT OUT=*libref.data-table* < *options* > **;**

The OUTPUT statement creates an output table that contains the design matrix, which is computed after the model parameters are created (and levelized when CLASS variables are included). The variables in the input data table are *not* included in the output data table, in order to avoid data duplication for large data tables; however, variables that you specify in the COPYVARS= option are included.

You must specify the following option:

OUT=*libref.data-table*
names the output data table for PROC MODELMATRIX to use. You must specify this option before any other options. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to where the data table is to be stored, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 1131.

data-table specifies the name of the output data table.

You can also specify the following syntax elements:

PREFIX=*name*
enforces the column names of the design matrix to be a user-defined prefix appended to a sequence of one-based indices in the OUTDESIGN= data set. The default prefix is Col and the column name is formed by appending the column number to the prefix, so that by default the columns are named Col1, Col2, and so on.

COPYVAR=*variable*

COPYVARS=(*variables***)**
transfers one or more *variables* from the input data table to the output data table and appends them to the right end of the design matrix in the specified order.

WEIGHT Statement

WEIGHT *variable* **;**

The *variable* in the WEIGHT statement is used as a weight. Observations that have nonpositive or missing weights are considered missing in the produced design matrix. If a WEIGHT statement is not included, all observations that are used are assigned a weight of 1.

Details: MODELMATRIX Procedure

Computational Method

Multithreading

The MODELMATRIX procedure uses the parallel and multithreaded capabilities of the host environment. When the input data table is distributed across multiple nodes, the resulting design matrix is written in a distributed fashion. The user can specify the number of threads. For more information about how PROC MODELMATRIX uses threads, see the section "Multithreading" on page 102 in Chapter 2, "Shared Concepts."

Displayed Output

The following sections describe the output that PROC MODELMATRIX produces. The output is organized into various tables, which are discussed in their order of appearance.

Column Names of the Design Matrix

The "OutDesignInfo" table displays the column names (valid SAS names) that are assigned to the model parameters (which can include complex parameters from interaction effects and levelization of the classification variables). Unless the user specifies a different value for the PREFIX= option, the default prefix (Col) is used for naming the columns; hence the column names are Col1, Col2, and so on. The "OutDesignInfo" table lists the names of the columns in the design matrix that correspond to each model parameter.

Number of Observations

The "Number of Observations" table displays the number of observations that are read from the input data table and the number of observations that are used in constructing the design matrix. If you specify a FREQ statement, this table displays the sum of frequencies that are read and used. If you specify a WEIGHT statement, the table displays the sum of $f_i w_i$ that are read and used.

Class Level Information

The "Class Level Information" table lists the levels of every variable that you specify in the CLASS statement. You can adjust the order of the CLASS variable levels by specifying the ORDER= option in the CLASS statement.

If the classification variables use a nonsingular parameterization, the "Class Level Information" table also displays the reference value for each variable. This table is produced for each BY group separately.

Dimensions

The "Dimensions" table displays the number of effects and the number of parameters for each BY group.

Model Information

The "Model Information" table lists a summary of the model variables. It includes the data source name and the name of the response variables. If you specify the FREQ and WEIGHT statements, the names of the FREQ and WEIGHT variables are displayed in the "Model Information" table.

Effect Details

If you specify the DETAILS option in the EFFECT statement, a table is displayed that lists all constructed effects in the model.

ODS Table Names

Each table that the MODELMATRIX procedure creates has a name associated with it. You must use this name to refer to the table when you use ODS statements. These names are listed in Table 20.5.

NOTE: The EFFECT statement also create tables which are not listed in this section. For information about these tables, see "ODS Table Names" on page 33 in Chapter 2, "Shared Concepts."

Table 20.5 ODS Tables Produced by PROC MODELMATRIX

Table Name	Description	Statement	Option
ClassInfo	Level information from the CLASS statement	CLASS	Default
Dimensions	Model dimensions	Default	
ModelInfo	Information about the modeling environment	Default	
NObs	Number of observations read and used	Default	
OutDesignInfo	Column names of the design matrix	Default	
OutputCASTables	A special ODS table that has information about the tables that are created	OUTPUT DISPLAYOUT	Default
Timing	Absolute and relative times for tasks performed by the procedure	Default	

Examples: MODELMATRIX Procedure

The examples in this chapter are available in the GitHub repository located at https://github.com/sascommunities/doc-supplement-casstat.

NOTE: Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections "Using CAS Sessions and CAS Engine Librefs" on page 7 and "Loading a SAS Data Set onto a CAS Server" on page 7 in Chapter 2, "Shared Concepts."

Example 20.1: Spline Effects and BY-Group Processing

(View the complete code for this example (modelmatrixex1.sas) in the example repository.)

This example showcases several features of the MODELMATRIX procedure. It demonstrates the use of classification variables in a MODEL statement along with the use of interaction and SPLINE effects in a MODEL statement. In addition, it demonstrates how the procedure handles BY-group processing for a particular model.

The following DATA step produces analysis data that contain a variable that has 20 observations. These statements assume that your libref is named mylib, as in the section "Using CAS Sessions and CAS Engine Librefs" on page 1131, but you can substitute any appropriately defined libref.

```
data mylib.sample;
call streaminit(1);
    do byVar=1 to 2;
      do i=1 to 10;
               if byVar=1 then c=rand("Integer",0,3);
          else  if byVar=2 then c=rand("Integer",0,1);
          x=rand("Normal");
          y=1;
          wgt = 1;
          if i=5 and byVar=2 then frq  = -1; else frq=1;
          index + 1;
          output;
      end;
    end;
run;
```

The preceding DATA step generates a data table (mylib.sample) that contains 20 observations. It has two continuous variables (x and y) and a single classification variable (c), which can have a maximum of four unique values for 10 observations and a maximum of two unique values for the rest of the observations. Also, a BY variable named byVar is defined that divides the data table into two equally sized BY groups, each of which has 10 observations. Also included are the frq and wgt variables that are going to be used as the FREQ and WEIGHT variables, respectively. For simplicity, all the WEIGHT variables are assigned a value of 1. The same applies to FREQ variables, with the exception of the observation where INDEX=15, which

has an invalid FREQ value (that is, –1). This invalid FREQ value demonstrates how the MODELMATRIX procedure treats observations that have invalid FREQ values.

The following DATA step and PROC PRINT statement show the contents of the mylib.sample data table. The output of these statements is displayed in Output 20.1.1.

```
data sample;
  set mylib.sample;
run;

proc print data=sample;
run;
```

Output 20.1.1 Contents of the mylib.sample Data Table

Obs	byVar	i	c	x	y	wgt	frq	index
1	1	1	3	0.46319	1	1	1	1
2	1	2	3	0.00741	1	1	1	2
3	1	3	2	-0.06703	1	1	1	3
4	1	4	3	1.52070	1	1	1	4
5	1	5	2	0.51495	1	1	1	5
6	1	6	3	1.00574	1	1	1	6
7	1	7	3	0.36049	1	1	1	7
8	1	8	3	-1.75802	1	1	1	8
9	1	9	2	1.02322	1	1	1	9
10	1	10	1	0.33004	1	1	1	10
11	2	1	1	1.48699	1	1	1	11
12	2	2	0	-0.31012	1	1	1	12
13	2	3	1	0.60637	1	1	1	13
14	2	4	0	1.96293	1	1	1	14
15	2	5	0	-1.21157	1	1	-1	15
16	2	6	1	-1.22244	1	1	1	16
17	2	7	1	0.53368	1	1	1	17
18	2	8	1	1.27392	1	1	1	18
19	2	9	0	-0.99679	1	1	1	19
20	2	10	0	-1.55454	1	1	1	20

To demonstrate the capabilities of the MODELMATRIX procedure, the following code includes BY-group processing, CLASS variables, the interaction effect, and the SPLINE effect. It also demonstrates how you can change the name of the output data table by using the PREFIX= option. Moreover, the OUTPUT statement enables you to include the input data table variables as additional columns in the output data table by using the COPYVARS= option.

```
proc modelmatrix data=mylib.sample nthreads=1;
  by byVar;
  class c;
  freq frq;
  weight wgt;
  effect spl= spline(x);
  model y = x c spl x*c;
  output out=mylib.designMat prefix=param copyVars=(c frq index);
run;
```

The output tables are displayed in Output 20.1.2, Output 20.1.3, and Output 20.1.4.

Output 20.1.2 Model Information and Column Names of the Design Matrix

The MODELMATRIX Procedure

Model Information	
Data Source	SAMPLE
Response Variable	y
Frequency Variable	frq
Weight Variable	wgt

Column Names of the Design Matrix	
Name	**Parameter**
param1	Intercept
param2	x
param3	c 1
param4	c 2
param5	c 3
param6	c 0
param7	spl 1
param8	spl 2
param9	spl 3
param10	spl 4
param11	spl 5
param12	spl 6
param13	spl 7
param14	x * c 1
param15	x * c 2
param16	x * c 3
param17	x * c 0

The "Column Names of the Design Matrix" table in Output 20.1.2 shows that the defined model creates a total of 17 parameters across the two BY groups. Therefore, the design matrix has 17+3 columns (the additional columns are reserved for the variables c, frq, and index that you append to the design matrix by using the COPYVARS= option).

Apart from the intercept, the continuous variable x is included as a main effect. However, because c is defined as a classification variable, all of its unique values (0–3) are considered to be separate parameters (a total of four). Output 20.1.2 shows that the SPLINE effect adds seven more parameters to the model. Finally, the interaction of x and c ($x * c$) adds four more parameters.

Also shown in Output 20.1.2 is the "Model Information" table, which displays a brief summary of the response variable, the FREQ variable, and the WEIGHT variable.

The rest of the output tables depend on the BY groups. Therefore, they are produced separately and are shown in Output 20.1.3 and Output 20.1.4.

Output 20.1.3 Number of Observations, Class Level Information, and Dimensions Tables for BY Group 2

The MODELMATRIX Procedure

byVar=1

Number of Observations Read	10
Number of Observations Used	10
Sum of Frequencies Read	10
Sum of Frequencies Used	10
Sum of Weights Read	10
Sum of Weights Used	10

byVar=1

Class Level Information

Class	Levels	Values
c	3	1 2 3

byVar=1

Dimensions	
Number of Effects	5
Number of Parameters	15

Output 20.1.4 Number of Observations, Class Level Information, and Dimensions Tables for BY Group 2

The MODELMATRIX Procedure

byVar=2

Number of Observations Read	10
Number of Observations Used	9
Sum of Frequencies Read	9
Sum of Frequencies Used	9
Sum of Weights Read	9
Sum of Weights Used	9

byVar=2

Class Level Information

Class	Levels	Values
c	2	0 1

byVar=2

Dimensions	
Number of Effects	5
Number of Parameters	13

For each BY group, PROC MODELMATRIX displays the number of observations that are read and used. Because the FREQ and WEIGHT variables are defined, the procedure displays the number of FREQ and WEIGHT variables that are read and used. You can see that BY group 2 (Output 20.1.4) has one invalid

observation (the observation where INDEX=15). Therefore, the number of FREQ variables that are used is 9, which is different from BY group 1, where all FREQ variables have a value of 1.

The "Dimensions" table shows the number of effects and parameters that are considered within each BY group. Notice that the number of considered effects for both BY groups is equal to five. However, neither BY group includes all the parameters listed in the OutDesignInfo table, so for all BY groups, the number of parameters is less than 17.

This fact can be seen in the "Class Level Information" tables shown in Output 20.1.3 and Output 20.1.4. These tables show the levelization of the classification variables within each BY group. You can see that none of the BY groups include all levels of c (that is, 0, 1, 2, and 3), which is why the number of parameters is less than 17 for all BY groups.

PROC MODELMATRIX constructs a global design matrix. Therefore, it constructs a table that includes the union of all parameters in each BY group. To inspect the design matrix, follow the same steps that are given in the previous example. The constructed table with 20 rows can be displayed by using the following statements. The output is displayed in Output 20.1.5.

```
data work.designMat;
  set mylib.designMat;
run;

proc print data=designMat;
  format BEST 5. param1-param17;
  var index c frq param1-param7 param17;
  where
    (index < 11) or
    (index = 15);
run;
```

Output 20.1.5 Rows with index Values of 1–10 and Row with index Value of 15 from Design Matrix

Obs	index	c	frq	param1	param2	param3	param4	param5	param6	param7	param17
1	1	3	1	1	0.46319	0	0	1	.	0.00000	.
2	2	3	1	1	0.00741	0	0	1	.	0.00000	.
3	3	2	1	1	-0.06703	0	1	0	.	0.00000	.
4	4	3	1	1	1.52070	0	0	1	.	0.00000	.
5	5	2	1	1	0.51495	0	1	0	.	0.00000	.
6	6	3	1	1	1.00574	0	0	1	.	0.00000	.
7	7	3	1	1	0.36049	0	0	1	.	0.00000	.
8	8	3	1	1	-1.75802	0	0	1	.	0.16667	.
9	9	2	1	1	1.02322	0	1	0	.	0.00000	.
10	10	1	1	1	0.33004	1	0	0	.	0.00000	.
15	15	0	-1	.	.	.	.	.	.	.	.

Output 20.1.5 displays rows of the design matrix whose index value is 1–10 and the row whose index value is 15. Only the index, c, frq, param1–param7, and param17 variables are shown. As expected, the generated data table contains 20 rows associated with each observation and 20 columns (17 parameters plus three columns for the input variables c, frq, and index). You can see that the column names are param1 through param17, based on the PREFIX= option specified in the OUTPUT statement.

The "Column Names of the Design Matrix" table lists the model parameters with their corresponding column

names in the design matrix. Columns param3 through param6 contain all levels of the c variable. The rows in the design matrix whose index value is less than 11 belong to the first BY group (byVar=1), and according to the "Class Information" table, level $c3$ does not exist in either BY group. This observation can be confirmed by the fact that the param6 and param17 columns of the design matrix (which are associated with the $c3$ and $x * c3$ parameters, respectively) are empty.

You can also see that the row of the design matrix whose index value is 15 is all blanks. This is because the value of the defined FREQ variable for this observation is invalid and equal to –1 (see the frq column in the design matrix), and therefore this observation is not considered when the design matrix is created.

References

Collier Books (1987). *The 1987 Baseball Encyclopedia Update*. New York: Macmillan.

Time Inc. (1987). "What They Make." *Sports Illustrated* (April 20): 54–81.

Chapter 21
MVOUTLIER Procedure

Contents

Overview: MVOUTLIER Procedure **1152**
PROC MVOUTLIER Features 1152
PROC MVOUTLIER Compared with Other SAS Software 1153
Using CAS Sessions and CAS Engine Librefs 1155
Getting Started: MVOUTLIER Procedure **1156**
Syntax: MVOUTLIER Procedure **1162**
PROC MVOUTLIER Statement 1162
BY Statement 1169
DISPLAY Statement 1169
DISPLAYOUT Statement 1170
ID Statement 1171
OUTPUT Statement 1171
VAR Statement 1173
Details: MVOUTLIER Procedure **1173**
Displayed Output 1173
Model Information 1173
Number of Observations 1174
Number of Variables 1174
Eigenvalues for Initial Robust Subspace 1174
Eigenvalues for Final Subspace 1174
Eigenvectors 1174
Loadings 1174
Outlier and Leverage Cutoffs 1175
Diagnostic Summary 1175
Outlier and Leverage Diagnostics 1175
Timing Information 1175
ODS Table Names 1176
ODS Graphics 1176
Examples: MVOUTLIER Procedure **1177**
Example 21.1: Breast Cancer Diagnosis 1177
Example 21.2: Ensuring Reproducible Observation Order 1184
Example 21.3: Using Marginal Outlyingness and Leverage 1195
References **1200**

Overview: MVOUTLIER Procedure

The MVOUTLIER procedure performs robust principal component analysis (PCA) to identify outliers and leverage points in any numeric multivariate data set. It is especially useful for correlated high-dimensional data, because the PCA subspace serves as a robust lower-dimensional representation of the data—robust in the sense of minimizing the impact of extreme observations while estimating the covariance structure.

The MVOUTLIER procedure divides observations into four main categories in terms of their relationship to the subspace:

- a homogeneous group that is close to the center of the PCA subspace
- high-leverage observations that are close to the subspace but far from its center
- outlying observations that are far from the subspace but close to its center after being projected onto it
- observations that are both far from the subspace and far from its center after being projected onto it

Often the lower-dimensional representation of the data that you obtain from robust PCA is used for some other multivariate analysis, such as cluster analysis or discriminant analysis. An initial identification of outliers and leverage points from robust PCA can better inform the subsequent analysis.

The method combines and extends two popular PCA approaches: one based on robust covariance estimation, and the other based on projection pursuit. In addition to its purpose of identifying outliers and leverage points, it also serves the traditional purpose of PCA—dimension reduction—by projecting the data onto an optimal subspace that contains most of the information, even if there are outliers.

PROC MVOUTLIER applies naturally to any situation that involves numeric multivariate data in which either anomaly detection or PCA (or both) is desired.

PROC MVOUTLIER Features

The MVOUTLIER procedure provides the following features:

- performs robust dimension reduction that assumes a particular contamination level
- determines the number of principal components needed to explain a target proportion of variance
- displays a scree plot to help choose the final number of principal components
- calculates cutoff values for the two metrics—orthogonal and robust score distances—that identify outliers and leverage points
- optionally incorporates criteria for "marginal" outliers and leverage points, either to identify observations that are below but close to cutoff values or to hide the more moderate outliers and leverage points from being displayed
- displays a diagnostic plot that separates the data into four quadrants on the basis of high and low outlyingness and leverage, or up to nine sectors if marginal outlier and leverage criteria are used
- displays a summary table that contains counts of outliers and leverage points
- displays a table of observations that have the most extreme outlyingness or leverage
- produces an output data table that contains observationwise statistics
- enables reproducibility of observation order
- exploits all the available cores and concurrent threads. For information about how PROC MVOUTLIER uses threads, see the section "Multithreading" on page 102.

Because PROC MVOUTLIER runs on CAS, it also does the following:

- enables you to run on a cluster of machines that distribute the data and the computations
- enables you to run in single-machine mode on CAS

PROC MVOUTLIER Compared with Other SAS Software

A number of other SAS products have features in procedures, CAS actions, and solutions that address outlier detection, leverage, PCA, and robust estimation:

- SAS Visual Statistics
  - The **pca** action set and its associated PCA procedure perform PCA for dimension reduction by using methods such as eigenvalue decomposition, nonlinear iterative partial least squares, Gram-Schmidt orthogonalization, and random projection. They mostly supersede the capabilities of the PRINCOMP and HPPRINCOMP procedures in SAS/STAT.
- SAS Visual Analytics
  - The **outlier** action in the **dataPreprocess** action set performs univariate outlier detection and treatment. Outlier cutoff thresholds can be defined as percentiles, standard or robust Z-scores, or user-supplied values. Outliers can be either removed or replaced with location estimates, lower or upper cutoff thresholds, or user-supplied values.
  - The **rustats** action in the **dataPreprocess** action set computes many different robust univariate statistics, such as quantiles, centralized and standardized moments, and skewness estimates.
- Machine Learning
  - The **robustpca** action in the **robustPca** action set and its associated RPCA procedure perform robust PCA by using principal component pursuit to decompose the input data matrix into the sum of a low-rank matrix $\mathbf{L}$ and a sparse matrix $\mathbf{S}$. PCA is performed on $\mathbf{L}$, and anomaly detection is performed on $\mathbf{S}$. Outliers are defined as observations whose count of variables that exceed a user-supplied standardized sparse value threshold exceeds a user-supplied count threshold.
  - The **mwpca** action in the **robustPca** action set and its associated MWPCA procedure perform moving windows PCA to capture changes in principal components over time. The same type of robust PCA that is performed in the **robustpca** action can be used in each window to exclude outliers from the analysis.
  - The **kPca** action set and its associated KPCA procedure perform nonlinear kernel-based PCA. Rather than finding a linear projection onto a subspace, it embeds the data in a high-dimensional Hilbert space and uses kernel-based linear dimensionality reduction to find a nonlinear manifold that contains the input data.
- Model Studio (Data Mining and Machine Learning Projects)
  - The Anomaly Detection Data Mining Preprocessing node uses the support vector data description (SVDD) formulation to identify and exclude outliers.

  - The Feature Extraction Data Mining Preprocessing node performs dimension reduction based on a choice of several types of PCA, such as traditional, singular value decomposition, and robust PCA that is based on the **robustpca** action in the **robustPca** action set.

- SAS/STAT
  - The CALIS procedure identifies outliers and leverage points in structural equation models. It provides options to downweight outliers by using robust estimation methods based on iteratively weighted least squares in computing parameter estimates, model fit statistics, case-level residuals, residual M-distances, and leverage M-distances. The ALPHAOUT= and ALPHALEV= options in the PROC CALIS statement, like the ALPHAOUTLIER= and ALPHALEVERAGE= options in the PROC MVOUTLIER statement, enable adjustment of the outlier and leverage cutoffs.
  - The ROBUSTREG procedure identifies outliers and leverage points and downweights outliers in linear regression analysis. Its robust estimation methods include Huber M estimation and methods with high breakdown value, such as least trimmed squares, S estimation, and MM estimation. In analysis of leverage points for covariates, it provides several options for adjusting cutoff values. Many computations involve a generalized version of FAST-MCD, a minimum covariance determinant (MCD) estimation method similar to the deterministic MCD methods used in PROC MVOUTLIER.
  - The FACTOR procedure performs PCA by default and also implements several varieties of factor analysis and factor rotation. The NFACTORS= and PROPORTION= options in the PROC FACTOR statement are analogous to the NPRINCOMP= and PROPVARIANCE= options, respectively, in the PROC MVOUTLIER statement. The FACTOR procedure also provides a third way to determine the number of factors: the MINEIGEN= option in the PROC FACTOR statement, which specifies the smallest eigenvalue for which to retain a factor in reducing dimension.

- SAS/QC
  - The MVPMODEL, MVPMONITOR, and MVPDIAGNOSE procedures provide traditional PCA as an alternative to classical T^2 charts, intended for situations in which there is high correlation among process variables.

- SAS IML
  - The MCD and MVE subroutines perform robust estimation of multivariate location and scatter. The MCD subroutine uses a FAST-MCD estimator, and the MVE subroutine uses a minimum volume ellipsoid estimator.

Using CAS Sessions and CAS Engine Librefs

SAS Cloud Analytic Services (CAS) is the analytic server and associated cloud services in SAS Viya. This section describes how to create a CAS session and set up a CAS engine libref that you can use to connect to the CAS session. It assumes that you have a CAS server already available; contact your system administrator if you need help starting and terminating a server. This CAS server is identified by specifying the host on which it runs and the port on which it listens for communications. To simplify your interactions with this CAS server, the host information and port information for the server are stored as SAS option values that are retrieved automatically whenever this CAS server needs to be accessed. You can examine the host and port values for the server at your site by using the following statements:

```
proc options option=(CASHOST CASPORT);
run;
```

In addition to starting a CAS server, your system administrator might also have created a CAS session and a CAS engine libref for your use. You can define your own sessions and CAS engine librefs that connect to the CAS server as shown in the following statements:

```
cas mysess;
libname mylib cas sessref=mysess;
```

The CAS statement creates the CAS session named mysess, and the LIBNAME statement creates the mylib CAS engine libref that you use to connect to this session. It is not necessary to explicitly name the CASHOST and CASPORT of the CAS server in the CAS statement, because these values are retrieved from the corresponding SAS option values.

If you have created the mysess session, you can terminate it by using the TERMINATE option in the CAS statement as follows:

```
cas mysess terminate;
```

For more information about the CAS and LIBNAME statements, see the section "Introduction to Shared Concepts" on page 6 in Chapter 2, "Shared Concepts."

Getting Started: MVOUTLIER Procedure

(View the complete code for this example at https://github.com/sascommunities/doc-supplement-casstat/tree/main/Examples/mvogs1.sas.)

This example demonstrates how to identify outliers and leverage points in a robust principal component analysis. It illustrates two different strategies for choosing the number of principal components to retain for the final subspace:

- Perform only the initial steps of the analysis in order to obtain information about the amount of variance explained by each component, use that information to choose the number of components, and then specify your chosen number in a full analysis.
- Specify a desired proportion of variance to be explained by the principal components, and let the procedure choose the number of components automatically in a full analysis.

The following DATA step creates the data table mylib.cars, which consists of 11 measurements on 111 cars.

NOTE: Input data must be a stored in a table on your CAS server, and a CAS session must be set up. For more information, see "Using CAS Sessions and CAS Engine Librefs" on page 7 and "Loading a SAS Data Set onto a CAS Server" on page 7.

```
data mylib.cars;
   format MakeModel $char20.;
   input MakeModel Length Wheelbase Width Height FrontHd RearHd /
         FrtLegRoom RearSeating FrtShld RearShld Luggage;
   datalines;
AcuraIntegra          -0.2222222     0  -0.5  -2.5   0  -0.5
 0  -0.6666667     -1     -1     1
DodgeColt             -2.2222222  -1.6    -1     0  -1     0
-2          -2   -0.5  -0.75    -1
DodgeOmni             -1.7777778  -0.6  -0.5     1  -1  -1.5
-1          -1  -1.75   -1.5   0.5

   ... more lines ...

NissanAxxess          -0.7777778   0.2  -0.5     8   1   2.5
-2   0.6666667   0.25   1.25  -8.5
NissanVan             -0.1111111  -1.8  -0.5  15.5   0     3
-1         -19      1   2.25    -8
VolkswagenVanagon      0.1111111    -1   2.5  20.5   3     7
-5   6.3333333   3.25  -7.25    -8
 ;
```

These statements assume that your libref is named mylib, but you can substitute any appropriately defined libref.

Many of the measurements are highly correlated with each other. Consequently, PCA is a potentially useful method to reduce the dimension and find the most important sources of variation.

In order to determine the number of principal components to use, you can use the following statements to produce a table and scree plot of the eigenvalues of the initial robust subspace:

```
ods graphics on;
proc mvoutlier data=mylib.cars initonly plots=scree;
   var :;
   id MakeModel;
run;
```

When you specify the INITONLY option in the PROC MVOUTLIER statement, the procedure saves computational time and resources by stopping as soon as these eigenvalues are computed and displayed. The variable name wildcard, which is the colon (:) in the VAR statement, specifies all variables, and the procedure automatically excludes the MakeModel variable because it is nonnumeric. The MakeModel variable that is specified in the ID statement displays MakeModel values instead of automatically assigned observation numbers wherever there is observationwise output. The default CONTAMINATION= option value, 0.25, assumes that 25% of the observations lie outside a more homogeneous group to be used in robust computations.

Model information and observation counts are shown in Figure 21.1 and Figure 21.2.

Figure 21.1 Model Information

The MVOUTLIER Procedure

Model Information	
Data Source	CARS
Assumed Contamination Proportion	0.25
Random Number Seed	1

Figure 21.2 Number of Observations

Number of Observations Read	111
Number of Observations Used	111

Figure 21.3 shows that the data are full rank—having inherent dimensionality equal to the number of variables—and that the initial robust subspace estimation did not perform any dimension reduction.

Figure 21.3 Number of Variables and Initial Subspace Dimensions

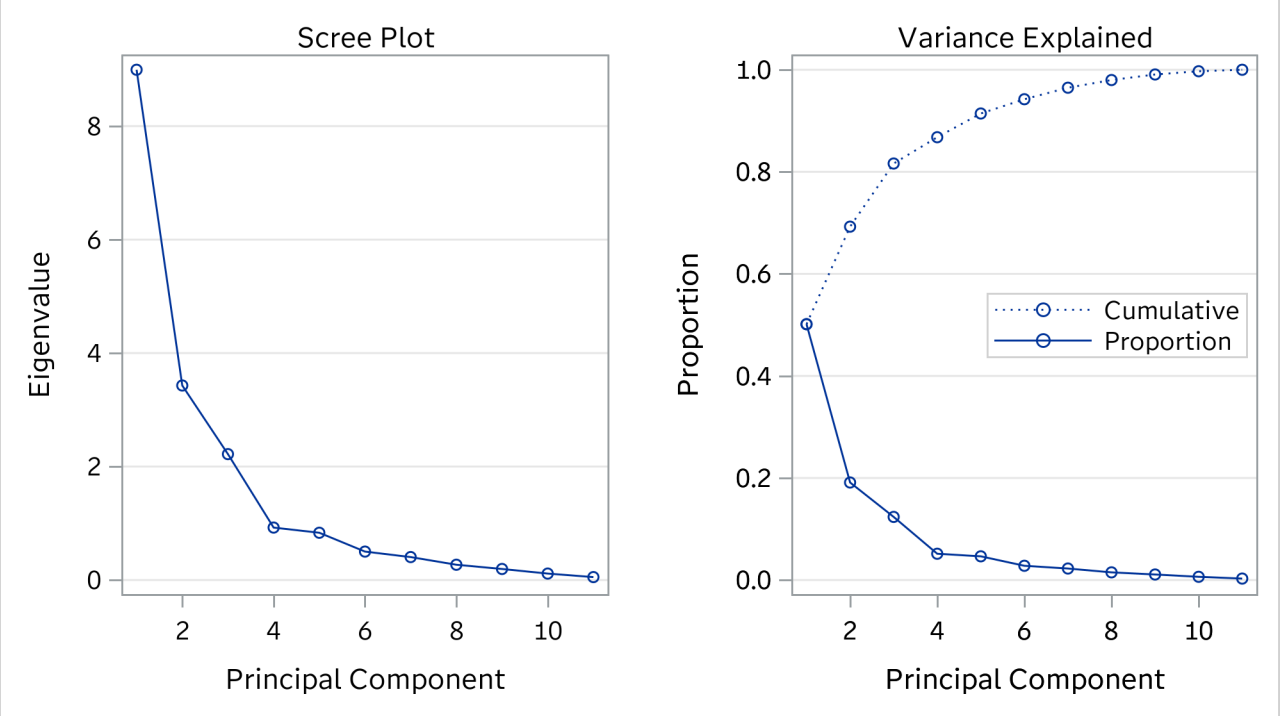
Number of Variables	11
Inherent Data Dimensionality	11

The eigenvalues of the initial robust subspace, which convey the relative proportions of variance explained by the principal components at this stage, are displayed from largest to smallest in tabular form in Figure 21.4 and in graphical form in Figure 21.5.

Figure 21.4 Eigenvalues of Initial Robust Subspace

Eigenvalues for Initial Robust Subspace				
	Eigenvalue	Difference	Proportion	Cumulative
1	8.992968	5.564212	0.5014	0.5014
2	3.428756	1.209667	0.1912	0.6925
3	2.219089	1.294457	0.1237	0.8163
4	0.924631	0.091042	0.0516	0.8678
5	0.833590	0.332053	0.0465	0.9143
6	0.501537	0.096708	0.0280	0.9423
7	0.404829	0.134910	0.0226	0.9648
8	0.269918	0.074896	0.0150	0.9799
9	0.195022	0.080927	0.0109	0.9907
10	0.114096	0.062195	0.0064	0.9971
11	0.051901		0.0029	1.0000

Figure 21.5 Scree Plot of Proportion of Variance Explained



The cumulative proportion of variance explained can inform your choice of how many principal components to retain for the final subspace—a choice that must be made at this point in order to proceed. You can set this number yourself by using the NPRINCOMP= option in the PROC MVOUTLIER statement, or you can use the PROPVARIANCE= option in the PROC MVOUTLIER statement to set it automatically to the number required to explain a target proportion of variance.

The following statements set that target to 0.9 (90%), which should automatically choose five principal components for the final subspace, and perform an analysis of outliers:

```
proc mvoutlier data=mylib.cars propvariance=0.9 plots=diagnostic;
   var :;
   id MakeModel;
   display NVars EigenvaluesFinal Cutoffs DiagSummary Diagnostics;
run;
```

Figure 21.6 confirms that five principal components have been chosen, and Figure 21.7 shows the eigenvalues of this final five-dimensional subspace.

Figure 21.6 Number of Variables and All Subspace Dimensions

The MVOUTLIER Procedure

Number of Variables	11
Inherent Data Dimensionality	11
Dimension of Initial Robust Subspace	11
Number of Principal Components	5

Figure 21.7 Eigenvalues of Final Subspace

Eigenvalues for Final Subspace

	Eigenvalue	Difference	Proportion	Cumulative
1	10.040616	6.199163	0.5522	0.5522
2	3.841453	1.391151	0.2113	0.7634
3	2.450302	1.417185	0.1347	0.8982
4	1.033117	0.214541	0.0568	0.9550
5	0.818576		0.0450	1.0000

The defaults of 0.025 for the ALPHAOUTLIER= and ALPHALEVERAGE= options define outliers as observations in the most extreme 2.5% of the estimated distribution of orthogonal distances, and they define leverage points as observations in the most extreme 2.5% of the estimated distribution of robust score distances. The corresponding cutoff values are shown in Figure 21.8.

Figure 21.8 Outlier and Leverage Cutoffs

Outlier and Leverage Cutoffs

Criterion	Distance Type	Cutoff Alpha	Cutoff Value
Outlier	**Orthogonal**	**0.025**	2.13882
Leverage	**Robust Score**	**0.025**	3.58225

Figure 21.9 shows the counts of observations that are identified as outliers and/or leverage points, and Figure 21.10 shows their orthogonal and robust score distances along with asterisks (*) to indicate outlyingness and leverage.

Figure 21.9 Diagnostic Summary

Diagnostics Summary	
Number of Outlier and Leverage Observations	18
Number of Outlier Only Observations	3
Number of Leverage Only Observations	8

Figure 21.10 Outlier and Leverage Diagnostics

Outlier and Leverage Diagnostics

	Distances			
MakeModel	**Orthogonal**	**Robust Score**	**Outlier**	**Leverage**
Nissan300ZX	24.43432	12.39203	*	*
NissanVan	24.90684	5.77386	*	*
MitsubishiWagon	24.82356	5.68030	*	*
ChevroletCorvette	23.89312	12.13117	*	*
MazdaRX7	23.81631	11.76551	*	*
HondaCivicCRX	22.70962	12.68565	*	*
MazdaMX5Miata	22.62107	11.86225	*	*
VolkswagenVanagon	16.09549	13.86779	*	*
ChevroletAstro	15.13165	11.60724	*	*
FordAerostar	12.10034	11.67271	*	*
MazdaMPV	11.19946	9.79535	*	*
DodgeCaravan	11.19346	9.48298	*	*
DodgeGrandCaravan	8.26960	10.43624	*	*
ChevroletLuminaAPV	7.59583	10.35672	*	*
NissanAxxess	6.29221	9.46475	*	*
Porsche944	2.46439	5.73802	*	*
VolkswagenCorrado	2.43722	4.20155	*	*
FordLTDCrownVictor	2.41450	4.15970	*	*
ChevroletCaprice	2.74830	3.45528	*	
ChryslerImperial	2.31883	1.70116	*	
GEOStorm	2.16179	2.16873	*	
Mercedes-Benz190	1.03003	5.19725		*
MitsubishiSigma	1.02774	4.25643		*
LincolnTownCar	2.04233	4.23069		*
CadillacBrougham	0.96754	4.20716		*
CadillacDeVille	0.55593	3.71779		*
NissanPulsarNX	0.62815	3.67994		*
BuickElectra	1.20799	3.61147		*
DodgeDaytona	1.01100	3.58359		*

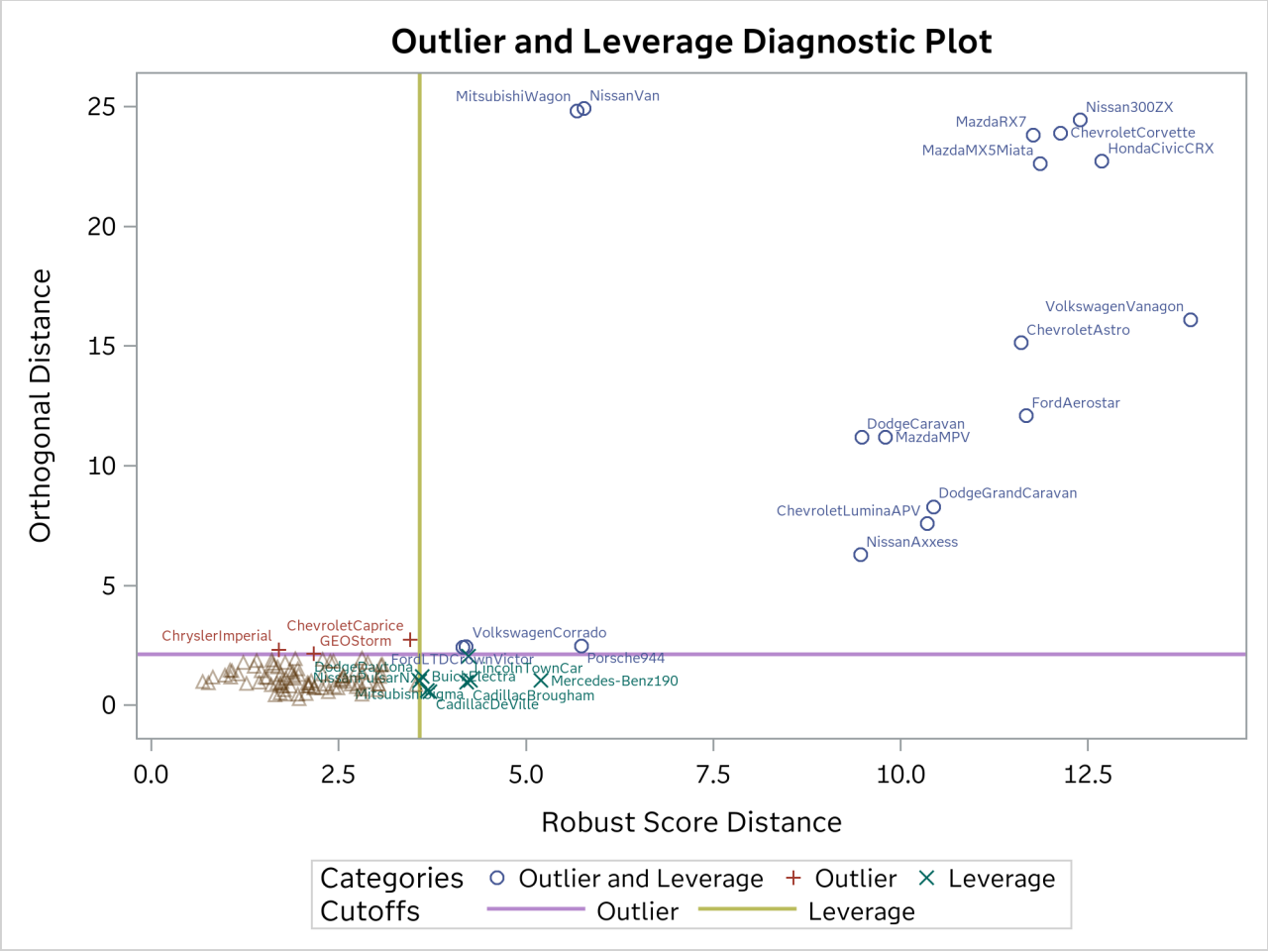
"Leverage" observations lie close to the final subspace but, when projected onto it, appear extreme. "Outliers" lie far away from the final subspace but, when projected onto it, no longer appear extreme. "Outlier and Leverage" observations appear extreme both relative to and within the subspace. The remaining observations exceed neither of the distance cutoff values.

The ALPHAMARGINALOUTLIER= and ALPHAMARGINALLEVERAGE= options in the PROC MVOUTLIER statement control which observations to include in the Diagnostics table. By

default, they are set to the values of the ALPHAOUTLIER= and ALPHALEVERAGE= options in the PROC MVOUTLIER statement, respectively, such that only the observations identified as outliers and/or leverage points are included. You can include *more* observations by *increasing* one or both alpha values, and you can include *fewer* observations by *decreasing* these values. You can also place an upper limit on the total number of observations to include in the Diagnostics table by using the DIAGNOSTICOPTIONS(MAXOBS=) option.

A graphical representation of the outlier and leverage classifications is shown in Figure 21.11.

Figure 21.11 Outlier and Leverage Diagnostic Plot



The ALPHAMARGINALOUTLIER= and ALPHAMARGINALLEVERAGE= suboptions of the PLOTS=DIAGNOSTIC option control which observations are labeled in the diagnostic plot. By default, they are set to the values of the ALPHAMARGINALOUTLIER= and ALPHAMARGINALLEVERAGE= options in the PROC MVOUTLIER statement, respectively. You can also place an upper limit on the total number of observations to include in the diagnostic plot by using the PLOTS=DIAGNOSTIC(MAXOBS=) option.

Syntax: MVOUTLIER Procedure

The following statements are available in the MVOUTLIER procedure:

PROC MVOUTLIER < *options* > **;**
BY *variables* **;**
VAR *variables* **;**
ID *variables* **;**
DISPLAY < *table-list* > < / *options* > **;**
DISPLAYOUT *table-spec-list* < / *options* > **;**
OUTPUT OUT=*libref.data-table* < *keyword* < **=***name* > >. . . < *keyword* < **=***name* > > < *options* > **;**

The PROC MVOUTLIER statement is required.

The rest of this section provides detailed syntax information about each of the preceding statements, beginning with the PROC MVOUTLIER statement. The remaining statements are described in alphabetical order.

PROC MVOUTLIER Statement

PROC MVOUTLIER < *options* > **;**

The PROC MVOUTLIER statement invokes the MVOUTLIER procedure. Optionally, it also identifies the input data set, specifies the analyses to be performed, and controls displayed output. Table 21.1 summarizes the *options* available in the PROC MVOUTLIER statement by function. They are then described fully in alphabetical order.

Table 21.1 PROC MVOUTLIER Statement Options

Option	Description
Option Related to Input Data	
DATA=	Specifies the input data table
Options Related to Number of Principal Components	
INITONLY	Stops the analysis just before the final number of principal components is determined
NPRINCOMP=	Specifies the number of principal components for the final subspace
NPRINCOMPMAX=	Specifies the largest feasible number of principal components, used in conjunction with the PROPVARIANCE= option
PROPVARIANCE=	Specifies the target proportion of variance to be explained by the principal components
Option Related to Robustness	
CONTAMINATION=	Specifies the fraction of observations that are corrupted
Options Related to Outlier and Leverage Assessment	
ALPHALEVERAGE=	Determines the robust score distance cutoff value that is used to identify leverage points
ALPHAOUTLIER=	Determines the orthogonal distance cutoff value that is used to identify outliers
Options Related to Output Tables and Graphs	
ALPHAMARGINALLEV=	Determines the robust score distance cutoff value to use for inclusion in tables or labeling in plots
ALPHAMARGINALOUT=	Determines the orthogonal distance cutoff value to use for inclusion in tables or labeling in plots
DIAGNOSTICOPTIONS	Specifies options for the Diagnostics table
EIGENVEC	Produces the Eigenvectors table
LOADINGS	Produces the Loadings table
PLOTS	Specifies options that control the details of the plots
PREFIX=	Specifies a prefix for naming the principal components
Options Related to Computation	
APPLYROWORDER	Specifies that the procedure use a prespecified data distribution
SEED=	Specifies the seed to use for random number generation

The following list provides details about these *options*.

ALPHALEVERAGE=*p*

ALPHALEV=*p*

specifies the tail probability that determines the robust score distance cutoff value that is used to identify leverage points, where $0 \leq p \leq 1$. By default, ALPHALEVERAGE=0.025.

ALPHAMARGINALLEVERAGE=*p*

ALPHAMARGINALLEV=*p*

ALPHAMARGLEV=*p*

specifies the tail probability that determines the robust score distance cutoff value to use for inclusion in the Diagnostics table or labeling in the diagnostic plot, where $0 \leq p \leq 1$. An observation whose robust score distance exceeds the cutoff determined by the ALPHAMARGINALLEVERAGE= option value is, by default, included in the Diagnostics table and labeled in the diagnostic plot. However, it is still possible for the value of the DIAGNOSTICOPTIONS(MAXOBS=) option to exclude the observation from the Diagnostics table, or for either the PLOTS=DIAGNOSTIC(MAXOBS=) option value or the PLOTS=DIAGNOSTIC(LABEL=NONE) option to remove the observation's label in the diagnostic plot.

The higher the value, the more observations are included or labeled. By default, the option is set to the value of the ALPHALEVERAGE= option, which causes observations to be included or labeled according to the ALPHALEVERAGE= option value.

A value greater than the ALPHALEVERAGE= option value might include or label some observations whose robust score distances do *not* exceed the cutoff determined by the ALPHALEVERAGE= option value. A value less than the ALPHALEVERAGE= option value might exclude or disable labels for some observations whose robust score distances *do* exceed the cutoff determined by the ALPHALEVERAGE= option value.

ALPHAMARGINALOUTLIER=*p*

ALPHAMARGINALOUT=*p*

ALPHAMARGOUT=*p*

specifies the tail probability that determines the orthogonal distance cutoff value to use for inclusion in the Diagnostics table or labeling in the diagnostic plot, where $0 \leq p \leq 1$. An observation whose orthogonal distance exceeds the cutoff determined by the ALPHAMARGINALOUTLIER= option value is, by default, included in the Diagnostics table and labeled in the diagnostic plot. However, it is still possible for the value of the DIAGNOSTICOPTIONS(MAXOBS=) option to exclude the observation from the Diagnostics table, or for either the PLOTS=DIAGNOSTIC(MAXOBS=) option value or the PLOTS=DIAGNOSTIC(LABEL=NONE) option to remove the observation's label in the diagnostic plot.

The higher the value, the more observations are included or labeled. By default, the option is set to the value of the ALPHAOUTLIER= option, which causes observations to be included or labeled according to the ALPHAOUTLIER= option value.

A value greater than the ALPHAOUTLIER= option value might include or label some observations whose orthogonal distances do *not* exceed the cutoff determined by the ALPHAOUTLIER= option value. A value less than the ALPHAOUTLIER= option value might exclude or disable labels for some observations whose orthogonal distances *do* exceed the cutoff determined by the ALPHAOUTLIER= option value.

ALPHAOUTLIER=*p*

ALPHAOUT=*p*

specifies the tail probability that determines the orthogonal distance cutoff value that is used to identify outliers, where $0 \leq p \leq 1$. By default, ALPHAOUTLIER=0.025.

APPLYROWORDER
uses a data distribution and row order as determined by a previous **partition** action call. For more information, see the section "The APPLYROWORDER Option" on page 107 in Chapter 2, "Shared Concepts."

CONTAMINATION=*p*

CONTAM=*p*

CORRUPTED=*p*
specifies the fraction of observations that are corrupted, where $0 \leq p \leq 0.5$. By default, CONTAMINATION=0.25.

DATA=*libref.data-table*
names the input data table for PROC MVOUTLIER to use. The default is the most recently created data table. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to the data, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 1155.

data-table specifies the name of the input data table.

DIAGNOSTICOPTIONS <(*options***)>**

DIAGOPTIONS <(*options***)>**

DIAGOPTS <(*options***)>**
specifies options for the Diagnostics table.

You can specify the following *options*:

MAXOBS=*number*
specifies the maximum number of observations to include in the Diagnostics table. If *number* is less than the number of observations, then observations are prioritized from highest to lowest as follows:

1. Outlier and Leverage, sorted by $(d_o/c_o)^2 + (d_s/c_s)^2$, where d_o and c_o are orthogonal distance and its cutoff, and c_s and o_s are robust score distance and its cutoff
2. Outlier, sorted by orthogonal distance
3. Leverage, sorted by robust score distance
4. Neither, sorted by the same formula as Outlier and Leverage

SHOWOBSID

OBSID
includes the automatically assigned observation ID in the Diagnostics table. This option is automatically enabled if you omit the ID statement.

EIGENVEC
creates the Eigenvectors table, which is produced only if you specify this option.

INITONLY
INITIALONLY
stops the analysis just before the point where the final number of principal components is determined. This saves computation time if you want to obtain only the information relevant to determining how many principal components to retain for the final subspace.

LOADINGS
creates the Loadings table, which is produced only if you specify this option.

NPRINCOMP=*number*
NCOMP=*number*
NPC=*number*
N=*number*
specifies the number of principal components to retain for the final subspace. You must specify either this option or the PROPVARIANCE= option. You cannot specify both.

NPRINCOMPMAX=*number*
NCOMPMAX=*number*
NPCMAX=*number*
NMAX=*number*
specifies the largest feasible number of principal components that you would expect to retain for the final subspace, given the target proportion of variance to explain. This number does not limit the number of components that are actually used; rather, it is used to calculate an observation subset size that must be calculated before the final number of components is determined. It is used in conjunction with the PROPVARIANCE= option; by default, NPRINCOMPMAX=10.

PLOTS <(*global-plot-options***) > <= ***plot-request* **<(***options***) > >**
PLOTS <(*global-plot-options***) > <= (***plot-request* **<(***options***) > < . . . ***plot-request* **<(***options***) > >) >**
controls the plots that are produced through ODS Graphics. When you specify only one *plot-request*, you can omit the parentheses around it.

You can specify the following *global-plot-option*:

UNPACKPANEL
UNPACK
suppresses paneling in the scree plot. By default, multiple plots can appear in an output panel. Specify UNPACKPANEL if you want each plot to appear in a separate panel. You can specify PLOTS(UNPACKPANEL) to unpack the default plots. You can also specify UNPACKPANEL as a suboption of the SCREE option: PLOTS=SCREE(UNPACKPANEL).

You can specify the following *plot-requests*:

ALL
produces all appropriate plots. You can specify other options along with ALL; for example, to request all plots and set the maximum number of observations for the diagnostic plot to 500, specify PLOTS=(ALL DIAGNOSTIC(MAXOBS=500)).

NONE
suppresses the display of all plots.

DIAGNOSTIC < (*diagnostic-options*) >
produces a plot of orthogonal distances versus robust score distances that identifies outliers and leverage points.

You can specify the following *diagnostic-options*:

ALPHAMARGINALLEVERAGE=*p*

ALPHAMARGINALLEV=*p*

ALPHAMARGLEV=*p*
specifies the tail probability that determines the robust score distance cutoff value for labeling an observation in the diagnostic plot, where $0 \leq p \leq 1$. An observation whose robust score distance exceeds the cutoff determined by the ALPHAMARGINALLEVERAGE= option value is labeled in the diagnostic plot unless the PLOTS=DIAGNOSTIC(LABEL=NONE) option is specified or the PLOTS=DIAGNOSTIC(MAXOBS=) option value excludes the observation. The higher the value, the more labels are used.

The default value depends on whether you specify the ALPHAMARGINALLEVERAGE= option in the PROC MVOUTLIER statement. If it is specified, then the default is set to its value. If it is omitted, then the default is set to the value of the ALPHALEVERAGE= option in the PROC MVOUTLIER statement.

ALPHAMARGINALOUTLIER=*p*

ALPHAMARGINALOUT=*p*

ALPHAMARGOUT=*p*
specifies the tail probability that determines the orthogonal distance cutoff value for labeling an observation in the diagnostic plot, where $0 \leq p \leq 1$. An observation whose orthogonal distance exceeds the cutoff determined by the ALPHAMARGINALOUTLIER= option value is labeled in the diagnostic plot unless the PLOTS=DIAGNOSTIC(LABEL=NONE) option is specified or the PLOTS=DIAGNOSTIC(MAXOBS=) option value excludes the observation. The higher the value, the more labels are used.

The default value depends on whether you specify the ALPHAMARGINALOUTLIER= option in the PROC MVOUTLIER statement. If it is specified, then the default is set to its value. If it is omitted, then the default is set to the value of the ALPHAOUTLIER= option in the PROC MVOUTLIER statement.

LABEL=NONE | ID | NUMBER
specifies the type of label to use for outliers and leverage points in the diagnostic plot. You can specify the following values:

NONE shows no labels.

ID uses values of the first ID variable.

NUMBER uses an automatically assigned observation ID.

MAXOBS=*number*
specifies the maximum number of observations to display in the diagnostic plot. If *number* is less than the number of observations, then observations are prioritized from highest to lowest as follows:

1. Outlier and Leverage, sorted by $(d_o/c_o)^2 + (d_s/c_s)^2$, where d_o and c_o are orthogonal distance and its cutoff, and c_s and o_s are robust score distance and its cutoff
2. Outlier, sorted by orthogonal distance
3. Leverage, sorted by robust score distance
4. Neither, sorted by the same formula as Outlier and Leverage

SCREE <(*scree-options***)>**

EIGEN <(*scree-options***)>**

EIGENVALUE <(*scree-options***)>**
produces a scree plot of eigenvalues and proportion of variance explained for the initial robust subspace, at the stage where the final number of principal components must be chosen. By default, both plots appear in the same panel.

You can specify the following *scree-options*:

NPRINCOMPDISPLAY=*number*

NCOMPDISPLAY=*number*

NDISPLAY=*number*
specifies the number of principal components to display in the scree plot. If you omit both NPRINCOMPDISPLAY= and PROPVARDISPLAYMAX=, then all components are displayed. If you specify both, then NPRINCOMPDISPLAY= takes precedence.

PROPVARDISPLAYMAX=*p*

PROPVARIANCEDISPLAYMAX=*p*

PROPVARDISPMAX=*p*
sets the number of principal components to display in the scree plot to the lowest number that explains at least the specified proportion of variance, where $0 \leq p \leq 1$. If you omit both PROPVARDISPLAYMAX= and NPRINCOMPDISPLAY=, then all components are displayed. If you specify both, then NPRINCOMPDISPLAY= takes precedence.

UNPACKPANEL

UNPACK
suppresses paneling for the scree plot so that the eigenvalue plot and proportion of variance plot appear in separate panels.

PREFIX=*name*

specifies a prefix for naming the principal components in the Eigenvectors and Loadings tables. The default *name* is Prin, which produces the component names Prin1, Prin2, Prin3, and so on. The number of characters in the prefix plus the number of digits required to designate the variables must not exceed the current name length that is defined by the VALIDVARNAME= system option.

PROPVARIANCE=*p*

PROPORTIONVARIANCE=*p*

PROPVAR=*p*

specifies the target proportion of variance to be explained by the principal components, where $0 \leq p \leq 1$. You must specify either this option or the NPRINCOMP= option. You cannot specify both. If you specify the PROPVARIANCE= option, then the NPRINCOMPMAX= option also applies.

SEED=*number*

RANDOMSEED=*number*

specifies the seed to use for random number generation.

BY Statement

BY *variables* **;**

You can specify a BY statement in PROC MVOUTLIER to obtain separate analyses of observations in groups that are defined by the values of the BY variables. If you specify more than one BY statement, only the last one specified is used. For more information, see the discussion of BY-group processing in the "Grouping Data" section of *SAS Programmers Guide: Essentials*.

DISPLAY Statement

DISPLAY < *table-list* > < / *options* > **;**

The DISPLAY statement enables you to specify a list of display tables to display or exclude. This statement is similar to the ODS SELECT, ODS EXCLUDE, and ODS TRACE statements. However, the DISPLAY statement can improve performance when a large number of tables could be generated (such as in BY-group processing). The procedure processes the DISPLAY statement on a CAS server and thus sends only a subset of ODS tables to the SAS client. Because ODS statements are processed on a SAS client, first all the generated display tables are sent to the client, and then the client creates a subset.

If you use both DISPLAY and ODS statements together, the DISPLAY statement takes precedence over the ODS statements. Note that the ODS EXCLUDE statement processes tables that are sent to the client after they have been filtered by the DISPLAY statement. In some cases, it might appear that the ODS EXCLUDE statement is taking precedence because it can further filter the tables. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

You can specify the *table-list* as a list of table names, paths, partial pathnames, and regular expressions.

The table names that you can specify are listed in the section "ODS Table Names" on page 1176. A path is a table name that is prefixed with dot-separated grouping information. For example, a

SelectionSummary table that a procedure produces during a selection routine might have the path *Bygroup1.Summary.SelectionSummary*. A partial pathname does not include all groups; for example, *SelectionSummary* and *Summary.SelectionSummary* are partial pathnames for *Bygroup1.Summary.SelectionSummary*.

When you specify a table name or partial pathname, all display tables whose paths end in the specified name are selected for display or exclusion. For example, both *SelectionSummary* and *Summary.SelectionSummary* select *Bygroup1.Summary.SelectionSummary*.

A regular expression is enclosed in forward slashes (/). For example, specifying "/tions/" selects all pathnames that contain the substring "tions"; in particular, the *Bygroup1.Summary.SelectionSummary* table is selected. Specifying "!/tions/" selects all pathnames that do not contain the substring "tions"; in particular, the *Bygroup1.Summary.SelectionSummary* table is not selected.

You can specify the following *options* after a slash (/):

CASESENSITIVE
performs a case-sensitive comparison of table names in the *table-list* to display table names when tables are subsetted for display. To preserve case, you must enclose table names in the *table-list* in quotation marks.

EXCLUDE
displays all display tables except those that you specify in the *table-list*.

EXCLUDEALL
suppresses display of all tables. This option takes precedence over the other options.

TRACE
displays the display table names, labels, and paths.

DISPLAYOUT Statement

DISPLAYOUT *table-spec-list* < / *options* > **;**

The DISPLAYOUT statement enables you to create output tables from your displayed output. This statement is similar to the ODS OUTPUT statement. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

The *table-spec-list* specifies a list of output tables to create. Each entry in the list has either a *key=value* format or a *key* format:

key=value specifies *key* as the ODS table name, path, or partial pathname, and specifies *value* as the output table name.

key specifies *key* as the ODS table name and also as the output table name.

The ODS table names that you can specify are listed in the section "ODS Table Names" on page 1176. You cannot specify the ODS table named OutputCasTables in the *table-spec-list*.

Table names and partial pathnames are discussed under the DISPLAY statement. The DISPLAYOUT statement does not support regular expressions.

You can specify the following *options* after a slash (/):

INCLUDEALL
creates output tables for all display tables. The name of the created output table is the same as the name of the corresponding display table. If you specify this option, the *table-spec-list* specification is ignored.

NOREPLACE
does not replace any existing output table of the same name.

REPEATED
replicates all output tables on all nodes.

ID Statement

ID *variables* **;**

In output that includes information about individual observations, the variables that you list in the ID statement are displayed instead of the automatically assigned observation ID. You can use these variables to identify each observation. If you omit the ID statement, the automatically assigned observation ID is used to identify the observations.

OUTPUT Statement

OUTPUT OUT=*libref.data-table*
< **COPYVARS=***(variables)* >
< *keyword* < **=***name* > > . . . < *keyword* < **=***name* > > **;**

The OUTPUT statement creates a data table that contains observationwise statistics, which are computed after PROC MVOUTLIER fits the model. If you do not specify a *keyword*, then the orthogonal distance and robust score distance are included.

The variables in the input data table are *not* included in the output data table, in order to avoid data duplication for large data tables; however, variables that you specify in the COPYVARS= option *are* included. ID variables are automatically included, and if no ID variables are specified then the automatically assigned observation ID is included.

You must specify the following option:

OUT=*libref.data-table*
names the output data table for PROC MVOUTLIER to use. You must specify this option before any other options. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to where the data table is to be stored, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 1155.

data-table specifies the name of the output data table.

You can also specify the following syntax elements:

COPYVARS=(*variables***)**

COPYVAR=*variable*

transfers one or more *variables* from the input data table to the output data table.

keyword **<=***name***>**

specifies a statistic to include in the output data table and optionally a *name* that is used either as a prefix, for statistics that have more than one variable, or as a name otherwise. If you do not provide a *name*, the MVOUTLIER procedure assigns a default *name* according to the type of statistic that you specify.

You can specify the following *keywords* to add statistics to the OUTPUT data table:

LEVERAGE

specifies a leverage indicator that is set to 1 if the robust score distance exceeds its cutoff and 0 otherwise. The default variable name is Leverage.

NODEID

specifies the node ID of the computer that processes the observation. The default variable name is NodeId.

OBSID

specifies the automatically assigned observation ID. The default variable name is ObsId.

ORTHDIST

specifies the orthogonal distance between the observation and the final subspace. Orthogonal distances are scale-dependent in PROC MVOUTLIER; this means that they are interpreted with respect to the scale of the data. You should consider standardizing or rescaling the data prior to the analysis carefully, on a case-by-base basis. The default variable name is OrthDist.

OUTLIER

specifies an outlier indicator that is set to 1 if the orthogonal distance exceeds its cutoff and 0 otherwise. The default variable name is Outlier.

SCORE

specifies principal component scores for each principal component. The default variable prefix is Score.

SCOREDIST

specifies the robust score distance, which is the distance between the observation after being projected onto the final subspace and the robust location estimate of the subspace mean. The default variable name is ScoreDist.

The output variables that contain the specified statistic are named as follows, according to the *keyword* that you specify:

- The *keyword* SCORE defines an output variable for each principal component, so the variables that correspond to each successive component are named by appending the component number to the prefix. For example, if the model has three principal components, then SCORE=T produces the variables T1, T2, and T3.
- The *keywords* LEVERAGE, NODEID, OBSID, ORTHDIST, OUTLIER, and SCOREDIST each define a single output variable, so the variable name matches the prefix.

VAR Statement

VAR *variables* **;**

The VAR statement lists the numeric variables to be analyzed. If you omit this statement, all numeric variables that are not specified in other statements are analyzed.

Details: MVOUTLIER Procedure

Displayed Output

The following sections describe the output that PROC MVOUTLIER produces. The output is organized into various tables, which are discussed in their order of appearance.

Model Information

The "Model Information" table displays the following basic information about the model and input options:

- Input data table
- Number of analysis variables
- orderBy and groupBy variables, if the input data table has been processed by the **`partition`** action in the **`table`** action set
- Assumed contamination proportion
- Requested number of principal components, if you specify the NPRINCOMP= option in the PROC MVOUTLIER statement
- Target proportion of variance explained and maximum number of principal components, if you specify the PROPVARIANCE= option in the PROC MVOUTLIER statement
- Alpha for outlier cutoff
- Alpha for marginal outlier cutoff, if you specify the ALPHAMARGINALOUTLIER= option in the PROC MVOUTLIER statement
- Alpha for leverage cutoff
- Alpha for marginal leverage cutoff, if you specify the ALPHAMARGINALLEVERAGE= option in the PROC MVOUTLIER statement
- Seed for random numbers

Number of Observations

The "Number of Observations" table displays the number of observations that are read from the input data table and the number of observations that are used in the analysis.

Number of Variables

The "Number of Variables" table displays the following information about the data dimension at different stages in the analysis:

- Number of analysis variables
- Inherent data dimension, which is the rank of the input data matrix
- Dimension of the initial robust subspace, which is the rank of the initial least-outlying subset of observations and also an upper bound for the final number of principal components
- Number of principal components to be extracted for the final subspace

Eigenvalues for Initial Robust Subspace

The "Eigenvalues for Initial Robust Subspace" table displays eigenvalues of the initial robust subspace, along with the difference between successive eigenvalues, the proportion of variance explained by each eigenvalue, and the cumulative proportion of variance explained.

If the number of principal components to retain for the final subspace is not specified explicitly using the NPRINCOMP= option in the PROC MVOUTLIER statement, these eigenvalues are used to determine the final number of principal components.

Eigenvalues for Final Subspace

The "Eigenvalues for Final Subspace" table displays eigenvalues of the final subspace, along with the difference between successive eigenvalues, the proportion of variance explained by each eigenvalue, and the cumulative proportion of variance explained.

Eigenvectors

The "Eigenvectors" table displays the eigenvectors of the final subspace. The EIGENVEC option in the PROC MVOUTLIER statement is required for this table to be displayed.

Eigenvectors describe the orientation of the principal components relative to the original variables. You can project the original variables onto the final subspace by multiplying them by the eigenvector matrix.

Loadings

The "Loadings" table displays the loadings of the final subspace; loadings are the weights of each original variable in each of the principal components.

The loadings describe how much each variable contributes to each principal component. They are the scaled eigenvectors, which result from multiplying each eigenvector by the square root of its corresponding eigenvalue.

Outlier and Leverage Cutoffs

The "Outlier and Leverage Cutoffs" table displays the orthogonal and robust score distance cutoff values that are based in the ALPHAOUTLIER= and ALPHALEVERAGE= options in the PROC MVOUTLIER statement. An observation is identified as an outlier if its orthogonal distance exceeds the outlier cutoff and as a leverage point if its robust score distance exceeds the leverage cutoff.

If you specify the ALPHAMARGINALOUTLIER= option in the PROC MVOUTLIER statement, the orthogonal distance cutoff value that controls inclusion in the Diagnostics table and labeling in the diagnostic plot is displayed. Similarly, if you specify the ALPHAMARGINALLEVERAGE= option in the PROC MVOUTLIER statement, the robust score distance cutoff value that controls inclusion in the Diagnostics table and labeling in the diagnostic plot is displayed.

Diagnostic Summary

The "Diagnostic Summary" table displays counts of observations that are identified as both outlier and leverage, outlier only, or leverage only. If you specify the DIAGNOSTICOPTIONS(MAXOBS=) option in the PROC MVOUTLIER statement and one or more observations are excluded from the Diagnostics table because of their value, then the same three counts for excluded observations are also displayed.

Outlier and Leverage Diagnostics

The "Outlier and Leverage Diagnostics" table displays information about the observations that have the most extreme orthogonal and robust score distances. Observations are included if they exceed either the outlier cutoff determined by the ALPHAMARGINALOUTLIER= option in the PROC MVOUTLIER statement or the leverage cutoff determined by the ALPHAMARGINALLEVERAGE= option in the PROC MVOUTLIER statement, or both.

The table includes an outlier indicator, which shows an asterisk (*) for observations whose orthogonal distances exceed the cutoff determined by the ALPHAOUTLIER= option in the PROC MVOUTLIER statement, and a leverage indicator, which shows an asterisk (*) for observations whose robust score distances exceed the cutoff determined by the ALPHALEVERAGE= option in the PROC MVOUTLIER statement.

The orthogonal and robust score distances are also included, along with either the values of the ID variables (if you specify the ID statement) or automatically assigned observation IDs (if you omit the ID statement, or if you specify the DIAGNOSTICOPTIONS(SHOWOBSID) option in the PROC MVOUTLIER statement).

Timing Information

The "Timing" table displays the elapsed time of each main task of the procedure.

ODS Table Names

Each table that the MVOUTLIER procedure creates has a name associated with it. You must use this name to refer to the table when you use the DISPLAY statement, the DISPLAYOUT statement, or the Output Delivery System (ODS) statements. These names are listed in Table 21.2.

Table 21.2 ODS Tables Produced by PROC MVOUTLIER

Table Name	Description	Required Statement / Option
Cutoffs	Orthogonal and robust score distance cutoffs for outlyingness and leverage, respectively	Default output
Diagnostics	Most extreme outliers and leverage points	Default output
DiagSummary	Counts of observations identified as outliers and/or leverage points	Default output
EigenvaluesFinal	Eigenvalues for the final subspace	Default output
EigenvaluesInit	Eigenvalues for the initial robust subspace	Default output
Eigenvectors	Eigenvectors	EIGENVEC option in the PROC MVOUTLIER statement
Loadings	Loadings	LOADINGS option in the PROC MVOUTLIER statement
ModelInfo	Basic information about the model and input options	Default output
NObs	Number of observations read and used	Default output
NVars	Number of variables, subspace dimensions, and principal components	Default output
OutputCasTables	A special ODS table that has information about all the CAS tables that are created during a CAS action execution	DISPLAYOUT statement or OUTPUT statement
Timing	Absolute and relative times of tasks performed by the procedure	Default output

ODS Graphics

You can refer to every graph produced through ODS Graphics by name. The names of the graphs that PROC MVOUTLIER generates are listed in Table 21.3, along with a description of each graph and the required statements and options.

Table 21.3 Graphics Produced by PROC MVOUTLIER

ODS Graph Name	Plot Description	Required Statement / Option
DiagnosticPlot	Plot of orthogonal distances versus robust score distances that identifies outliers and leverage points	Default output | PLOTS=DIAGNOSTIC
ScreePlot	Series plot of eigenvalues of the principal components that define the initial robust subspace and, if the UNPACKPANEL option is omitted, a series plot of the variance explained by the principal components	Default output | PLOTS=SCREE
VariancePlot	Series plot of the variance explained by the principal components that define the initial robust subspace	PLOTS=SCREE(UNPACKPANEL)

Examples: MVOUTLIER Procedure

The examples in this chapter are available in the GitHub repository located at `https://github.com/sascommunities/doc-supplement-casstat`.

NOTE: Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections "Using CAS Sessions and CAS Engine Librefs" on page 7 and "Loading a SAS Data Set onto a CAS Server" on page 7 in Chapter 2, "Shared Concepts."

Example 21.1: Breast Cancer Diagnosis

(View the complete code for this example (mvoex1.sas) in the example repository.)

The Wisconsin breast cancer diagnosis data set at `https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29` is frequently used to assess classification methods in statistical data analysis and machine learning (Dua and Graff 2019). It consists of 30 different features that are computed from digitized images of 569 specimens from fine-needle aspiration of breast mass biopsies, along with each specimen's diagnosis as either benign or malignant.

The data set that this example uses is modified from the original in the same way as on the Outlier Detection Datasets (ODDS) website `http://odds.cs.stonybrook.edu/wbc/`, keeping the 357 specimens that have a benign diagnosis and sampling 21 specimens at random from the 212 specimens that have a malignant diagnosis. The diagnosis information is removed, and the collection of 378 specimens is assumed to be corrupted data, with a contamination percentage of $21/357 \approx 6\%$.

The goal is to use PROC MVOUTLIER to identify the malignant cases as outliers in a robust principal component analysis.

The following DATA step creates the data table mylib.wbc, which consists of a unique specimen number and 30 features for each breast mass. These statements assume that your libref is named mylib, but you can substitute any appropriately defined libref.

```
data mylib.wbc;
   input Specimen mean_radius mean_texture mean_perimeter mean_area mean_smoothness
     mean_compactness mean_concavity mean_concave_points mean_symmetry
     mean_fractal_dimension /
     stderr_radius stderr_texture stderr_perimeter stderr_area stderr_smoothness
     stderr_compactness stderr_concavity stderr_concave_points stderr_symmetry
     stderr_fractal_dimension /
     max_radius max_texture max_perimeter max_area max_smoothness max_compactness
     max_concavity max_concave_points max_symmetry max_fractal_dimension;
   datalines;
78  11.71 16.67 74.72 423.6 0.1051  0.06095 0.03592   0.026    0.1339 0.05945
 0.4489 2.508  3.258  34.37 0.006578 0.0138   0.02662   0.01307  0.01359  0.003707
  13.33 25.48 86.16 546.7 0.1271  0.1028  0.1046   0.06968  0.1712 0.07343
230 10.57 20.22 70.15 338.3 0.09073 0.166   0.228     0.05941  0.2188 0.0845
 0.1115 1.231  2.363  7.228 0.008499 0.07643  0.1535    0.02919  0.01617  0.0122
  10.85 22.82 76.51 351.9 0.1143  0.3619  0.603    0.1465   0.2597 0.12

   ... more lines ...

340 9.295 13.9  59.96 257.8 0.1371  0.1225  0.03332   0.02421  0.2197 0.07696
 0.3538 1.13   2.388  19.63 0.01546  0.0254   0.02197   0.0158   0.03997  0.003901
  10.57 17.84 67.84 326.6 0.185   0.2097  0.09996  0.07262  0.3681 0.08982
;
```

For easier interpretation of results, the Specimen numbers in the mylib.wbc data table have been assigned intentionally so that specimens 1 through 21 correspond to the breast masses that have a malignant diagnosis. The order of observations has been shuffled randomly and uniformly, and the analysis itself is blind to the actual cancer diagnoses.

The following statements request outlier and leverage diagnostics for a robust principal component analysis that automatically chooses the number of principal components that are needed to explain at least 95% of the variance, assuming that 6% of the data are corrupted:

```
ods graphics on;
proc mvoutlier data=mylib.wbc contamination=0.06 propvariance=0.95
               plots=diagnostic;
   var mean_: stderr_: max_:;
   id Specimen;
run;
```

Model information and observation counts are shown in Output 21.1.1 and Output 21.1.2.

Output 21.1.1 Model Information

The MVOUTLIER Procedure

Model Information	
Data Source	WBC
Assumed Contamination Proportion	0.06
Target Proportion of Variance Explained	0.95
Maximum Number of Principal Components	10
Alpha for Outlier Cutoff	0.025
Alpha for Leverage Cutoff	0.025
Random Number Seed	1

Output 21.1.2 Number of Observations

Number of Observations Read	378
Number of Observations Used	378

The default maximum of 10 principal components merely provides an approximation at the beginning of the algorithm for the largest feasible number of principal components that will be selected later in the algorithm. It is unlikely to affect the final results. The defaults of 0.025 for the ALPHAOUTLIER= and ALPHALEVERAGE= options define outliers as observations in the most extreme 2.5% of the estimated distribution of orthogonal distances, and they define leverage points as observations in the most extreme 2.5% of the estimated distribution of robust score distances.

Output 21.1.3 reveals that the data are highly correlated, having a low rank (6) compared to the number of parameters (30). The initial robust subspace further reduces the dimension by one; this subspace is the basis for selecting the final number of principal components to retain, and its dimension (5) serves as an upper bound.

Output 21.1.3 Number of Variables and Subspace Dimensions

Number of Variables	30
Inherent Data Dimensionality	6
Dimension of Initial Robust Subspace	5
Number of Principal Components	1

Output 21.1.4 shows that the very first eigenvalue of the initial robust subspace explains more than 98% of the variance, surpassing the target of 95% that is specified by the PROPVARIANCE=0.95 option. The choice of one component is stated explicitly in the last row of Output 21.1.3.

Output 21.1.4 Eigenvalues of Initial Robust Subspace

Eigenvalues for Initial Robust Subspace				
	Eigenvalue	**Difference**	**Proportion**	**Cumulative**
1	42247	41793	0.9867	0.9867
2	454.116191	391.459322	0.0106	0.9973
3	62.656869	19.597906	0.0015	0.9988
4	43.058963	34.062911	0.0010	0.9998
5	8.996052		0.0002	1.0000

Output 21.1.5 shows the cutoff values for orthogonal and robust score distances that define outliers and leverage points, respectively, and Output 21.1.6 reveals how many observations have been identified as outliers, leverage points, or both.

Output 21.1.5 Outlier and Leverage Cutoffs

Outlier and Leverage Cutoffs			
Criterion	Distance Type	Cutoff Alpha	Cutoff Value
Outlier	Orthogonal	0.025	37.08827
Leverage	Robust Score	0.025	2.24140

Output 21.1.6 Diagnostic Summary

Diagnostics Summary	
Number of Outlier and Leverage Observations	20
Number of Outlier Only Observations	33
Number of Leverage Only Observations	6

Output 21.1.7 shows the observations that are identified as outliers and/or leverage points, along with their orthogonal and robust score distances.

Output 21.1.7 Outlier and Leverage Diagnostics

Outlier and Leverage Diagnostics

	Distances			
Specimen	Orthogonal	Robust Score	Outlier	Leverage
10	496.09547	8.80586	*	*
16	387.09211	8.13514	*	*
13	312.81436	7.63759	*	*
14	309.36770	7.62184	*	*
4	273.18288	5.77433	*	*
1	243.75137	4.91080	*	*
21	196.21139	7.40746	*	*
17	172.62684	4.54684	*	*
6	163.74464	3.21604	*	*
5	117.57986	5.91316	*	*
18	89.57778	6.68718	*	*
7	128.80917	3.48292	*	*
15	114.38481	4.38325	*	*
9	109.40463	4.45952	*	*
8	124.43058	2.29155	*	*
3	110.41369	2.61779	*	*
11	46.23540	4.36563	*	*
19	49.25423	2.26227	*	*
333	40.30458	2.56671	*	*
122	40.06217	2.28570	*	*
20	112.33501	1.41540	*	
52	107.44375	1.34342	*	
130	69.24444	1.79275	*	
220	68.79486	1.02270	*	
313	67.34181	1.50509	*	
12	65.41784	0.71579	*	
119	63.35109	0.53793	*	
205	58.17500	1.75012	*	
229	57.90716	2.21517	*	
309	54.54635	1.55000	*	
25	54.51965	0.16329	*	
118	54.20949	0.31952	*	
299	53.81194	0.94907	*	
338	53.73247	0.56334	*	
167	53.70707	1.42096	*	
351	52.30720	1.19933	*	
305	51.55609	2.06384	*	
153	51.15072	1.03945	*	
211	49.24074	1.90891	*	
365	48.80931	0.50434	*	
226	47.07374	0.25861	*	
44	46.54005	0.50673	*	
347	46.44564	0.80163	*	
336	46.30055	1.80473	*	
156	44.56949	0.39557	*	
320	40.85402	0.27731	*	

Output 21.1.7 *continued*

Outlier and Leverage Diagnostics				
	Distances			
Specimen	Orthogonal	Robust Score	Outlier	Leverage
192	39.13451	0.09833	*	
335	38.97059	1.48313	*	
115	38.28772	0.24360	*	
272	37.95233	0.57202	*	
374	37.74269	1.30882	*	
249	37.45408	0.07291	*	
174	37.31004	1.55980	*	
321	27.47529	4.19319		*
95	26.32730	3.15858		*
224	9.31692	2.93580		*
254	13.80710	2.58113		*
57	14.31740	2.43122		*
77	8.20405	2.36407		*

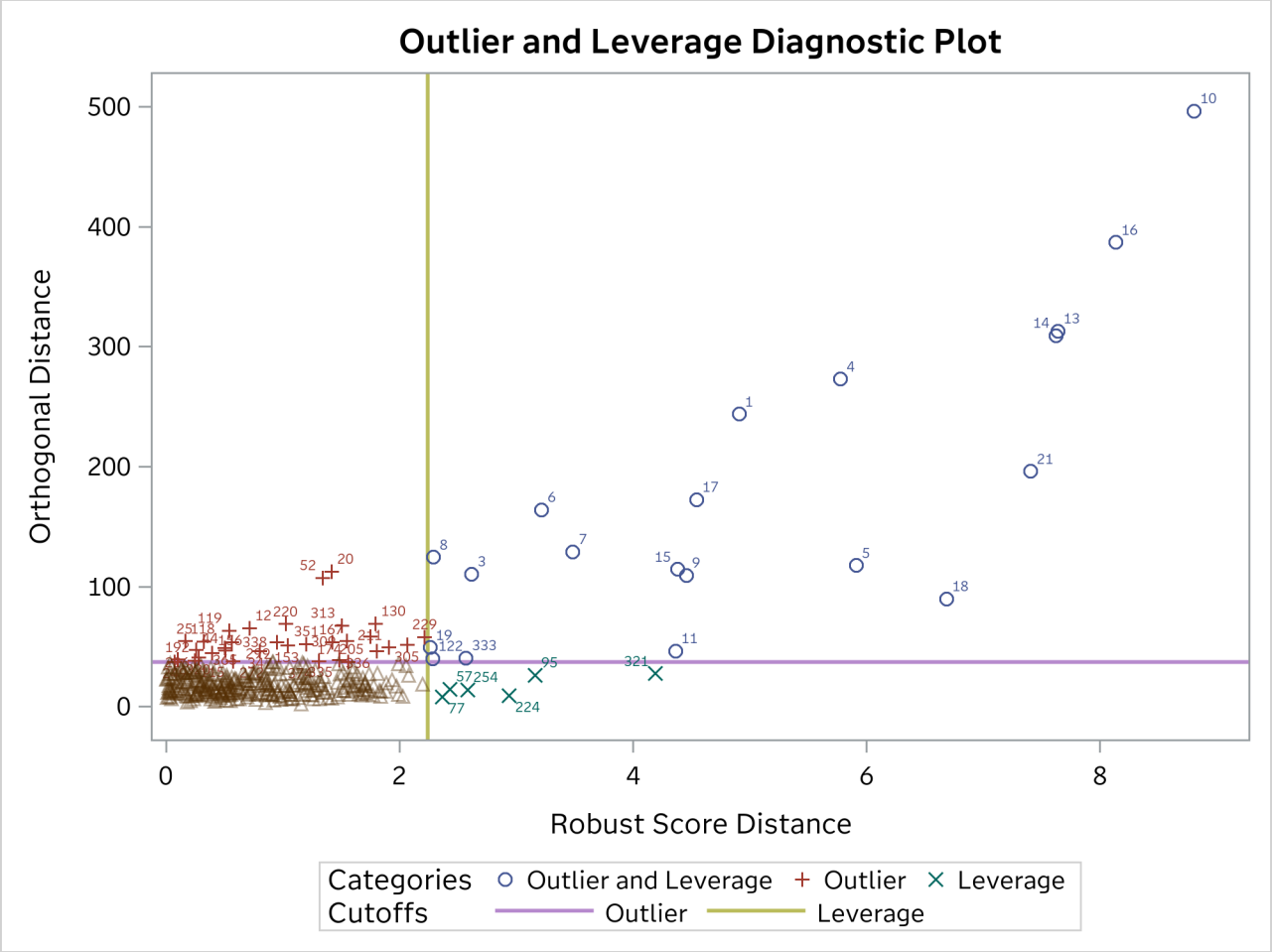
Entries in the Diagnostics table are ordered as follows:

- First, the 20 observations identified as both outliers and leverage points, ordered by a combination of orthogonal and robust score distances—in particular, highest to lowest sum of squares of each of the two distances divided by its corresponding cutoff value.
- Next, the 33 observations identified as outliers but not leverage points, ordered from highest to lowest orthogonal distance.
- Last, the 6 observations identified as leverage points but not outliers, ordered from highest to lowest robust score distance.

The Diagnostics table would finally show any of the remaining 319 observations that meet its inclusion criterion, ordered by the same combination of distances as for the first set of observations. By default, as in this example, only the observations that are identified as outliers or leverage points (or both) are included.

Output 21.1.8 shows the diagnostic plot of orthogonal and robust score distances. It divides the observations into quadrants that are defined by the outlier and leverage point cutoffs.

Output 21.1.8 Outlier and Leverage Diagnostic Plot



The inclusion criteria for both the Diagnostics table and the diagnostic plot are further explored in Example 21.2.

Note that 18 of the 21 specimens that were diagnosed as malignant are among the 20 observations identified as both outliers and leverage points. These observations are orthogonal outliers in the sense that they are far from the one-dimensional subspace that was found by robust PCA to explain more than 98% of the variance of the collection of all 30 features. They are also leverage points in the sense that they would be likely to have a large influence in data analyses performed on the one-dimensional representation of the 30-dimensional data set.

Two of the remaining specimens that were diagnosed as malignant (specimens 20 and 12) are among the most extreme outliers that are not also identified as leverage points.

Specimen 2 is the only specimen diagnosed as malignant whose observation is not identified as either an outlier or a leverage point.

Example 21.2: Ensuring Reproducible Observation Order

(View the complete code for this example (mvoex2.sas) in the example repository.)

This example demonstrates how to use the APPLYROWORDER option in the PROC MVOUTLIER statement and the **partition** action in the **table** action set to ensure that observations in an input data table are processed in the same order in repeated MVOUTLIER procedure calls.

Why might you want to have reproducible row order? In the random PCA step of PROC MVOUTLIER's algorithm, a set of observation pairs is randomly sampled from the input data table. Even if you use the same SEED= option value in the PROC MVOUTLIER statement, a change in the order of the data on the CAS server could cause different sets of pairs to be selected and produce different results.

In this example, you learn how to specify an ordering of your data table and instruct PROC MVOUTLIER to use it.

The data that this example uses are from Example 21.1; they consist of 30 different features that are computed from digitized images of 378 specimens from fine-needle aspiration of breast mass biopsies that are associated with a mix of benign and malignant diagnoses. The following DATA step generates the data. These statements assume that your libref is named mylib, but you can substitute any appropriately defined libref.

```
data mylib.wbc;
   input Specimen mean_radius mean_texture mean_perimeter mean_area mean_smoothness
     mean_compactness mean_concavity mean_concave_points mean_symmetry
     mean_fractal_dimension /
     stderr_radius stderr_texture stderr_perimeter stderr_area stderr_smoothness
     stderr_compactness stderr_concavity stderr_concave_points stderr_symmetry
     stderr_fractal_dimension /
     max_radius max_texture max_perimeter max_area max_smoothness max_compactness
     max_concavity max_concave_points max_symmetry max_fractal_dimension;
   datalines;
78  11.71 16.67 74.72 423.6 0.1051  0.06095 0.03592   0.026    0.1339 0.05945
 0.4489 2.508  3.258  34.37 0.006578 0.0138   0.02662   0.01307  0.01359  0.003707
  13.33 25.48 86.16 546.7 0.1271  0.1028  0.1046   0.06968  0.1712 0.07343
230 10.57 20.22 70.15 338.3 0.09073 0.166   0.228     0.05941  0.2188 0.0845
 0.1115 1.231  2.363  7.228 0.008499 0.07643  0.1535    0.02919  0.01617  0.0122
  10.85 22.82 76.51 351.9 0.1143  0.3619  0.603    0.1465   0.2597 0.12

   ... more lines ...

340 9.295 13.9  59.96 257.8 0.1371  0.1225  0.03332   0.02421  0.2197 0.07696
 0.3538 1.13   2.388  19.63 0.01546  0.0254   0.02197   0.0158   0.03997  0.003901
  10.57 17.84 67.84 326.6 0.185   0.2097  0.09996  0.07262  0.3681 0.08982
;
```

In preparation for a demonstration of the **partition** action in the **table** action set and the APPLYROWORDER option in the PROC MVOUTLIER statement, several options are added to the last run of the MVOUTLIER procedure in Example 21.1 to show more information about observations:

- The SHOWOBSID suboption of the DIAGNOSTICOPTIONS option in the PROC MVOUTLIER statement shows automatically assigned observation numbers in the Diagnostics table.

- The OUTPUT statement creates an output table displaying observationwise statistics that include the automatically assigned observation number.
- The MAXOBS= suboption of the DIAGNOSTIC plot option limits the number of observations shown in the diagnostic plot. This enables you to better identify observations in the crowded homogeneous cluster in the lower left corner of the plot.

The following SAS statements add these options and display the first 25 observations in the output data table:

```
ods graphics on;
proc mvoutlier data=mylib.wbc contamination=0.06 propvariance=0.95
               plots=diagnostic(maxobs=100) diagnosticoptions(showobsid);
   var mean_: stderr_: max_:;
   id Specimen;
   output out=mylib.wbcOut obsid orthdist scoredist outlier leverage;
run;
proc cas;
   fetch / table={name="wbcOut"} to=25;
run;
quit;
```

Output 21.2.1 through Output 21.2.5 show all relevant output tables, which will serve as a basis for comparison later in this example.

Output 21.2.1 Outlier and Leverage Cutoffs

The MVOUTLIER Procedure

Outlier and Leverage Cutoffs

Criterion	Distance Type	Cutoff Alpha	Cutoff Value
Outlier	Orthogonal	0.025	37.08827
Leverage	Robust Score	0.025	2.24140

Output 21.2.2 Diagnostic Summary

Diagnostics Summary	
Number of Outlier and Leverage Observations	20
Number of Outlier Only Observations	33
Number of Leverage Only Observations	6

Output 21.2.3 Outlier and Leverage Diagnostics

Outlier and Leverage Diagnostics

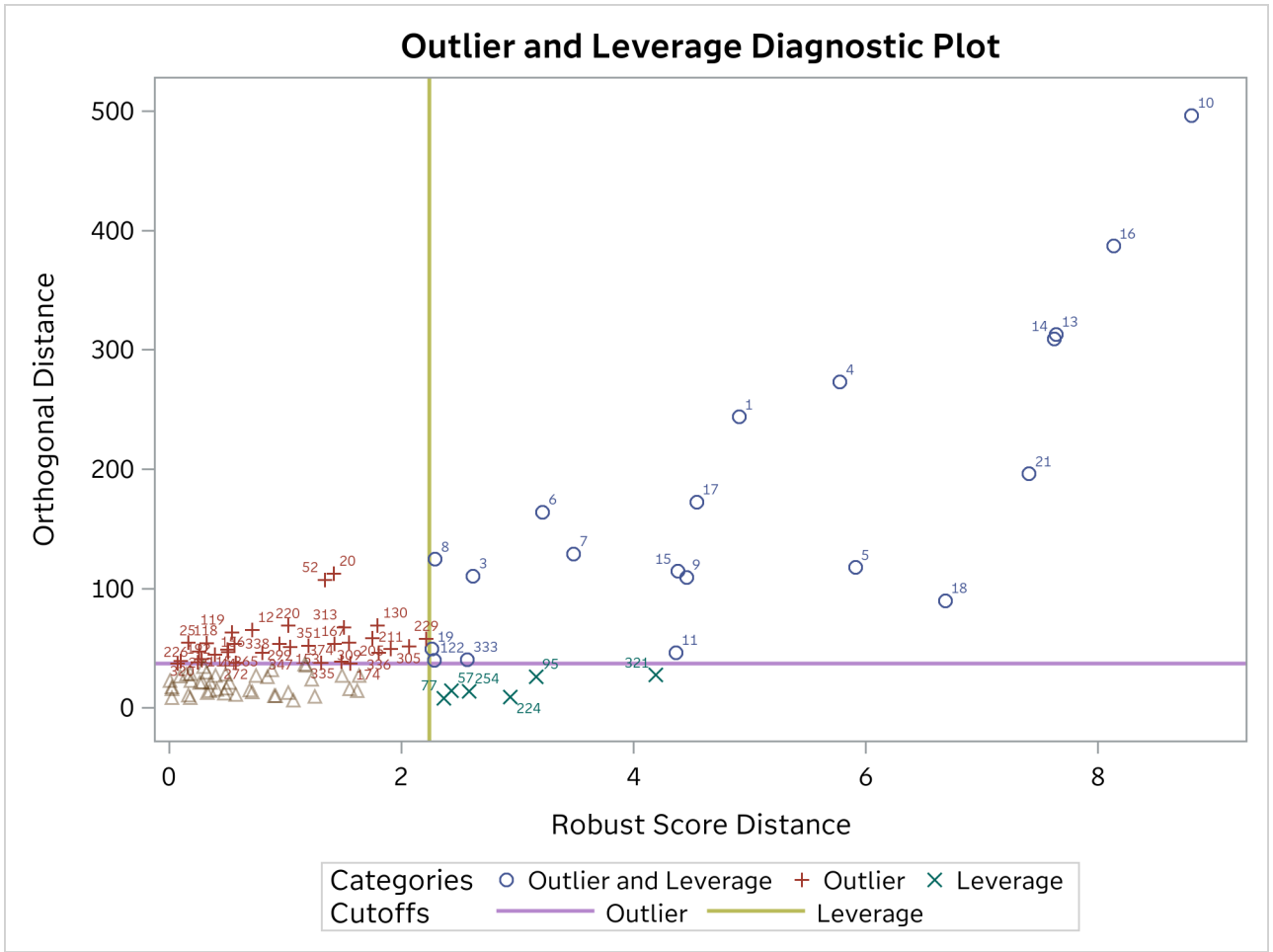
Observation ID	Distances Specimen	Orthogonal	Robust Score	Outlier	Leverage
104	10	496.09547	8.80586	*	*
156	16	387.09211	8.13514	*	*
291	13	312.81436	7.63759	*	*
167	14	309.36770	7.62184	*	*
302	4	273.18288	5.77433	*	*
225	1	243.75137	4.91080	*	*
13	21	196.21139	7.40746	*	*
108	17	172.62684	4.54684	*	*
27	6	163.74464	3.21604	*	*
107	5	117.57986	5.91316	*	*
174	18	89.57778	6.68718	*	*
118	7	128.80917	3.48292	*	*
154	15	114.38481	4.38325	*	*
143	9	109.40463	4.45952	*	*
21	8	124.43058	2.29155	*	*
69	3	110.41369	2.61779	*	*
4	11	46.23540	4.36563	*	*
259	19	49.25423	2.26227	*	*
140	333	40.30458	2.56671	*	*
149	122	40.06217	2.28570	*	*
48	20	112.33501	1.41540	*	
101	52	107.44375	1.34342	*	
98	130	69.24444	1.79275	*	
355	220	68.79486	1.02270	*	
305	313	67.34181	1.50509	*	
45	12	65.41784	0.71579	*	
73	119	63.35109	0.53793	*	
189	205	58.17500	1.75012	*	
188	229	57.90716	2.21517	*	
206	309	54.54635	1.55000	*	
15	25	54.51965	0.16329	*	
171	118	54.20949	0.31952	*	
239	299	53.81194	0.94907	*	
297	338	53.73247	0.56334	*	
367	167	53.70707	1.42096	*	
58	351	52.30720	1.19933	*	
3	305	51.55609	2.06384	*	
139	153	51.15072	1.03945	*	
245	211	49.24074	1.90891	*	
249	365	48.80931	0.50434	*	
268	226	47.07374	0.25861	*	
217	44	46.54005	0.50673	*	
57	347	46.44564	0.80163	*	
200	336	46.30055	1.80473	*	
141	156	44.56949	0.39557	*	
24	320	40.85402	0.27731	*	

Output 21.2.3 *continued*

Outlier and Leverage Diagnostics

Observation ID	Distances Specimen	Orthogonal	Robust Score	Outlier	Leverage
146	192	39.13451	0.09833	*	
298	335	38.97059	1.48313	*	
32	115	38.28772	0.24360	*	
238	272	37.95233	0.57202	*	
287	374	37.74269	1.30882	*	
333	249	37.45408	0.07291	*	
185	174	37.31004	1.55980	*	
136	321	27.47529	4.19319		*
202	95	26.32730	3.15858		*
315	224	9.31692	2.93580		*
135	254	13.80710	2.58113		*
26	57	14.31740	2.43122		*
129	77	8.20405	2.36407		*

Output 21.2.4 Outlier and Leverage Diagnostic Plot



Output 21.2.5 Output Table of Observationwise Statistics (First 25 Rows)

Results from table.fetch

Selected Rows from Table WBCOUT

Index	Specimen	ObsId	OrthDist	ScoreDist	Outlier	Leverage
1	78	1	28.043969557	0.1499496411	0	0
2	230	2	36.900701819	1.1667894512	0	0
3	305	3	51.556090694	2.0638381366	1	0
4	11	4	46.235399592	4.3656250738	1	1
5	40	5	15.831539945	1.5531652102	0	0
6	235	6	34.772940945	0.2957964126	0	0
7	360	7	10.271250968	0.9061491512	0	0
8	48	8	22.839693578	0.0073048859	0	0
9	271	9	26.291831166	0.8429809141	0	0
10	105	10	10.194229545	0.9171930648	0	0
11	169	11	21.74899431	0.2835209857	0	0
12	160	12	23.067429241	0.1881234981	0	0
13	21	13	196.21138686	7.407463584	1	1
14	62	14	16.285906627	0.0246852948	0	0
15	25	15	54.519645524	0.1632855402	1	0
16	291	16	31.85568662	0.8825715999	0	0
17	290	17	27.347137675	0.4008692102	0	0
18	236	18	28.681218893	0.1821974567	0	0
19	193	19	6.563707931	1.0661950515	0	0
20	369	20	14.690649774	1.621307106	0	0
21	8	21	124.43057812	2.2915470212	1	1
22	92	22	14.968526859	0.6901826142	0	0
23	210	23	17.847364252	0.0214776138	0	0
24	320	24	40.85401525	0.2773109703	1	0
25	367	25	13.418372599	0.7162997572	0	0

The MVOUTLIER procedure assigns observation numbers as follows:

1. Nodes in your distributed system and threads within nodes are numbered.
2. Observations are numbered across nodes and threads according to the order in which the observations are processed, counting from lowest- to highest-numbered threads on lowest- to highest-numbered nodes.

What the **`partition`** action and APPLYROWORDER option provide for you is a way to ensure that observations are processed in such a way that their automatically assigned numbers have the same order as the sort order of any variable (or set of variables) of your choosing in your input data table.

The following statements use the **`partition`** action to create a copy of the mylib.wbc data table called mylib.wbcSorted that—if you use the APPLYROWORDER option—is automatically recognized by PROC MVOUTLIER as a data table to be processed in the same sort order as the Specimen variable:

```
data mylib.wbc;
   set mylib.wbc;
   Part = 1;
run;

proc cas;
   action table.partition / table={name="wbc", groupby={name="Part"},
                                   orderby={name="Specimen"}},
                            casout={name="wbcSorted", replace=1};
run;
quit;
```

NOTE: Although this example does not use the groupBy information, use of the APPLYROWORDER option does require at least one groupBy variable to be specified. Instead of setting the groupBy variable Part to a single value, you could use the variable to specify partitions of your data such that all observations that have the same partition key would reside on the same node and same thread in your distributed system.

The following statements add the APPLYROWORDER option to run the previous analysis again but with reproducible row order:

```
proc mvoutlier data=mylib.wbcSorted contamination=0.06 propvariance=0.95
               plots=diagnostic(maxObs=100) diagnosticoptions(showobsid)
               applyroworder;
   var mean_: stderr_: max_:;
   id Specimen;
   output out=mylib.wbcOut obsid orthdist scoredist outlier leverage;
run;
proc cas;
   fetch / table={name="wbcOut"} to=25;
run;
quit;
```

The orderBy and groupBy variables are shown in the "Model Information" table in Output 21.2.6, and Output 21.2.7 through Output 21.2.11 show the other relevant output tables.

Output 21.2.6 Model Information

The MVOUTLIER Procedure

Model Information	
Data Source	WBCSORTED
Input Data Ordered by Variable	Specimen
Input Data Grouped by Variable	Part
Assumed Contamination Proportion	0.06
Target Proportion of Variance Explained	0.95
Maximum Number of Principal Components	10
Alpha for Outlier Cutoff	0.025
Alpha for Leverage Cutoff	0.025
Random Number Seed	1

Output 21.2.7 Outlier and Leverage Cutoffs

Outlier and Leverage Cutoffs			
Criterion	Distance Type	Cutoff Alpha	Cutoff Value
Outlier	Orthogonal	0.025	36.98829
Leverage	Robust Score	0.025	2.24140

Output 21.2.8 Diagnostic Summary

Diagnostics Summary	
Number of Outlier and Leverage Observations	20
Number of Outlier Only Observations	35
Number of Leverage Only Observations	6

Output 21.2.9 Outlier and Leverage Diagnostics

Outlier and Leverage Diagnostics

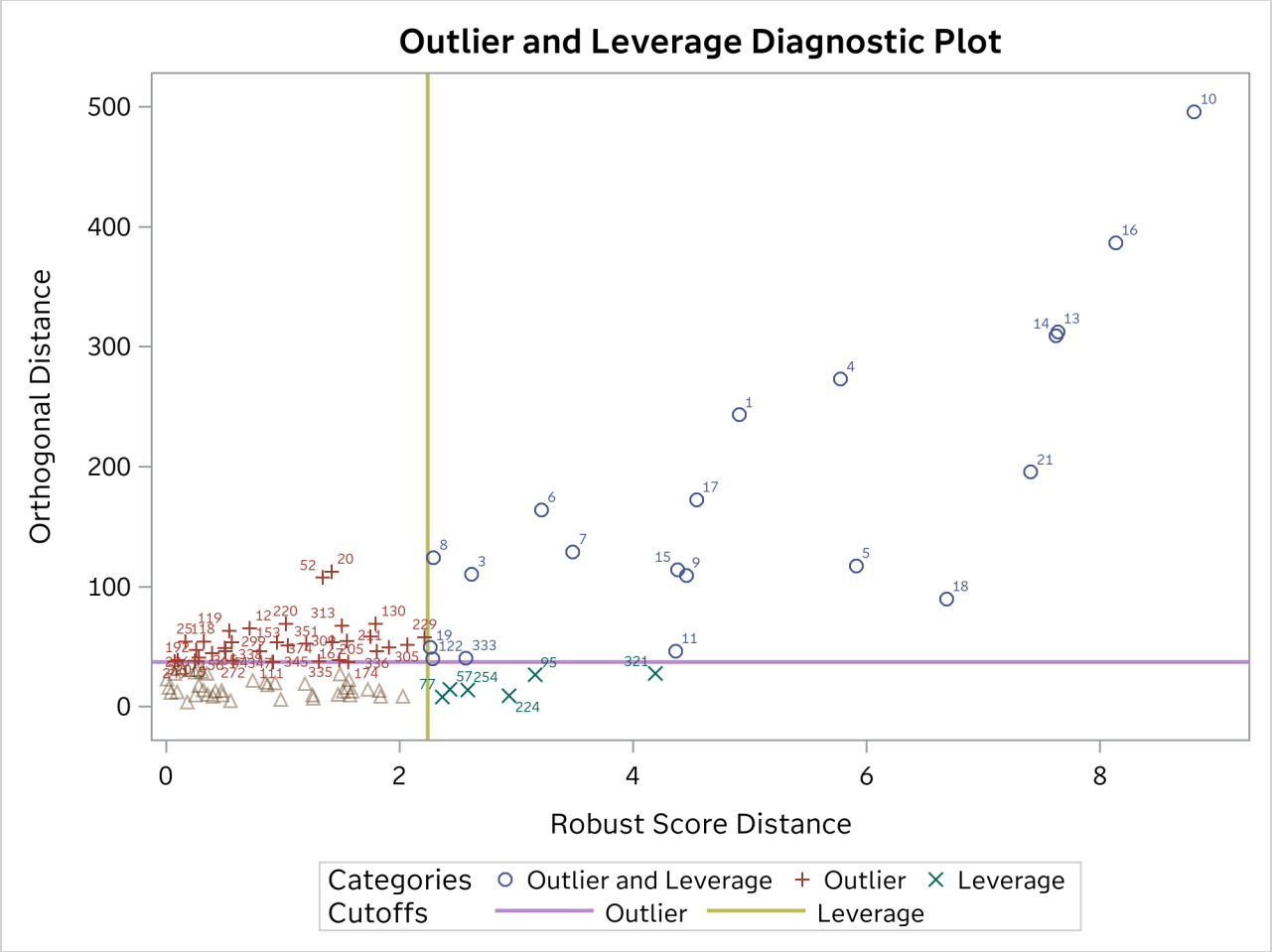
Observation ID	Distances Specimen	Distances Orthogonal	Robust Score	Outlier	Leverage
10	10	495.71971	8.80648	*	*
16	16	386.75882	8.13561	*	*
13	13	312.50597	7.63797	*	*
14	14	309.05602	7.62222	*	*
4	4	272.98508	5.77465	*	*
1	1	243.59471	4.91109	*	*
21	21	195.90905	7.40769	*	*
17	17	172.48740	4.54704	*	*
6	6	163.67504	3.21624	*	*
5	5	117.37891	5.91327	*	*
18	18	89.83703	6.68702	*	*
7	7	128.72379	3.48308	*	*
15	15	114.26608	4.38336	*	*
9	9	109.28815	4.45962	*	*
8	8	124.41534	2.29167	*	*
3	3	110.37660	2.61792	*	*
11	11	46.24236	4.36560	*	*
19	19	49.23837	2.26232	*	*
333	333	40.33861	2.56664	*	*
122	122	40.04624	2.28573	*	*
20	20	112.36545	1.41555	*	
52	52	107.47733	1.34355	*	
130	130	69.25272	1.79282	*	
220	220	68.95912	1.02260	*	
313	313	67.36693	1.50517	*	
12	12	65.48832	0.71588	*	
119	119	63.42980	0.53802	*	
205	205	58.18624	1.75019	*	
229	229	57.92159	2.21507	*	
309	309	54.56927	1.55007	*	
25	25	54.42002	0.16321	*	
118	118	54.30176	0.31959	*	
299	299	53.86747	0.94914	*	
338	338	53.80860	0.56341	*	
167	167	53.70197	1.42093	*	
351	351	52.34745	1.19938	*	
305	305	51.55042	2.06389	*	
153	153	51.10016	1.03938	*	
211	211	49.24323	1.90896	*	
365	365	48.94580	0.50428	*	
226	226	47.16791	0.25867	*	
44	44	46.46008	0.50666	*	
347	347	46.38854	0.80157	*	
336	336	46.30818	1.80478	*	
156	156	44.47286	0.39561	*	
320	320	40.94970	0.27736	*	

Output 21.2.9 *continued*

Outlier and Leverage Diagnostics

Observation ID	Distances Specimen	Distances Orthogonal	Robust Score	Outlier	Leverage
192	192	39.02397	0.09839	*	
335	335	38.94355	1.48306	*	
115	115	38.24257	0.24357	*	
272	272	38.09764	0.57197	*	
374	374	37.70901	1.30876	*	
249	249	37.55109	0.07296	*	
174	174	37.33178	1.55984	*	
111	111	37.13390	0.90918	*	
345	345	37.11105	0.91833	*	
321	321	27.44212	4.19318		*
95	95	26.36944	3.15854		*
224	224	9.31874	2.93578		*
254	254	13.83667	2.58109		*
57	57	14.44157	2.43120		*
77	77	8.19893	2.36405		*

Output 21.2.10 Outlier and Leverage Diagnostic Plot



Output 21.2.11 Output Table of Observationwise Statistics (First 25 Rows)

Results from table.fetch

Selected Rows from Table WBCOUT

Index	Specimen	ObsId	OrthDist	ScoreDist	Outlier	Leverage
1	1	1	243.59471301	4.9110864971	1	1
2	2	2	27.272788024	1.487267159	0	0
3	3	3	110.37659663	2.6179240761	1	1
4	4	4	272.98508178	5.7746482373	1	1
5	5	5	117.3789082	5.9132723292	1	1
6	6	6	163.67503771	3.2162415595	1	1
7	7	7	128.72379023	3.4830757341	1	1
8	8	8	124.41533849	2.2916712967	1	1
9	9	9	109.28814599	4.4596210565	1	1
10	10	10	495.71970769	8.8064819062	1	1
11	11	11	46.242356234	4.3655972869	1	1
12	12	12	65.488316443	0.7158776034	1	0
13	13	13	312.50596593	7.6379685574	1	1
14	14	14	309.05602221	7.6222150025	1	1
15	15	15	114.26607641	4.3833618928	1	1
16	16	16	386.75881587	8.1356115353	1	1
17	17	17	172.48740307	4.5470412373	1	1
18	18	18	89.837031261	6.687023211	1	1
19	19	19	49.238367629	2.2623213549	1	1
20	20	20	112.3654457	1.4155461771	1	0
21	21	21	195.90904516	7.4076922756	1	1
22	22	22	19.967762969	0.9333303724	0	0
23	23	23	9.7725471442	0.4778177215	0	0
24	24	24	12.994915406	1.5165252349	0	0
25	25	25	54.420019209	0.1632066232	1	0

The results in Output 21.2.7 through Output 21.2.11 do in fact differ from those in Output 21.2.1 through Output 21.2.5. Most notable is that two extra observations (Specimens 111 and 345) have been identified as outliers in the second analysis. Their orthogonal distances have changed only slightly, but so has the orthogonal distance cutoff, and that is enough to cross the threshold of the default ALPHAOUTLIER=0.025 option.

You can also see small differences in the orthogonal and robust score differences for specimens 8, 11, 21, and 25 because of their inclusion in both Output 21.2.5 and Output 21.2.11. Different observations have also been selected in the lower left corner of the diagnostic plot (Output 21.2.4 and Output 21.2.10).

Example 21.3: Using Marginal Outlyingness and Leverage

(View the complete code for this example (mvoex3.sas) in the example repository.)

This example demonstrates two ways that you can use the ALPHAMARGINALOUTLIER= and ALPHAMARGINALLEVERAGE= options in the PROC MVOUTLIER statement to control which observations to include in the Diagnostics table or label in the diagnostic plot:

- Specify a marginal alpha value that is *higher* than the corresponding main alpha value to add more observations to the Diagnostics table and label them in the diagnostic plot. The observations that are added or newly labeled are the ones between the main cutoff and the marginal cutoff.
- Specify a marginal alpha value that is *lower* than the corresponding main alpha value to remove some observations from the Diagnostics table and remove their labels in the diagnostic plot. The observations and labels that are removed are the ones beyond the marginal cutoff, which in this case is more restrictive than the main cutoff.

The data that this example uses are from Example 21.1; they consist of 30 different features that are computed from digitized images of 378 specimens from fine-needle aspiration of breast mass biopsies that are associated with a mix of benign and malignant diagnoses. The following DATA step generates the data. These statements assume that your libref is named mylib, but you can substitute any appropriately defined libref.

```
data mylib.wbc;
   input Specimen mean_radius mean_texture mean_perimeter mean_area mean_smoothness
     mean_compactness mean_concavity mean_concave_points mean_symmetry
     mean_fractal_dimension /
     stderr_radius stderr_texture stderr_perimeter stderr_area stderr_smoothness
     stderr_compactness stderr_concavity stderr_concave_points stderr_symmetry
     stderr_fractal_dimension /
     max_radius max_texture max_perimeter max_area max_smoothness max_compactness
     max_concavity max_concave_points max_symmetry max_fractal_dimension;
   datalines;
78  11.71 16.67 74.72 423.6 0.1051  0.06095 0.03592   0.026    0.1339 0.05945
 0.4489 2.508  3.258  34.37 0.006578 0.0138   0.02662   0.01307  0.01359  0.003707
  13.33 25.48 86.16 546.7 0.1271  0.1028  0.1046   0.06968  0.1712 0.07343
230 10.57 20.22 70.15 338.3 0.09073 0.166   0.228     0.05941  0.2188 0.0845
 0.1115 1.231  2.363  7.228 0.008499 0.07643  0.1535    0.02919  0.01617  0.0122
  10.85 22.82 76.51 351.9 0.1143  0.3619  0.603    0.1465   0.2597 0.12

   ... more lines ...

340 9.295 13.9  59.96 257.8 0.1371  0.1225  0.03332   0.02421  0.2197 0.07696
 0.3538 1.13   2.388  19.63 0.01546  0.0254   0.02197   0.0158   0.03997  0.003901
  10.57 17.84 67.84 326.6 0.185   0.2097  0.09996  0.07262  0.3681 0.08982
;
```

The following statements set the ALPHAMARGINALOUTLIER= option to a value higher than the default ALPHAOUTLIER=0.025 option value and the ALPHAMARGINALLEVERAGE= option to a value lower than the default ALPHALEVERAGE=0.025 option value:

```
ods graphics on;
proc mvoutlier data=mylib.wbc contamination=0.06 propvariance=0.95
               plots=diagnostic(maxobs=100) diagnosticoptions(showobsid)
               alphamarginaloutlier=0.05 alphamarginalleverage=0.0125;
   var mean_: stderr_: max_:;
   id Specimen;
run;
```

The main and marginal alpha values are displayed in the "Model Information" table (Output 21.3.1, compared to Output 21.1.1), and their corresponding outlier and leverage cutoff values are displayed in the "Outlier and Leverage Cutoffs" table (Output 21.3.2, compared to Output 21.1.5).

Output 21.3.1 Model Information

The MVOUTLIER Procedure

Model Information	
Data Source	WBC
Assumed Contamination Proportion	0.06
Target Proportion of Variance Explained	0.95
Maximum Number of Principal Components	10
Alpha for Outlier Cutoff	0.025
Alpha for Leverage Cutoff	0.025
Alpha for Marginal Outlier Cutoff	0.05
Alpha for Marginal Leverage Cutoff	0.0125
Random Number Seed	1

Output 21.3.2 Outlier and Leverage Cutoffs

Outlier and Leverage Cutoffs			
Criterion	Distance Type	Cutoff Alpha	Cutoff Value
Outlier	Orthogonal	0.0250	37.08827
Marginal Outlier	Orthogonal	0.0500	33.71677
Leverage	Robust Score	0.0250	2.24140
Marginal Leverage	Robust Score	0.0125	2.49771

An extra row in the "Diagnostic Summary" table (Output 21.3.3, compared to Output 21.1.6) indicates that two leverage observations have been excluded from the "Outlier and Leverage Diagnostics" table (Output 21.3.4, compared to Output 21.1.7). Labels are removed from these same observations, specimens 57 and 77, in the diagnostic plot (Output 21.3.5, compared to Output 21.1.8).

Output 21.3.3 Diagnostic Summary

Diagnostics Summary	
Number of Outlier and Leverage Observations	20
Number of Outlier Only Observations	33
Number of Leverage Only Observations	6
Leverage Only Excluded from Diagnostics Table	2

Output 21.3.4 Outlier and Leverage Diagnostics

Outlier and Leverage Diagnostics

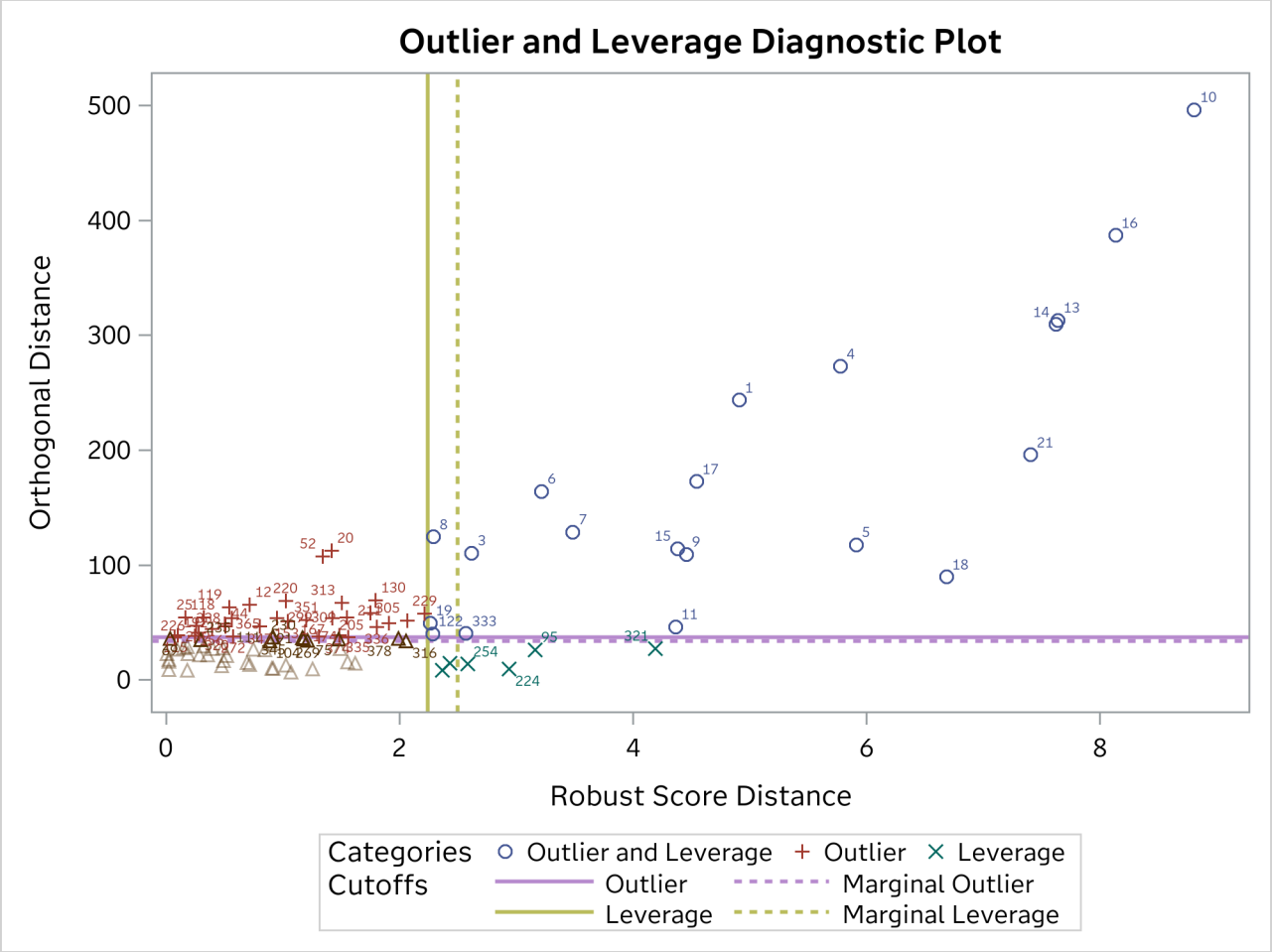
Observation ID	Specimen	Distances Orthogonal	Robust Score	Outlier	Leverage
104	10	496.09547	8.80586	*	*
156	16	387.09211	8.13514	*	*
291	13	312.81436	7.63759	*	*
167	14	309.36770	7.62184	*	*
302	4	273.18288	5.77433	*	*
225	1	243.75137	4.91080	*	*
13	21	196.21139	7.40746	*	*
108	17	172.62684	4.54684	*	*
27	6	163.74464	3.21604	*	*
107	5	117.57986	5.91316	*	*
174	18	89.57778	6.68718	*	*
118	7	128.80917	3.48292	*	*
154	15	114.38481	4.38325	*	*
143	9	109.40463	4.45952	*	*
21	8	124.43058	2.29155	*	*
69	3	110.41369	2.61779	*	*
4	11	46.23540	4.36563	*	*
259	19	49.25423	2.26227	*	*
140	333	40.30458	2.56671	*	*
149	122	40.06217	2.28570	*	*
48	20	112.33501	1.41540	*	
101	52	107.44375	1.34342	*	
98	130	69.24444	1.79275	*	
355	220	68.79486	1.02270	*	
305	313	67.34181	1.50509	*	
45	12	65.41784	0.71579	*	
73	119	63.35109	0.53793	*	
189	205	58.17500	1.75012	*	
188	229	57.90716	2.21517	*	
206	309	54.54635	1.55000	*	
15	25	54.51965	0.16329	*	
171	118	54.20949	0.31952	*	
239	299	53.81194	0.94907	*	
297	338	53.73247	0.56334	*	
367	167	53.70707	1.42096	*	
58	351	52.30720	1.19933	*	
3	305	51.55609	2.06384	*	
139	153	51.15072	1.03945	*	
245	211	49.24074	1.90891	*	
249	365	48.80931	0.50434	*	
268	226	47.07374	0.25861	*	
217	44	46.54005	0.50673	*	
57	347	46.44564	0.80163	*	
200	336	46.30055	1.80473	*	
141	156	44.56949	0.39557	*	
24	320	40.85402	0.27731	*	

Output 21.3.4 *continued*

Outlier and Leverage Diagnostics

Observation ID	Distances Specimen	Distances Orthogonal	Robust Score	Outlier	Leverage
146	192	39.13451	0.09833	*	
298	335	38.97059	1.48313	*	
32	115	38.28772	0.24360	*	
238	272	37.95233	0.57202	*	
287	374	37.74269	1.30882	*	
333	249	37.45408	0.07291	*	
185	174	37.31004	1.55980	*	
136	321	27.47529	4.19319		*
202	95	26.32730	3.15858		*
315	224	9.31692	2.93580		*
135	254	13.80710	2.58113		*
356	378	35.85650	1.99079		
191	316	33.95978	2.05621		
274	75	35.81491	1.47680		
2	230	36.90070	1.16679		
30	91	35.68037	1.17855		
292	345	37.05436	0.91828		
276	111	36.98284	0.90923		
221	269	34.20987	1.21084		
261	104	33.91601	0.88908		
352	69	36.02067	0.03580		
6	235	34.77294	0.29580		

Output 21.3.5 Outlier and Leverage Diagnostic Plot



Several outlier observations have been added at the bottom of the "Outlier and Leverage Diagnostics" table (Output 21.3.4), and these same observations have acquired labels in the diagnostic plot (Output 21.3.5).

The more forgiving marginal outlier cutoff and the more restrictive marginal leverage cutoff in this example are somewhat analogous to the concepts of noninferiority and superiority tests, because they have margins in opposite directions to each other.

References

Dua, D., and Graff, C. (2019). "UCI Machine Learning Repository." School of Information and Computer Sciences, University of California, Irvine. http://archive.ics.uci.edu/ml.

Hubert, M., Rousseeuw, P. J., and Vanden Branden, K. (2005). "ROBPCA: A New Approach to Robust Principal Component Analysis." *Technometrics* 47:64–79.

Chapter 22
NLMOD Procedure

Contents

Overview: NLMOD Procedure **1202**
PROC NLMOD Features 1202
PROC NLMOD Compared with Other SAS Procedures 1203
PROC NLMOD Compared with the HPNLMOD Procedure 1203
PROC NLMOD Compared with the NLIN Procedure 1203
PROC NLMOD Compared with the NLMIXED Procedure 1203
Using CAS Sessions and CAS Engine Librefs 1203
Getting Started: NLMOD Procedure **1204**
Least Squares Model 1204
Binomial Model 1206
Syntax: NLMOD Procedure **1208**
PROC NLMOD Statement 1208
BOUNDS Statement 1211
BY Statement 1211
DISPLAY Statement 1211
DISPLAYOUT Statement 1212
ESTIMATE Statement 1213
ID Statement 1213
MODEL Statement 1214
PARAMETERS Statement 1214
PREDICT Statement 1217
RESTRICT Statement 1218
Programming Statements 1219
Details: NLMOD Procedure **1221**
Least Squares Estimation 1221
Built-In Log-Likelihood Functions 1222
Multithreading 1224
Optimization Algorithms 1224
Displayed Output 1225
ODS Table Names 1227
Examples: NLMOD Procedure **1229**
Example 22.1: Segmented Model 1229
References **1232**

Overview: NLMOD Procedure

The NLMOD procedure fits nonlinear regression models with standard or general distributions in SAS Viya.

PROC NLMOD enables you to specify the model by using SAS programming statements, which give you greater flexibility in modeling the relationship between the response variable and independent (regressor) variables than SAS procedures that use a more structured MODEL statement. The NLMOD procedure uses either nonlinear least squares or maximum likelihood to fit nonlinear regression models.

PROC NLMOD Features

The NLMOD procedure does the following:

- is highly multithreaded during all phases of analytic execution
- computes analytical derivatives of user-provided expressions for more robust parameter estimations
- evaluates user-provided expressions and their confidence limits by using the ESTIMATE and PREDICT statements
- estimates parameters without specifying a particular distribution function by using the least squares method
- estimates parameters by using the maximum likelihood method when either a built-in distribution function is specified or a likelihood function is provided
- exploits all the available cores and concurrent threads, regardless of execution mode. For information about how PROC NLMOD uses threads, see the section "Multithreading" on page 102 in Chapter 2, "Shared Concepts."

Because the NLMOD procedure runs on CAS, it also does the following:

- enables you to run in distributed mode on a cluster of machines that distribute the data and the computations
- enables you to run in single-machine mode on CAS

PROC NLMOD Compared with Other SAS Procedures

The NLMOD procedure provides nonlinear regression modeling functionality that is comparable to that of the HPNLMOD, NLIN, and NLMIXED procedures in SAS/STAT software.

PROC NLMOD Compared with the HPNLMOD Procedure

The functionality of the NLMOD procedure is identical to the HPNLMOD procedure, which is a high-performance procedure. The NLMOD procedure is the next generation of the HPNLMOD procedure, and it was developed specifically for SAS Viya. Both procedures are designed to run on a cluster of machines that distribute the data and the computations. Both procedures perform computations in multiple threads.

Both the NLMOD and HPNLMOD procedures fit nonlinear regression models.

PROC NLMOD Compared with the NLIN Procedure

Like the NLIN procedure in SAS/STAT software, the NLMOD procedure estimates parameters by using least squares minimization for models that are specified by SAS programming statements. However, PROC NLMOD can also perform maximum likelihood estimation when information about the response variable's distribution is provided. PROC NLMOD also has a RESTRICT statement for specifying restrictions on parameter estimates that are more general than those available in PROC NLIN. Because the NLMOD and NLIN procedures use different optimization techniques, the available options that control the estimation process and resulting parameter estimates can differ between these procedures when equivalent models and data are analyzed.

PROC NLMOD Compared with the NLMIXED Procedure

Although it does not support the specification of random effects, PROC NLMOD is similar to the NLMIXED procedure in SAS/STAT. Both procedures perform maximum likelihood estimation by using the same programming syntax and set of distributions to specify the model's mean term. In addition, both PROC NLMOD and PROC NLMIXED use the same optimization techniques and options. However, PROC NLMIXED does not support least squares parameter estimation.

Using CAS Sessions and CAS Engine Librefs

SAS Cloud Analytic Services (CAS) is the analytic server and associated cloud services in SAS Viya. This section describes how to create a CAS session and set up a CAS engine libref that you can use to connect to the CAS session. It assumes that you have a CAS server already available; contact your system administrator if you need help starting and terminating a server. This CAS server is identified by specifying the host on which it runs and the port on which it listens for communications. To simplify your interactions with this CAS server, the host information and port information for the server are stored as SAS option values that are retrieved automatically whenever this CAS server needs to be accessed. You can examine the host and port values for the server at your site by using the following statements:

```
proc options option=(CASHOST CASPORT);
run;
```

In addition to starting a CAS server, your system administrator might also have created a CAS session and a CAS engine libref for your use. You can define your own sessions and CAS engine librefs that connect to the CAS server as shown in the following statements:

```
cas mysess;
libname mylib cas sessref=mysess;
```

The CAS statement creates the CAS session named mysess, and the LIBNAME statement creates the mylib CAS engine libref that you use to connect to this session. It is not necessary to explicitly name the CASHOST and CASPORT of the CAS server in the CAS statement, because these values are retrieved from the corresponding SAS option values.

If you have created the mysess session, you can terminate it by using the TERMINATE option in the CAS statement as follows:

```
cas mysess terminate;
```

For more information about the CAS and LIBNAME statements, see the section "Introduction to Shared Concepts" on page 6 in Chapter 2, "Shared Concepts."

Getting Started: NLMOD Procedure

The most common use of the NLMOD procedure is to estimate the parameters in a model in which the response variable is a nonlinear function of one or more of the parameters.

Least Squares Model

(View the complete code for this example at https://github.com/sascommunities/doc-supplement-casstat/tree/main/Examples/nlmodgs1.sas.)

The Michaelis-Menten model of enzyme kinetics (Ratkowsky 1990, p. 59) relates a substrate's concentration to its catalyzed reaction rate. You can analyze the Michaelis-Menten model by using a least squares estimation because it does not specify how the reaction rate is distributed around its predicted value. The relationship between reaction rate and substrate concentration is

$$f(\mathbf{x}, \boldsymbol{\theta}) = \frac{\theta_1 x_i}{\theta_2 + x_i}, \quad \text{for } i = 1, 2, \ldots, n$$

where x_i represents the concentration for n trials and $f(\mathbf{x}, \boldsymbol{\theta})$ is the reaction rate. The vector $\boldsymbol{\theta}$ contains the rate parameters.

The input data must be a table on your CAS server, and a CAS session must be set up. For more information, see the sections "Using CAS Sessions and CAS Engine Librefs" on page 7 and "Loading a SAS Data Set onto a CAS Server" on page 7 in Chapter 2, "Shared Concepts." The following DATA step creates the Enzyme data table, which consists of experimental measurements of reaction rate and concentration, in your CAS session:

```
data mylib.Enzyme;
   input conc rate @@;
   datalines;
0.26 124.7   0.30 126.9
0.48 135.9   0.50 137.6
0.54 139.6   0.68 141.1
0.82 142.8   1.14 147.6
1.28 149.8   1.38 149.4
1.80 153.9   2.30 152.5
2.44 154.5   2.48 154.7
;
```

These statements assume that your libref is named mylib, but you can substitute any appropriately defined libref.

The following SAS statements estimate the parameters θ_1 and θ_2:

```
proc nlmod data=mylib.Enzyme;
   parms theta1=0 theta2=0;
   model rate ~ residual(theta1*conc / (theta2 + conc));
run;
```

The least squares estimation that PROC NLMOD performs for this enzyme kinetics problem produces the analysis of variance table in Figure 22.1. The table displays the degrees of freedom, sums of squares, and mean squares along with the model F test.

Figure 22.1 Nonlinear Least Squares Analysis of Variance

The NLMOD Procedure

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Approx Pr > F
Model	2	290116	145058	88537.2	<.0001
Error	12	19.6606	1.6384		
Uncorrected Total	14	290135			

An intercept was not specified for this model.

Finally, Figure 22.2 displays the parameter estimates, standard errors, t statistics, and 95% confidence intervals for θ_1 and θ_2.

Figure 22.2 Parameter Estimates and Approximate 95% Confidence Intervals

Parameter Estimates

Parameter	Estimate	Standard Error	DF	t Value	Approx Pr > |t|	Approximate 95% Confidence Limits Lower	Upper
theta1	158.1	0.6737	12	234.67	<.0001	156.6	159.6
theta2	0.0741	0.00313	12	23.69	<.0001	0.0673	0.0809

In the enzyme kinetics model, no information was supplied about the distribution of the reaction rate around the model's mean value. Therefore, the residual model distribution was specified to perform a least squares parameter fit.

Binomial Model

(View the complete code for this example at https://github.com/sascommunities/doc-supplement-casstat/tree/main/Examples/nlmodgs2.sas.)

In the example "Probit Model with Likelihood Function" in the chapter "The NLIN Procedure" in *SAS/STAT User's Guide*, cancer remission is modeled by expressing the maximum likelihood function for a binary distribution as a nonlinear least squares optimization. The following statements show an equivalent formulation of this model that uses PROC NLMOD and specifies the binary distribution explicitly:

```
data mylib.Remiss;
   input remiss cell smear infil li blast temp;
   label remiss = 'complete remission';
   like = 0;
   label like = 'dummy variable for nlin';
   datalines;
 1 0.8  .83 .66 1.9 1.10   .996
 1 0.9  .36 .32 1.4 0.74   .992
 0 0.8  .88 .70 0.8 0.176  .982
 0 1    .87 .87 0.7 1.053  .986
 1 0.9  .75 .68 1.3 0.519  .980
 0 1    .65 .65 0.6 0.519  .982
 1 0.95 .97 .92 1   1.23   .992
 0 0.95 .87 .83 1.9 1.354 1.020
 0 1    .45 .45 0.8 0.322  .999
 0 0.95 .36 .34 0.5 0     1.038
 0 0.85 .39 .33 0.7 0.279  .988
 0 0.7  .76 .53 1.2 0.146  .982
 0 0.8  .46 .37 0.4 0.38  1.006
 0 0.2  .39 .08 0.8 0.114  .990
 0 1    .90 .90 1.1 1.037  .990
 1 1    .84 .84 1.9 2.064 1.020
 0 0.65 .42 .27 0.5 0.114 1.014
 0 1    .75 .75 1   1.322 1.004
 0 0.5  .44 .22 0.6 0.114  .990
 1 1    .63 .63 1.1 1.072  .986
 0 1    .33 .33 0.4 0.176 1.010
 0 0.9  .93 .84 0.6 1.591 1.020
 1 1    .58 .58 1   0.531 1.002
 0 0.95 .32 .30 1.6 0.886  .988
 1 1    .60 .60 1.7 0.964  .990
 1 1    .69 .69 0.9 0.398  .986
 0 1    .73 .73 0.7 0.398  .986
 ;

proc nlmod data=mylib.Remiss corr;
   parms int=-10 a=-2 b=-1 c=6;
   linp = int + a*cell + b*li + c*temp;
   p = probnorm(linp);
   model remiss ~ binary(1-p);
```

```
run;
```

This binary distribution model displays information about the quality of the estimation that is different from the information displayed in the section "Least Squares Model" on page 1204. No analysis of variance table is produced for this model; fit statistics based on the value of the likelihood function are displayed in Figure 22.3.

Figure 22.3 Nonlinear Likelihood Function Statistics

The NLMOD Procedure

Fit Statistics	
-2 Log Likelihood	21.9002
AIC (smaller is better)	29.9002
AICC (smaller is better)	31.7183
BIC (smaller is better)	35.0835

Parameter estimates for the binary distribution model that uses the same quantities as in the section "Least Squares Model" on page 1204 are displayed in Figure 22.4.

Figure 22.4 Parameter Estimates and Approximate 95% Confidence Intervals

Parameter Estimates

Parameter	Estimate	Standard Error	DF	t Value	Approx Pr > |t|	Approximate 95% Confidence Limits Lower	Upper
int	-36.7548	32.3607	27	-1.14	0.2660	-103.2	29.6439
a	-5.6298	4.6376	27	-1.21	0.2353	-15.1454	3.8858
b	-2.2513	0.9790	27	-2.30	0.0294	-4.2599	-0.2426
c	45.1815	34.9095	27	1.29	0.2065	-26.4469	116.8

Syntax: NLMOD Procedure

The following statements are available in the NLMOD procedure:

PROC NLMOD < *options* > ;
BOUNDS *constraint* < , *constraint . . .* > ;
BY *variables* ;
DISPLAY < *table-list* >< / *options* > ;
DISPLAYOUT *table-spec-list*< / *options* > ;
ESTIMATE '*label*' *expression* < *options* > ;
ID *variables* ;
MODEL *dependent-variable* ∼ *distribution* ;
PARAMETERS < *parameter-specification* > < ,*. . .* , *parameter-specification* > < / *options* > ;
PREDICT '*label*' *expression* < *options* > ;
RESTRICT *restriction1* < , *restriction2 . . .* > ;
Programming Statements ;

The PROC NLMOD statement and exactly one MODEL statement are required.

PROC NLMOD Statement

PROC NLMOD < *options* > ;

The PROC NLMOD statement invokes the procedure. Table 22.1 summarizes important options in the PROC NLMOD statement by function. These and other options in the PROC NLMOD statement are then described fully in alphabetical order.

Table 22.1 PROC NLMOD Statement Options

Option	Description
Basic Options	
DATA=	Specifies the input data table
OUT=	Specifies the output data table
Output Options	
CORR	Specifies the correlation matrix
COV	Specifies the covariance matrix
ECORR	Specifies the correlation matrix of additional estimates
ECOV	Specifies the covariance matrix of additional estimates
DF	Specifies the default degrees of freedom
NOPRINT	Suppresses ODS output
NOITPRINT	Suppresses output about iterations within the optimization process
Optimization Options	
ABSCONV=	Tunes an absolute function convergence criterion
ABSFCONV=	Tunes an absolute difference function convergence criterion
ABSGCONV=	Tunes the absolute gradient convergence criterion

Table 22.1 *continued*

Option	Description
FCONV=	Tunes the relative function convergence criterion
GCONV=	Tunes the relative gradient convergence criterion
MAXITER=	Chooses the maximum number of iterations in any optimization
MAXFUNC=	Specifies the maximum number of function evaluations in any optimization
MAXTIME=	Specifies the upper limit (in seconds) of CPU time for any optimization
MINITER=	Specifies the minimum number of iterations in any optimization
TECHNIQUE=	Selects the optimization technique
Tolerance Options	
SINGULAR=	Tunes the general singularity criterion

The optimization options are fully described in the section "Optimization Options" on page 57 in Chapter 2, "Shared Concepts." Other options available in the PROC NLMOD statement are described in the following sections.

You can specify the following *options* in the PROC NLMOD statement:

ALPHA=α
specifies the level of significance α that is used in constructing $100(1-\alpha)\%$ confidence intervals. The value must be strictly between 0 and 1; the default value of $\alpha = 0.05$ results in 95% intervals. This value is used as the default confidence level for limits that are computed in the "Parameter Estimates" table and is used in the LOWER and UPPER options in the PREDICT statement.

CORR
requests the approximate correlation matrix for the parameter estimates.

COV
requests the approximate covariance matrix for the parameter estimates.

DATA=*libref.data-table*
names the input data table for PROC NLMOD to use. The default is the most recently created data table. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to the data, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 1203.

data-table specifies the name of the input data table.

DF=*n*
specifies the default number of degrees of freedom to use in the calculation of *p*-values and confidence limits for additional parameter estimates.

ECORR
requests the approximate correlation matrix for all expressions that are specified in ESTIMATE statements.

ECOV
requests the approximate covariance matrix for all expressions that are specified in ESTIMATE statements.

NOITPRINT
suppresses the display of the "Iteration History" table.

NOPRINT
suppresses the generation of ODS output.

OUT=*libref.data-table*
names the output data table for PROC NLMOD to use. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to where the data table is to be stored, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 1203.

data-table specifies the name of the output data table.

A single OUT= data table is created to contain all predicted values when more than one PREDICT statement is specified. An error message is generated if a PREDICT statement is specified and an OUT= data table is not specified. The variables in the input data table are *not* included in the output data table, in order to avoid data duplication for large data tables; however, variables that you specify in the ID statement are included.

SINGULAR=*number*
tunes the general singularity criterion that is applied in sweeps and inversions. The default is 1E4 times the machine epsilon; this product is approximately 1E-12 on most computers.

BOUNDS Statement

BOUNDS *constraint* < , *constraint . . .* > **;**

where *constraint* represents

< *number operator* > *parameter-list* < *operator number* >

Boundary constraints are specified in a BOUNDS statement. One- or two-sided boundary constraints are allowed. Elements in a list of boundary constraints are separated by commas. For example:

```
bounds 0 <= a1-a9 X <= 1, -1 <= c2-c5;
bounds b1-b10 y >= 0;
```

You can specify more than one BOUNDS statement. If you specify more than one lower (or upper) bound for the same parameter, the maximum (or minimum) of these is taken.

If the maximum l_j of all lower bounds is larger than the minimum of all upper bounds u_j for the same parameter θ_j, the boundary constraint is replaced by $\theta_j := l_j := \min(u_j)$, which is defined by the minimum of all upper bounds specified for θ_j.

BY Statement

BY *variables* **;**

You can specify a BY statement in PROC NLMOD to obtain separate analyses of observations in groups that are defined by the values of the BY variables. If you specify more than one BY statement, only the last one specified is used. For more information, see the discussion of BY-group processing in the "Grouping Data" section of *SAS Programmers Guide: Essentials*.

DISPLAY Statement

DISPLAY < *table-list* > < / *options* > **;**

The DISPLAY statement enables you to specify a list of display tables to display or exclude. This statement is similar to the ODS SELECT, ODS EXCLUDE, and ODS TRACE statements. However, the DISPLAY statement can improve performance when a large number of tables could be generated (such as in BY-group processing). The procedure processes the DISPLAY statement on a CAS server and thus sends only a subset of ODS tables to the SAS client. Because ODS statements are processed on a SAS client, first all the generated display tables are sent to the client, and then the client creates a subset.

If you use both DISPLAY and ODS statements together, the DISPLAY statement takes precedence over the ODS statements. Note that the ODS EXCLUDE statement processes tables that are sent to the client after they have been filtered by the DISPLAY statement. In some cases, it might appear that the ODS EXCLUDE statement is taking precedence because it can further filter the tables. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

You can specify the *table-list* as a list of table names, paths, partial pathnames, and regular expressions.

The table names that you can specify are listed in the section "ODS Table Names" on page 1227. A path is a table name that is prefixed with dot-separated grouping information. For example, a SelectionSummary table that a procedure produces during a selection routine might have the path *Bygroup1.Summary.SelectionSummary*. A partial pathname does not include all groups; for example, *SelectionSummary* and *Summary.SelectionSummary* are partial pathnames for *Bygroup1.Summary.SelectionSummary*.

When you specify a table name or partial pathname, all display tables whose paths end in the specified name are selected for display or exclusion. For example, both *SelectionSummary* and *Summary.SelectionSummary* select *Bygroup1.Summary.SelectionSummary*.

A regular expression is enclosed in forward slashes (/). For example, specifying "/tions/" selects all pathnames that contain the substring "tions"; in particular, the *Bygroup1.Summary.SelectionSummary* table is selected. Specifying "!/tions/" selects all pathnames that do not contain the substring "tions"; in particular, the *Bygroup1.Summary.SelectionSummary* table is not selected.

You can specify the following *options* after a slash (/):

CASESENSITIVE
performs a case-sensitive comparison of table names in the *table-list* to display table names when tables are subsetted for display. To preserve case, you must enclose table names in the *table-list* in quotation marks.

EXCLUDE
displays all display tables except those that you specify in the *table-list*.

EXCLUDEALL
suppresses display of all tables. This option takes precedence over the other options.

TRACE
displays the display table names, labels, and paths.

DISPLAYOUT Statement

DISPLAYOUT *table-spec-list* < / *options* > **;**

The DISPLAYOUT statement enables you to create output tables from your displayed output. This statement is similar to the ODS OUTPUT statement. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

The *table-spec-list* specifies a list of output tables to create. Each entry in the list has either a *key=value* format or a *key* format:

key=value specifies *key* as the ODS table name, path, or partial pathname, and specifies *value* as the output table name.

key specifies *key* as the ODS table name and also as the output table name.

The ODS table names that you can specify are listed in the section "ODS Table Names" on page 1227. You cannot specify the ODS table named OutputCasTables in the *table-spec-list*.

Table names and partial pathnames are discussed under the DISPLAY statement. The DISPLAYOUT statement does not support regular expressions.

You can specify the following *options* after a slash (/):

INCLUDEALL
creates output tables for all display tables. The name of the created output table is the same as the name of the corresponding display table. If you specify this option, the *table-spec-list* specification is ignored.

NOREPLACE
does not replace any existing output table of the same name.

REPEATED
replicates all output tables on all nodes.

ESTIMATE Statement

ESTIMATE '*label*' *expression* < *options* > **;**

The ESTIMATE statement enables you to compute an additional estimate that is a function of the parameter values. You must provide a quoted string to identify the estimate and then provide a valid SAS expression. Multiple ESTIMATE statements are permitted, and results from all ESTIMATE statements are listed in a common table. PROC NLMOD computes approximate standard errors for the estimates by using the delta method (Billingsley 1986). It uses these standard errors to compute corresponding t statistics, p-values, and confidence limits.

The ECOV option in the PROC NLMOD statement produces a table that contains the approximate covariance matrix of all the additional estimates that you specify. The ECORR option produces the corresponding correlation matrix.

You can specify the following *options*:

ALPHA=α
specifies the alpha level to be used to compute confidence limits. The default value corresponds to the ALPHA= option in the PROC NLMOD statement.

DF=d
specifies the degrees of freedom to be used to compute p-values and confidence limits. The default value corresponds to the DF= option in the PROC NLMOD statement.

ID Statement

ID *variables* **;**

The ID statement lists one or more variables from the input data table that are transferred to output data tables that PROC NLMOD creates, provided that the output data table contains one or more records per input observation. For example, when you use an OUT= option in the PROC NLMOD statement to produce prediction statistics, ID variables are added to the output data table.

By default, PROC NLMOD does not include all variables from the input data table in output data tables.

MODEL Statement

MODEL *dependent-variable* ∼ *distribution* **;**

The MODEL statement is the mechanism for either using a distribution specification to specify the distribution of the data or using the RESIDUAL distribution to specify a predicted value. You must specify a single dependent variable from the input data table, a tilde (∼), and then a distribution and its parameters. You can specify the following values for *distribution*:

RESIDUAL(*m*) or **LS**(*m*) specifies no particular distribution. Instead the sum of squares of the differences between *m* and the dependent variable is minimized.

NORMAL(*m*, *v*) specifies a normal (Gaussian) distribution that has mean *m* and variance *v*.

BINARY(*p*) specifies a binary (Bernoulli) distribution that has probability *p*.

BINOMIAL(*n*, *p*) specifies a binomial distribution that has count *n* and probability *p*.

GAMMA(*a*, *b*) specifies a gamma distribution that has shape *a* and scale *b*.

NEGBIN(*n*, *p*) specifies a negative binomial distribution that has count *n* and probability *p*.

POISSON(*m*) specifies a Poisson distribution that has mean *m*.

GENERAL(*ll*) specifies a general log-likelihood function that you construct by using SAS programming statements.

The MODEL statement must follow any SAS programming statements that you specify for computing parameters of the preceding distributions. For information about the built-in log-likelihood functions, see the section "Built-In Log-Likelihood Functions" on page 1222.

PARAMETERS Statement

PARAMETERS < *parameter-specification* > < *,. . . , parameter-specification* > < */ options* > **;**

PARMS < *parameter-specification* > < *,. . . , parameter-specification* > < */ options* > **;**

The purpose of the PARAMETERS statement is to provide starting values for the NLMOD procedure. You can provide values that define a single point in the parameter space or that define a set of points. For more information about the *parameter-specification*, see the section "Assigning Starting Values by Using a Parameter Specification" on page 1215.

You can specify the following *options* after a slash (/):

BEST=*i* > 0

specifies the maximum number of parameter grid points and the corresponding objective function values to display in the "Parameters" table. If you specify this option, the parameter grid points are listed in ascending order of objective function value. By default, all parameter grid points are displayed.

PDATA=*libref.data-table*

DATA=*libref.data-table*

specifies the data table that provides parameter starting values. *libref.data-table* is a two-level name, where *libref* refers to the library, and *data-table* specifies the name of the input data table. For more information about this two-level name, see the DATA= option and the section "Using CAS Sessions and CAS Engine Librefs" on page 1203.

START=*value*
DEFSTART=*value*
specifies a default starting value for all parameters.

Four methods are available for providing starting values to the optimization process. In order of precedence, the methods are as follows:

1. Specify values directly in the PARAMETERS statement.
2. Specify values in the PDATA= data table option.
3. Specify a single value for all parameters by using the START= option.
4. Use the default value 1.0.

The names that you assign to parameters must be valid SAS names and must not be the same as names of variables in the input data table (see the DATA= option in the PROC NLMOD statement). Parameters that are assigned starting values through the PARAMETERS statement can be omitted from the estimation if the expression in the MODEL statement does not depend on them.

Assigning Starting Values by Using a Parameter Specification

A *parameter-specification* has the following general form, where *name* identifies the parameter and *value-list* provides the set of starting values for the parameter:

name **=** *value-list*

Often the *value-list* contains only a single value, but you can use more general and flexible list specifications such as these:

m	a single value
m1, *m2*, . . . , *mn*	several values
m TO *n*	a sequence in which *m* equals the starting value, *n* equals the ending value, and the increment is 1
m TO *n* BY *i*	a sequence in which *m* equals the starting value, *n* equals the ending value, and the increment is *i*
m1, *m2* TO *m3*	mixed values and sequences

When you specify more than one value for a parameter, PROC NLMOD sorts the values in ascending order and removes duplicate values from the parameter list before forming the grid for the parameter search. If you specify several values for each parameter, PROC NLMOD evaluates the model at each point on the grid. The iterations then commence from the point on the grid that yields the smallest objective function value.

For example, the following PARMS statement specifies five parameters and sets their possible starting values as shown in the table:

```
parms  b0 = 0
       b1 = 4 to 8
       b2 = 0 to .6 by .2
       b3 = 1, 10, 100
       b4 = 0, .5, 1 to 4;
```

Possible Starting Values				
B0	B1	B2	B3	B4
0	4	0.0	1	0.0
	5	0.2	10	0.5
	6	0.4	100	1.0
	7	0.6		2.0
	8			3.0
				4.0

The objective function values are calculated for each of the $1 \times 5 \times 4 \times 3 \times 6 = 360$ combinations of possible starting values.

If you specify a starting value by using a *parameter-specification*, any starting values that are provided for this parameter through the PDATA= data table are ignored. The *parameter-specification* overrides the information in the PDATA= data table.

Assigning Starting Values from a Data Table That Resides in a Library

The PDATA= option in the PARAMETERS statement enables you to assign starting values for parameters by using a data table. You must specify the library where your data table resides and the data table name. The data table must contain at least two variables: a character variable named Parameter (or Parm) that identifies the parameter, and a numeric variable named Estimate (or Est) that contains the starting values. For example, the PDATA= option enables you to use the contents of the "ParameterEstimates" table from one PROC NLMOD run to supply starting values for a subsequent run, as follows:

```
proc nlmod data=mylib.D(obs=30);
   parameters alpha=100 beta=3 gamma=4;
   Switch = 1/(1+gamma*exp(beta*log(dose)));
   model y ~ residual(alpha*Switch);
   displayout ParameterEstimates=pest;
run;

proc nlmod data=mylib.D;
   parameters / pdata=mylib.pest;
   Switch = 1/(1+gamma*exp(beta*log(dose)));
   model y ~ residual(alpha*Switch);
run;
```

You can specify multiple values for a parameter in the PDATA= data table, and the parameters can appear in any order. The starting values are collected by parameter and arranged in ascending order, and duplicate values are removed. The parameter names in the PDATA= data table are not case-sensitive. For example, the following DATA step defines starting values for three parameters and a starting grid that has $1 \times 3 \times 1 = 3$ points:

```
data mylib.Test;
   input Parameter $ Estimate;
   datalines;
 alpha  100
 BETA     4
 beta     4.1
```

```
 beta      4.2
 beta      4.1
 gamma    30
;
```

PREDICT Statement

PREDICT '*label*' *expression* < *options* > **;**

PREDICT '*label*' **MEAN** < *options* > **;**

The PREDICT statement enables you to construct predictions of an expression across all the observations in the input data table. Multiple PREDICT statements are permitted. Results for all PREDICT statements are displayed in the output data table that you specify in the OUT= option in the PROC NLMOD statement. The variables in the input data table are *not* included in the output data table, in order to avoid data duplication for large data tables; however, variables that you specify in the ID statement are included.

You must specify the following arguments:

'*label*'
identifies the predicted expression.

expression | **MEAN**
provides the predicted value.You can specify the predicted value either by using a SAS programming expression that contains the input data table variables and parameters or by using the keyword MEAN. If you specify the keyword MEAN, the predicted mean value for the distribution specified in the MODEL statement is used. Predicted values are computed using the final parameter estimates. Standard errors of prediction are computed using the delta method (Billingsley 1986; Cox 1998).

You can also specify the following *options*:

ALPHA=α
specifies the alpha level to be used to compute confidence limits. The default value corresponds to the ALPHA= option in the PROC NLMOD statement.

DF=*d*
specifies the degrees of freedom to be used to compute confidence limits. The default value corresponds to the DF= option in the PROC NLMOD statement.

LOWER=*name*
specifies a variable that contains the lower confidence limit of the predicted value.

PRED=*name*
specifies a variable that contains the predicted value.

PROBT=*name*
specifies a variable that contains the *p*-value of the predicted value.

STDERR=*name*
specifies a variable that contains the standard error of the predicted value.

TVALUE=*name*
specifies a variable that contains the *t* statistic for the predicted value.

UPPER=*name*
specifies a variable that contains the upper confidence limit of the predicted value.

RESTRICT Statement

RESTRICT *restriction1* < , *restriction2 . . .* > **;**

The RESTRICT statement imposes linear restrictions on the model's parameters estimates. You can specify any number of RESTRICT statements.

Each *restriction* is written as an expression, optionally followed by an equality operator (=) or an inequality operator (<, >, <=, >=), followed by a second expression:

expression **<** *operator expression* **>**

The *operator* can be =, <, >, <= , or >=. The operator and second expression are optional. When they are omitted, the *operator* defaults to = and the second *expression* defaults to the value 0.

Restriction expressions can be composed of parameter names, arithmetic operators, functions, and constants. You cannot use comparison operators (such as = or <) or logical operators (such as &) in RESTRICT statement expressions. Parameters that are named in restriction expressions must be among the parameters that are estimated by the model. Restriction expressions cannot refer to other variables that are defined in the program or in the DATA= data table. The restriction expressions must be linear functions of the parameters.

The following example illustrates how to use the RESTRICT statement to impose a linear constraint on parameters:

```
proc nlmod;
   parms alpha beta;
   f = (x/alpha + beta)**2
   model y ~ residual(f);
   restrict beta < 2*(alpha + constant('pi'));
run;
```

The preceding RESTRICT statement represents the following model constraint:

$$\beta < 2(\alpha + \pi)$$

Programming Statements

Programming statements define the arguments of the MODEL, ESTIMATE, and PREDICT statements in PROC NLMOD. Most of the programming statements that you can use in the SAS DATA step can also be used in the NLMOD procedure. For a description of SAS programming statements, see *SAS Language Reference: Concepts*. You can use the following programming statements:

ABORT;
CALL *name* **[(** *expression* **[,** *expression* **. . .])];**
DELETE;
DO[*variable* **=** *expression*
[TO *expression***] [BY** *expression***]**
[, *expression* **[TO** *expression***] [BY** *expression* **] . . .]**
]
[WHILE *expression* **] [UNTIL** *expression* **] ;**
END;
GOTO *statement_label*;
IF *expression*;
IF *expression* **THEN** *program_statement*;
ELSE *program_statement*;
variable **=** *expression*;
variable **+** *expression*;
LINK *statement_label*;
PUT [*variable***] [=] [. . .];**
RETURN;
SELECT[(*expression***)];**
STOP;
SUBSTR(*variable***,** *index***,** *length* **)=** *expression*;
WHEN (*expression***)** *program_statement*;
OTHERWISE *program_statement*;

For the most part, these SAS programming statements work the same as they do in the SAS DATA step, as documented in *SAS Language Reference: Concepts*. However, they differ as follows:

- The ABORT statement does not allow any arguments.
- The DO statement does not allow a character index variable. Thus, PROC NLMOD supports the first of the following statements, but not the second:

```
do i = 1,2,3;

do i = 'A','B','C';
```

- In contrast to other procedures that share PROC NLMOD's programming syntax, PROC NLMOD does not support the LAG function. Because observations are not processed sequentially when high-performance analytical procedures perform the parameter optimization, information for computing lagged values is not available.

- The PUT statement, used mostly for program debugging in PROC NLMOD, supports only some of the features of the DATA step PUT statement, and it has some new features that the DATA step PUT statement does not have:
  - The PROC NLMOD PUT statement does not support line pointers, factored lists, iteration factors, overprinting, _INFILE_, the colon (:) format modifier, or "$".
  - The PROC NLMOD PUT statement supports expressions, but the expression must be enclosed in parentheses. For example, the following statement displays the square root of x:

```
put  (sqrt(x));
```

  - The PROC NLMOD PUT statement supports the item _PDV_, which displays a formatted listing of all variables in the program. For example, the following statement displays a much more readable listing of the variables than the _ALL_ print item:

```
put _pdv_;
```

- The WHEN and OTHERWISE statements enable you to specify more than one programming statement. That is, DO/END groups are not necessary for multiple WHEN statements. For example, the following syntax is valid:

```
select;
   when (exp1) stmt1;
               stmt2;
   when (exp2) stmt3;
               stmt4;
end;
```

When you write your programming statements, avoid defining variables that begin with an underscore (_), because they might conflict with internal variables that PROC NLMOD creates. The MODEL statement must follow any SAS programming statements that define or modify terms that are used to specify the model.

Special Variable _WEIGHT_

Defining the special variable _WEIGHT_ in your PROC NLMOD programming statements enables you either to request weighted nonlinear least squares estimation of parameters or to request that each observation in the input data set be weighted by a value.

You can define the _WEIGHT_ variable as in the following statement:

```
_weight_ = expression;
```

When you specify this statement, the expression on the right-hand side of the assignment statement is evaluated for each observation in the data set. You can simply provide a variable name instead of an expression. PROC NLMOD uses the value of the _WEIGHT_ variable in two ways, depending on which type of model you request:

- When you do not specify a particular distribution, PROC NLMOD performs the weighted nonlinear least squares method by using the value of the _WEIGHT_ expression for each observation. These

values multiplied by $1/\sigma^2$ are taken as inverse elements of the diagonal variance-covariance matrix of the dependent variable. The larger the _WEIGHT_ value, the more importance the observation is given.

- When you specify either a built-in distribution function or a general likelihood function, PROC NLMOD performs the maximum likelihood method by weighting each observation in the input data set by the value of the _WEIGHT_ expression.

Note that the _WEIGHT_ expression can be a function of the estimated parameters. For estimation purposes, the derivative of the _WEIGHT_ expression with respect to the parameters is not included in the gradient and Hessian of the loss function. This is usually the desired approach for iteratively reweighted least squares estimation. When the _WEIGHT_ expression is a function of the parameters, the gradient and Hessian that are used can lead to poor convergence or nonconvergence of the requested estimation. To have the derivative of the _WEIGHT_ expression with respect to the parameters included in the gradient and Hessian of the loss function, do not use the _WEIGHT_ programming statement. Instead, redefine the model as

$$(y - f(x,\beta)) \times \sqrt{\mathrm{wgt}(\beta)}$$

where y is the original dependent variable, $f(x,\beta)$ is the nonlinear model, and $\mathrm{wgt}(\beta)$ is the weight that is a function of the parameters.

Details: NLMOD Procedure

Least Squares Estimation

Models that are estimated by PROC NLMOD can be represented by using the equations

$$\begin{aligned} \mathbf{Y} &= \mathbf{f}(\boldsymbol{\beta}; \mathbf{z}_1, \ldots, \mathbf{z}_k) + \boldsymbol{\epsilon} \\ \mathrm{E}[\boldsymbol{\epsilon}] &= \mathbf{0} \\ \mathrm{Var}[\boldsymbol{\epsilon}] &= \sigma^2 \mathbf{I} \end{aligned}$$

where

$\mathbf{Y}$	is the $(n \times 1)$ vector of observed responses.
$\mathbf{f}$	is the nonlinear prediction function of parameters and regressor variables.
$\boldsymbol{\beta}$	is the vector of model parameters to be estimated.
$\mathbf{z}_1, \ldots, \mathbf{z}_k$	are the $(n \times 1)$ vectors of each of the k regressor variables.
$\boldsymbol{\epsilon}$	is the $(n \times 1)$ vector of residuals.
σ^2	is the variance of the residuals.

In these models, the distribution of the residuals is not specified and the model parameters are estimated using the least squares method. For the standard errors and confidence limits in the "ParameterEstimates" table to apply, the errors are assumed to be homoscedastic and uncorrelated and to have zero mean.

Built-In Log-Likelihood Functions

For models in which the distribution of model errors is specified, the NLMOD procedure estimates parameters by maximizing the value of a log-likelihood function for the specified distribution. The log-likelihood functions that PROC NLMOD uses for the supported error distributions are as follows:

$Y \sim \text{normal}(m, v)$

$$
\begin{aligned}
l(m, v; y) &= -\frac{1}{2}\left(\log\{2\pi\} + \frac{(y-m)^2}{v} + \log\{v\}\right) \\
\mathrm{E}[Y] &= m \\
\mathrm{Var}[Y] &= v \\
v &> 0
\end{aligned}
$$

$Y \sim \text{binary}(p)$

$$
\begin{aligned}
l_1(p; y) &= \begin{cases} y \ \log\{p\} & y > 0 \\ 0 & \text{otherwise} \end{cases} \\
l_2(p; y) &= \begin{cases} (1-y) \ \log\{1-p\} & y < 1 \\ 0 & \text{otherwise} \end{cases} \\
l(p; y) &= l_1(p; y) + l_2(p; y) \\
\mathrm{E}[Y] &= p \\
\mathrm{Var}[Y] &= p\,(1-p) \\
0 &< p < 1
\end{aligned}
$$

$Y \sim \text{binomial}(n, p)$

$$
\begin{aligned}
l_c &= \log\{\Gamma(n+1)\} - \log\{\Gamma(y+1)\} - \log\{\Gamma(n-y+1)\} \\
l_1(n, p; y) &= \begin{cases} y \ \log\{p\} & y > 0 \\ 0 & \text{otherwise} \end{cases} \\
l_2(n, p; y) &= \begin{cases} (n-y) \ \log\{1-p\} & n-y > 0 \\ 0 & \text{otherwise} \end{cases} \\
l(n, p; y) &= l_c + l_1(n, p; y) + l_2(n, p; y) \\
\mathrm{E}[Y] &= n\,p \\
\mathrm{Var}[Y] &= n\,p\,(1-p) \\
0 &< p < 1
\end{aligned}
$$

$Y \sim \text{gamma}(a, b)$

$$
\begin{aligned}
l(a, b; y) &= -a\log\{b\} - \log\{\Gamma(a)\} + (a-1)\log\{y\} - y/b \\
\mathrm{E}[Y] &= ab \\
\mathrm{Var}[Y] &= ab^2 \\
a &> 0 \\
b &> 0
\end{aligned}
$$

This parameterization of the gamma distribution differs from the parameterization that the GENSELECT procedure use. The scale parameter in PROC NLMOD is expressed as the inverse of the scale parameter that PROC GENSELECT uses. The PROC NLMOD parameter represents the scale of the magnitude of the residuals. You can estimate the scale parameter in PROC GENSELECT by using the following statements:

```
proc genselect;
   model y = x / dist=gamma s;
run;
```

PROC NLMOD uses the following statements to estimate the equivalent scale parameter:

```
proc nlmod;
   parms b0=1 b1=0 scale=14;
   linp = b0 + b1*x;
   mu   = exp(linp);
   b    = mu*scale;
   model y ~ gamma(1/scale,b);
run;
```

$Y \sim \text{negbin}(n, p)$

$$
\begin{aligned}
l(n, p; y) &= \log\{\Gamma(n + y)\} - \log\{\Gamma(n)\} - \log\{\Gamma(y + 1)\} \\
&\quad + n \log\{p\} + y \log\{1 - p\} \\
\text{E}[Y] &= n \left(\frac{1 - p}{p} \right) \\
\text{Var}[Y] &= n \left(\frac{1 - p}{p^2} \right) \\
n &\geq 0 \\
0 &< p < 1
\end{aligned}
$$

The value of parameter n can be a real number; it does not have to be an integer.

$Y \sim \text{Poisson}(m)$

$$
\begin{aligned}
l(m; y) &= y \log\{m\} - m - \log\{\Gamma(y + 1)\} \\
\text{E}[Y] &= m \\
\text{Var}[Y] &= m \\
m &> 0
\end{aligned}
$$

Multithreading

The NLMOD procedure allocates data to different threads and calculates the likelihood function, gradient, and Hessian by accumulating the values from all threads. For more information about how PROC NLMOD uses threads, see the section "Multithreading" on page 102 in Chapter 2, "Shared Concepts."

Optimization Algorithms

Several optimization techniques are available in PROC NLMOD. You can choose a particular optimizer by using the TECHNIQUE= option in the PROC NLMOD statement. Table 22.3 summarizes the optimization techniques available in PROC NLMOD.

Table 22.3 Optimization Techniques

TECHNIQUE=	Algorithm
TRUREG	Trust region method
NEWRAP	Newton-Raphson method with line search
NRRIDG	Newton-Raphson method with ridging
QUANEW	Quasi-Newton methods
DBLDOG	Double-dogleg method
CONGRA	Conjugate gradient methods
LEVMAR	Levenberg-Marquardt method
NMSIMP	Nelder-Mead simplex method

There is no algorithm for optimizing general nonlinear functions that always finds the global optimum for a general nonlinear optimization problem in a reasonable amount of time. Because no single optimization technique is always superior to others, PROC NLMOD provides a variety of optimization techniques that work well in various circumstances. However, you can devise problems for which none of the techniques in PROC NLMOD can find the correct solution. Moreover, nonlinear optimization can be computationally expensive in terms of time and memory, so you must be careful when matching an algorithm to a problem. The section "Choosing an Optimization Algorithm" on page 103 in Chapter 2, "Shared Concepts," is helpful in choosing a suitable optimization algorithm.

Displayed Output

The following sections describe the output that PROC NLMOD produces by default. The output is organized into various tables, which are discussed in their order of appearance.

Specifications

The "Specifications" table displays basic information about the model, such as the data source, the dependent variable, the distribution being modeled, and the optimization technique.

Number of Observations

The "Number of Observations" table displays the number of observations that are read from the input data table and the number of observations that are used in the analysis.

Dimensions

The "Dimensions" table displays the number of parameters that are estimated in the model and the number of upper and lower bounds that are imposed on the parameters.

Parameters

The "Parameters" table displays the initial values of parameters that are used to start the estimation process. You can limit this information by specifying the BEST= option in the PARAMETERS statement when you specify a large number of initial parameter value combinations. The parameter combinations and their corresponding objective function values are listed in increasing order of objective function value.

Iteration History

For each iteration of the optimization, the "Iteration History" table displays the number of function evaluations (including gradient and Hessian evaluations), the value of the objective function, the change in the objective function from the previous iteration, and the absolute value of the largest (projected) gradient element.

Convergence Status

The convergence status table is a small ODS table that appears as a message that indicates whether the optimization succeeded and which convergence criterion was met. If the optimization fails, the message indicates the reason for the failure. If you save the convergence status table to an output data set, a numeric Status variable is added that enables you to programmatically assess convergence. The values of the Status variable indicate the following:

0	Convergence was achieved or an optimization was not performed because TECHNIQUE=NONE.
1	The objective function could not be improved.
2	Convergence was not achieved because of a user interruption or because a limit (such as the maximum number of iterations or the maximum number of function evaluations) was reached. To modify these limits, see the MAXITER=, MAXFUNC=, and MAXTIME= options in the PROC NLMOD statement.

3 Optimization failed to converge because function or derivative evaluations failed at the starting values or during the iterations or because a feasible point that satisfies the parameter constraints could not be found in the parameter space.

Linear Constraints

The "Linear Constraints" table summarizes the linear constraints that are applied to the model by using the RESTRICT statements. The table lists all the constraints that are specified in the model, along with information about whether each constraint represents an inequality or equality condition and whether that constraint is active for the final parameter estimates.

Fit Statistics

The "Fit Statistics" table displays a variety of measures of fit, depending on whether the model was estimated using least squares or maximum likelihood. In both cases, smaller values of the fit statistics indicate better fit.

For least squares estimations, the table displays the sum of squares of errors and the variance of errors.

For maximum likelihood estimations, the table uses the following formulas to display information criteria, where p denotes the number of effective parameters, n denotes the number of observations used, and l is the log likelihood that is evaluated at the converged estimates:

$$\begin{aligned}
\mathrm{AIC} &= -2l + 2p \\
\mathrm{AICC} &= \begin{cases} -2l + 2pn/(n-p-1) & n > p+2 \\ -2l + 2p(p+2) & \text{otherwise} \end{cases} \\
\mathrm{BIC} &= -2l + p\log(n)
\end{aligned}$$

The information criteria values that are displayed in the "Fit Statistics" table are not based on a normalized log-likelihood function.

ANOVA

The "Analysis of Variance" (ANOVA) table is displayed only for least squares estimations. It displays the number of degrees of freedom and the sum of squares that are attributed to the model, the error, and the total. The ANOVA table also reports the variance of the model and the errors, the F statistic, and its probability for the model.

Parameter Estimates

The "Parameter Estimates" table displays the parameter estimates, their estimated (asymptotic) standard errors t statistics, and associated p-values for the hypothesis that the parameter is 0. Confidence limits, which are displayed for each parameter, are based on the value of the ALPHA= option that you specify in the PROC NLMOD statement.

Additional Estimates

The "Additional Estimates" table displays the same information as the "Parameter Estimates" table for the expressions that appear in the optional ESTIMATE statements. The table is generated when you specify one or more ESTIMATE statements. Because you can specify a separate ALPHA= option for each ESTIMATE statement, the "Additional Estimates" table also includes a column that indicates each confidence interval's corresponding significance level.

Covariance

The "Covariance" table appears when you specify the COV option in the PROC NLMOD statement. It displays a matrix of covariances between each pair of estimated parameters.

Correlation

The "Correlation" table appears when you specify the CORR option in the PROC NLMOD statement. It displays the correlation matrix for the estimated parameters.

Additional Estimates Covariance

The "Covariance of Additional Estimates" table appears when you specify the ECOV option in the PROC NLMOD statement. It displays a matrix of covariances between each pair of expressions that are specified in ESTIMATE statements.

Additional Estimates Correlation

The "Correlation of Additional Estimates" table appears when you specify the ECORR option in the PROC NLMOD statement. It displays the correlation matrix for the expressions that are specified in ESTIMATE statements.

Procedure Task Timing

The "Timing" table displays the amount of time (in seconds) that PROC NLMOD required to perform different tasks in the analysis.

ODS Table Names

Each table that the NLMOD procedure creates has a name associated with it. You must use this name to refer to the table when you use the DISPLAY statement, the DISPLAYOUT statement, or ODS statements. These names are listed in Table 22.4.

Table 22.4 ODS Tables Produced by PROC NLMOD

Table Name	Description	Statement	Option
AdditionalEstimates	Functions of estimated parameters and their associated statistics	ESTIMATE	Default
ANOVA	Least squares analysis of variance information	MODEL	RESIDUAL
Constraints	Information about the model's linear constraints	RESTRICT	Default
ConvergenceStatus	Optimization success and convergence information	PROC NLMOD	Default
CorrB	Parameter correlation matrix	PROC NLMOD	CORR
CovB	Parameter covariance matrix	PROC NLMOD	COV
Dimensions	Number of parameters and their bounds	PROC NLMOD	Default
ECorrB	Additional estimates' correlation matrix	PROC NLMOD	ECORR
ECovB	Additional estimates' covariance matrix	PROC NLMOD	ECOV
FitStatistics	Statistics about the quality of the fit	PROC NLMOD	Default
IterHistory	Optimizer iteration information	PROC NLMOD	Default
NObs	Number of observations read and used	PROC NLMOD	Default
OutputCasTables	A special ODS table that has information about all the CAS tables that are created during a CAS action execution	PROC NLMOD DISPLAYOUT	OUT=
ParameterEstimates	Parameter estimates and associated statistics	PROC NLMOD	Default
Parameters	Initial parameter values	PROC NLMOD	Default
Specifications	Basic model characteristics	PROC NLMOD	Default
Timing	Absolute and relative times for tasks performed by the procedure	PROC NLMOD	Default

Examples: NLMOD Procedure

The examples in this chapter are available in the GitHub repository located at https://github.com/sascommunities/doc-supplement-casstat.

Example 22.1: Segmented Model

(View the complete code for this example (nlmodex1.sas) in the example repository.)

Suppose you are interested in fitting a model that consists of two segments that connect in a smooth fashion. For example, the following model states that the mean of Y is a quadratic function in x for values of x less than x_0 and that the mean of Y is constant for values of x greater than x_0:

$$\mathrm{E}[Y|x] = \begin{cases} \alpha + \beta x + \gamma x^2 & \text{if } x < x_0 \\ c & \text{if } x \geq x_0 \end{cases}$$

In this model equation, α, β, and γ are the coefficients of the quadratic segment, and c is the plateau of the mean function. The NLMOD procedure can fit such a segmented model even when the join point, x_0, is unknown.

Suppose you also want to impose conditions on the two segments of the model. First, the curve should be continuous—that is, the quadratic and the plateau section need to meet at x_0. Second, the curve should be smooth—that is, the first derivative of the two segments with respect to x needs to coincide at x_0.

The continuity condition requires that

$$c = \mathrm{E}[Y|x_0] = \alpha + \beta x_0 + \gamma x_0^2$$

The smoothness condition requires that

$$\frac{\partial \mathrm{E}[Y|x_0]}{\partial x} = \beta + 2\gamma x_0 \equiv 0$$

If you solve for x_0 and substitute your result in the expression for c, the two conditions jointly imply that

$$x_0 = -\beta/2\gamma$$
$$c = \alpha - \beta^2/4\gamma$$

Although there are five unknowns, the model contains only three independent parameters. Together the continuity and smoothness restrictions completely determine two parameters, given the other three.

The following DATA step creates the data table for this example in your CAS session:

```
data mylib.A;
   input y x @@;
   datalines;
.46 1  .47  2 .57  3 .61  4 .62  5 .68  6 .69  7
.78 8  .70  9 .74 10 .77 11 .78 12 .74 13 .80 13
.80 15 .78 16
 ;
```

The following PROC NLMOD statements fit this segmented model:

```
proc nlmod data=mylib.A out=mylib.B;
   parms alpha=.45 beta=.05 gamma=-.0025;

   x0 = -.5*beta / gamma;

   if (x < x0) then
        yp = alpha + beta*x  + gamma*x*x;
   else
        yp = alpha + beta*x0 + gamma*x0*x0;

   model y ~ residual(yp);

   estimate 'join point' -beta/2/gamma;
   estimate 'plateau value c' alpha - beta**2/(4*gamma);
   predict 'predicted' yp pred=yp;
   predict 'response' y pred=y;
   predict 'x' x pred=x;
run;
```

The parameters of the model are α, β, and γ, which are represented in the PROC NLMOD statements by the variables alpha, beta, and gamma, respectively. To model the two segments, a conditional statement assigns the appropriate expression to the mean function, depending on the value of x_0. The ESTIMATE statements compute the values of x_0 and c. The PREDICT statement computes predicted values for plotting and saves them to data table b.

The results from fitting this model are shown in Output 22.1.1 through Output 22.1.3. The iterative optimization converges after six iterations (Output 22.1.1). Output 22.1.2 shows the estimated parameters. Output 22.1.3 indicates that the join point is 12.7477 and the plateau value is 0.7775.

Output 22.1.1 Nonlinear Least Squares Iterative Phase

Quadratic Model with Plateau

The NLMOD Procedure

Iteration History

Iteration	Evaluations	Objective Function	Change	Maximum Gradient
0	5	0.0035144531		7.184063
1	2	0.0007352716	0.00277918	2.145337
2	2	0.0006292751	0.00010600	0.032551
3	2	0.0006291261	0.00000015	0.002952
4	2	0.0006291244	0.00000000	0.000238
5	2	0.0006291244	0.00000000	0.000023
6	2	0.0006291244	0.00000000	2.313E-6

Convergence criterion (GCONV=1E-8) satisfied.

Output 22.1.2 Least Squares Analysis of the Quadratic Model

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Approx Pr > F
Model	2	0.1769	0.0884	114.22	<.0001
Error	13	0.0101	0.000774		
Corrected Total	15	0.1869			

Parameter Estimates

Parameter	Estimate	Standard Error	DF	t Value	Approx Pr > |t|	Approximate 95% Confidence Limits Lower	Upper
alpha	0.3921	0.0267	13	14.70	<.0001	0.3345	0.4497
beta	0.0605	0.00842	13	7.18	<.0001	0.0423	0.0787
gamma	-0.00237	0.000551	13	-4.30	0.0009	-0.00356	-0.00118

Output 22.1.3 Additional Estimates for the Quadratic Model

Additional Estimates

Label	Estimate	Standard Error	DF	t Value	Approx Pr > |t|	Alpha	Approximate Confidence Limits Lower	Upper
join point	12.7477	1.2781	13	9.97	<.0001	0.05	9.9864	15.5089
plateau value c	0.7775	0.0123	13	63.11	<.0001	0.05	0.7509	0.8041

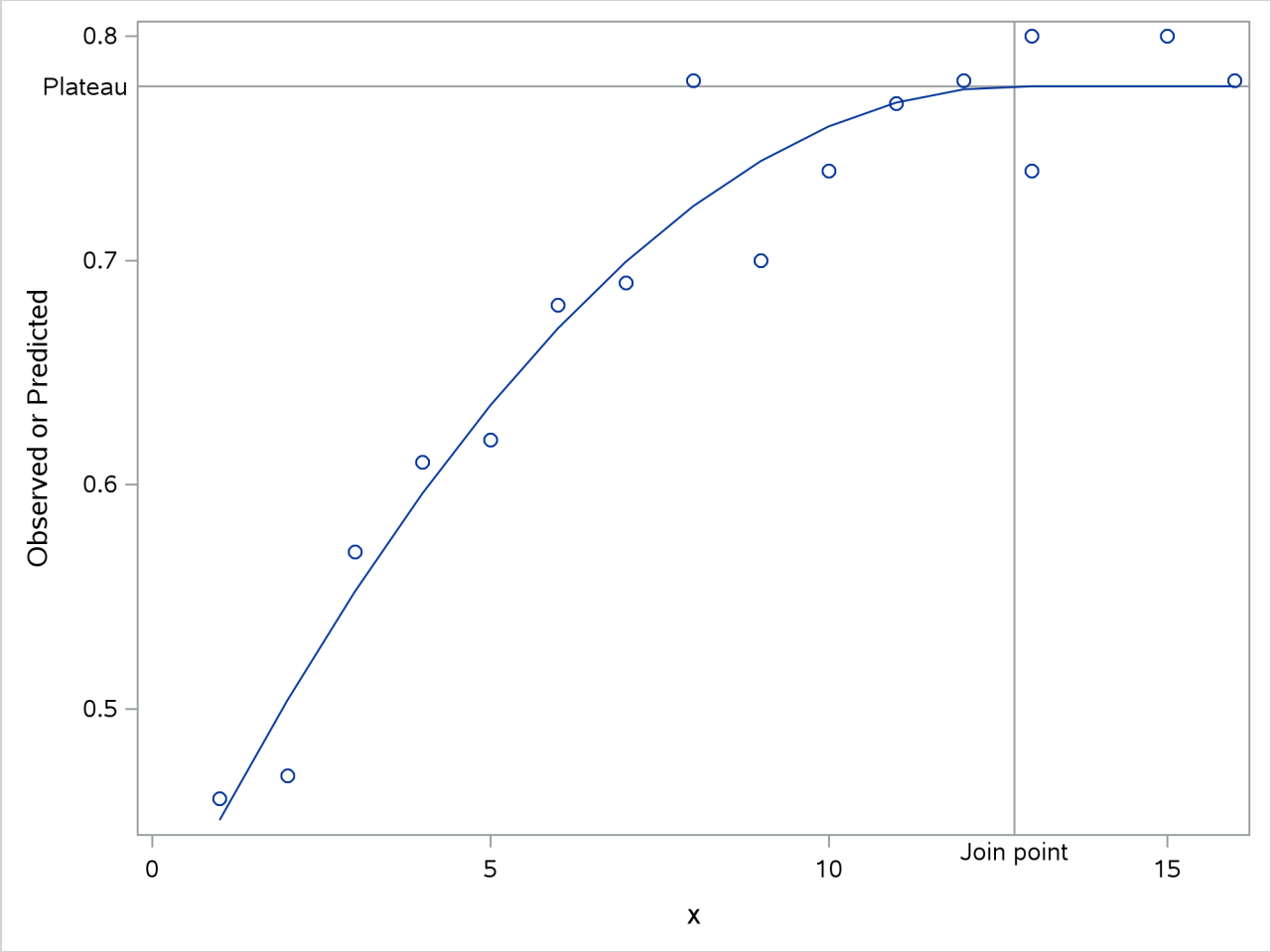
The following statements produce a graph, shown in Output 22.1.4, of the observed and predicted values along with reference lines for the join point and plateau estimates:

```
data B;
   set mylib.B;
run;

proc sort data = B;
   by x;
run;

proc sgplot data=B noautolegend;
   yaxis label='Observed or Predicted';
   refline 0.7775  / axis=y label="Plateau"    labelpos=min;
   refline 12.7477 / axis=x label="Join point" labelpos=min;
   scatter y=y  x=x;
   series  y=yp x=x;
run;
```

Output 22.1.4 Observed and Predicted Values for the Quadratic Model



References

Billingsley, P. (1986). *Probability and Measure*. 2nd ed. New York: John Wiley & Sons.

Cox, C. (1998). "Delta Method." In *Encyclopedia of Biostatistics*, edited by P. Armitage and T. Colton. New York: John Wiley & Sons.

Ratkowsky, D. (1990). *Handbook of Nonlinear Regression Models*. New York: Marcel Dekker.

Chapter 23
NMF Procedure

Contents

Overview: NMF Procedure **1234**
PROC NMF Features 1234
Using CAS Sessions and CAS Engine Librefs 1235
Getting Started: NMF Procedure **1236**
Syntax: NMF Procedure **1241**
PROC NMF Statement 1241
BY Statement 1245
DISPLAY Statement 1245
DISPLAYOUT Statement 1246
IMPUTE Statement 1247
OUTPUT Statement 1248
VAR Statement 1249
Details: NMF Procedure **1249**
Computing Matrix Factorization 1249
APG Method 1249
RANDOM Method 1250
Regularization 1251
Matrix Completion 1252
Initial Values 1253
Missing Values 1253
Displayed Output 1253
Model Information 1253
Regularization Information 1254
Number of Observations 1254
Imputation Summary 1254
Iteration Results 1254
Iteration Details 1254
Timing Information 1254
OutputCasTables Table 1254
ODS Table Names 1255
Examples: NMF Procedure **1255**
Example 23.1: Discovering Topics from Documents 1256
Example 23.2: Making Recommendations Using Matrix Completion 1260
References **1264**

Overview: NMF Procedure

The NMF procedure performs nonnegative matrix factorization in SAS Viya.

Nonnegative matrix factorization seeks to approximately decompose a nonnegative data matrix into two low-rank nonnegative factor matrices; that is, $\mathbf{X} \approx \mathbf{WH}$, where $\mathbf{X}$ is the nonnegative data matrix and $\mathbf{W}$ and $\mathbf{H}$ are the two low-rank nonnegative factor matrices. $\mathbf{W}$ is usually called the features (or basis) matrix, and $\mathbf{H}$ is usually called the weights (or coefficients) matrix, or vice versa, depending on the interpretation of the data matrix. You can use nonnegative matrix factorization for dimension reduction, feature identification, and feature extraction in applications in many different fields, such as image processing, text mining, bioinformatics, and spectral data analysis.

PROC NMF Features

The NMF procedure has the following main features:

- provides a VAR statement to specify the numeric variables for the analysis
- supports scaling and missing data imputation
- supports L_1-norm and L_2-norm regularized nonnegative matrix factorization
- produces output data tables that contain the low-dimensional factor matrices
- supports low-rank matrix completion to recover missing entries of the input data table that is incompletely sampled

PROC NMF implements the following methods:

- alternating proximal gradient
- compressed alternating proximal gradient method using random projections

Because PROC NMF runs on SAS Cloud Analytic Services (CAS), it also does the following:

- enables you to run on a cluster of machines that distribute the data and the computations
- enables you to run in single-machine mode
- exploits all the available cores and concurrent threads. For information about how PROC NMF uses threads, see the section "Multithreading" on page 102 in Chapter 2, "Shared Concepts."

Using CAS Sessions and CAS Engine Librefs

SAS Cloud Analytic Services (CAS) is the analytic server and associated cloud services in SAS Viya. This section describes how to create a CAS session and set up a CAS engine libref that you can use to connect to the CAS session. It assumes that you have a CAS server already available; contact your system administrator if you need help starting and terminating a server. This CAS server is identified by specifying the host on which it runs and the port on which it listens for communications. To simplify your interactions with this CAS server, the host information and port information for the server are stored as SAS option values that are retrieved automatically whenever this CAS server needs to be accessed. You can examine the host and port values for the server at your site by using the following statements:

```
proc options option=(CASHOST CASPORT);
run;
```

In addition to starting a CAS server, your system administrator might also have created a CAS session and a CAS engine libref for your use. You can define your own sessions and CAS engine librefs that connect to the CAS server as shown in the following statements:

```
cas mysess;
libname mylib cas sessref=mysess;
```

The CAS statement creates the CAS session named mysess, and the LIBNAME statement creates the mylib CAS engine libref that you use to connect to this session. It is not necessary to explicitly name the CASHOST and CASPORT of the CAS server in the CAS statement, because these values are retrieved from the corresponding SAS option values.

If you have created the mysess session, you can terminate it by using the TERMINATE option in the CAS statement as follows:

```
cas mysess terminate;
```

For more information about the CAS and LIBNAME statements, see the section "Introduction to Shared Concepts" on page 6 in Chapter 2, "Shared Concepts."

Getting Started: NMF Procedure

(View the complete code for this example at https://github.com/sascommunities/doc-supplement-casstat/tree/main/Examples/nmfgs.sas.)

NOTE: Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections "Using CAS Sessions and CAS Engine Librefs" on page 7 and "Loading a SAS Data Set onto a CAS Server" on page 7 in Chapter 2, "Shared Concepts."

The example in this section illustrates how to use the NMF procedure to factorize a document-by-term data matrix, where each row represents a document and each element in the document represents the weight of a certain term. The data that this example uses are generated from the Sashelp data set Sashelp.JunkMail, which contains 4,601 observations with 59 variables. The following step displays (in Figure 23.1) the variables in the Sashelp.JunkMail data set:

```
proc contents data=sashelp.JunkMail varnum;
   ods select position;
run;
```

Figure 23.1 Sashelp.JunkMail Data Set

The CONTENTS Procedure

Variables in Creation Order

#	Variable	Type	Len	Label
1	Test	Num	8	0 - Training, 1 - Test
2	Make	Num	8	
3	Address	Num	8	
4	All	Num	8	
5	_3D	Num	8	3D
6	Our	Num	8	
7	Over	Num	8	
8	Remove	Num	8	
9	Internet	Num	8	
10	Order	Num	8	
11	Mail	Num	8	
12	Receive	Num	8	
13	Will	Num	8	
14	People	Num	8	
15	Report	Num	8	
16	Addresses	Num	8	
17	Free	Num	8	
18	Business	Num	8	
19	Email	Num	8	
20	You	Num	8	
21	Credit	Num	8	
22	Your	Num	8	
23	Font	Num	8	
24	_000	Num	8	000
25	Money	Num	8	
26	HP	Num	8	
27	HPL	Num	8	
28	George	Num	8	
29	_650	Num	8	650
30	Lab	Num	8	
31	Labs	Num	8	
32	Telnet	Num	8	
33	_857	Num	8	857
34	Data	Num	8	
35	_415	Num	8	415
36	_85	Num	8	85
37	Technology	Num	8	
38	_1999	Num	8	1999
39	Parts	Num	8	
40	PM	Num	8	
41	Direct	Num	8	
42	CS	Num	8	
43	Meeting	Num	8	
44	Original	Num	8	
45	Project	Num	8	
46	RE	Num	8	

Figure 23.1 *continued*

The CONTENTS Procedure

Variables in Creation Order				
#	Variable	Type	Len	Label
47	Edu	Num	8	
48	Table	Num	8	
49	Conference	Num	8	
50	Semicolon	Num	8	
51	Paren	Num	8	
52	Bracket	Num	8	
53	Exclamation	Num	8	
54	Dollar	Num	8	
55	Pound	Num	8	
56	CapAvg	Num	8	Capital Run Length Average
57	CapLong	Num	8	Capital Run Length Longest
58	CapTotal	Num	8	Capital Run Length Total
59	Class	Num	8	0 - Not Junk, 1 - Junk

The Sashelp.JunkMail data set comes from a study that classifies whether an email is junk email or not. For the purpose of this example, two variables of the binary indicator, Test and Class, and three variables that record the lengths of uninterrupted sequences of capital letters, CapAvg, CapLong, and CapTotal, are deleted from Sashelp.JunkMail. The data set that this example uses then contains 4,601 observations (documents) with 54 variables (terms) that record the frequencies of some common words and characters in emails. The following DATA step generates the data:

```
data mylib.emails;
   set sashelp.JunkMail;
   drop test class capavg caplong captotal;
run;
```

These statements assume that your CAS engine libref is named mylib, but you can substitute any appropriately defined CAS engine libref.

The following statements invoke the NMF procedure, which requests nonnegative matrix factorization of the data, outputs the factor matrices **W** and **H** to output data tables, and produces the tables in Figure 23.2:

```
proc nmf data=mylib.emails seed=789 rank=5 outh=mylib.H;
   output out=mylib.W;
run;
```

Figure 23.2

displays the PROC NMF output. The "Model Information" table identifies the data source and shows that the nonnegative matrix factorization method is alternating proximal gradient, which is the default. Projected gradient is used as the stopping criterion, the input data are scaled to between 0 and 1, and the missing values are excluded from the analysis; these are the defaults. The table indicates that there are 54 variables used in the analysis and the target rank of factor matrices is five. The table also displays parameters that are used in the factorization method, including maximum number of iterations, the number of matrix updates at each iteration, the convergence tolerance, and the extrapolation weight coefficient. The random number seed is set to 789. Random number generation is used to initialize the factor matrices **W** and **H**. Changing the random number seed value changes the initial factor matrices, yielding different estimated factor matrices.

The "Number of Observations" table shows that all 4,601 observations in the input data are used in the analysis; all the observations are used because they all contain complete data.

The "Iteration Results" table displays the matrix factorization accuracy information, which includes the number of iterations, the relative error at which the iteration stops, and the initial and final projected gradient norms, because projected gradient is used as the stopping criterion. It also displays the sparsity of the factor matrices **W** and **H**.

Figure 23.2 Model Information, Number of Observations, and Iteration Results

The NMF Procedure

Model Information	
Data Source	EMAILS
Factorization Method	Alternating Proximal Gradient
Stopping Criterion	Projected Gradient
Scale Input Data	Yes
Missing Values	Excluded
Number of Variables	54
Target Rank of Factor Matrices	5
Maximum Number of Iterations	500
Matrix Updates at each Iteration	10
Convergence Tolerance	1E-7
Random Number Seed	789
Extrapolation Weight Coefficient	0.9999

Number of Observations Read	4601
Number of Observations Used	4601
Number of Observations with Missing Values	0

Iteration Results	
Number of Iterations	41
Relative Error	9.122635E-8
Initial Projected Gradient Norm	51894
Final Projected Gradient Norm	0.00473
Sparsity of Factor Matrix W	0.29155
Sparsity of Factor Matrix H	0.41481

The output data tables mylib.W and mylib.H contain the factor matrices **W** and **H**, respectively. Because the data that this example uses are treated as a document-by-term matrix, **H** can be seen as a features matrix that has five rows and 54 columns, and **W** can be seen as a weights matrix that has 4,061 rows and five columns. Each row in **H** is a feature, which can be considered as a document archetype that contains a set of terms. Each term's cell value defines the term's significance in the feature; the higher a term's cell value, the more the term contributes to the feature. A row in the weights matrix **W** represents an original document whose cell value defines the document's significance in a feature.

In applications of topic modeling, a feature is usually referred to as a topic. Nonnegative matrix factorization is capable of discovering topics from a document-by-term matrix or a term-by-document matrix, which would group together into a topic the terms that are somewhat coherent or probably related. Using the output data table mylib.H, you can obtain the most important terms (that is, the terms that have the largest cell values) in each topic. The following statements use the output data table mylib.H to produce the data table topics, which

contains the five discovered topics with the top 10 terms (sorted by descending cell values) to characterize each topic:

```
proc transpose data=mylib.H out=H2 name=Term prefix=Topic;
   id _Index_;
run;

%macro Top(din, rank, k, dout);
   %do i = 1 %to &rank;
      data b; set &din;
         keep Term Topic&i;
      proc sort data=b;
         by descending Topic&i;
      run;
      %if (&i = 1) %then %do;
         data &dout; merge b(obs=&k keep=Term);
            rename Term=Topic&i;
         run;
         %end;
      %else %do;
         data &dout; merge &dout b(obs=&k keep=Term);
            rename Term=Topic&i;
         run;
      %end;
   %end;
%mend;

%Top(H2, 5, 10, topics);

proc print data=topics;
run;
```

Figure 23.3

shows the contents of the topics data table.

Figure 23.3 Topics with Top 10 Terms

Obs	Topic1	Topic2	Topic3	Topic4	Topic5
1	You	HP	Your	Will	All
2	RE	George	Business	HP	Email
3	George	HPL	Our	Meeting	Exclamation
4	Exclamation	_650	Free	HPL	_000
5	Our	Labs	Internet	Address	Our
6	Edu	_85	Credit	_1999	Addresses
7	Free	Technology	Mail	Our	Free
8	Mail	Telnet	Receive	Lab	Over
9	Font	Lab	Email	Mail	Order
10	People	Direct	Make	PM	Mail

Syntax: NMF Procedure

The following statements are available in the NMF procedure:

```
PROC NMF < options > ;
    BY variables ;
    DISPLAY < table-list > < / options > ;
    DISPLAYOUT table-spec-list < / options > ;
    IMPUTE impute-options ;
    OUTPUT OUT=libref.data-table
            < COPYVARS=(variables) >
            < keyword < =prefix > > . . . < keyword < =prefix > > ;
    VAR variables ;
```

The PROC NMF statement is required. All other statements are optional. The following sections describe the PROC NMF statement and then describe the other statements in alphabetical order.

PROC NMF Statement

```
PROC NMF < options > ;
```

The PROC NMF statement invokes the NMF procedure. Table 23.1 summarizes the options available in the PROC NMF statement.

Table 23.1 PROC NMF Statement Options

Option	Description
Basic Options	
DATA=	Specifies the input data table
OUTH=	Specifies the output data table that contains the factor matrix **H**
Analysis Options	
ITDETAILS	Displays the matrix factorization accuracy for each iteration
METHOD=	Specifies the matrix factorization method to use
MISSING=	Specifies how to handle observations that have missing values
NOSCALE	Suppresses scaling of the numeric variables to analyze to between 0 and 1
PREFIX=	Specifies a prefix for naming the columns in the factor matrix **W**
RANK=	Specifies the target rank of the low-dimensional factor matrices to compute
REG=	Specifies the regularization method to use
SEED=	Specifies the seed value for pseudorandom number generation
STOP=	Specifies the stopping criterion

The following list provides details about these *options*.

DATA=*libref.data-table*
names the input data table for PROC NMF to use. The default is the most recently created data table. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to the data, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 1235.

data-table specifies the name of the input data table.

ITDETAILS
ITERDETAIL
lists the matrix factorization accuracy information for each iteration.

METHOD=APG< (*apg-options***) > | RANDOM< (***random-options***) >**
specifies the matrix factorization method to use. You can specify the following values:

APG< (*apg-options***) >**
requests the alternating proximal gradient method. You can specify the following *apg-options*:

DELTA=*n*
specifies the coefficient to use to control the extrapolation weight, where $0 < n < 1$. By default, DELTA=0.9999.

MAXITER=*n*
specifies the maximum number of iterations to perform. By default, MAXITER=500.

TOL=*n*
TOLERANCE=*n*
specifies the convergence criterion. By default, TOL=1E–7.

UPDATES=*n*
specifies the number of updates on the factor matrices $\mathbf{W}$ and $\mathbf{H}$ at each iteration. By default, UPDATES=10. If you specify the IMPUTE statement, then by default UPDATES=1.

RANDOM< (*random-options***) >**
request the compressed alternating proximal gradient method using random projections. You can specify the following *random-options*:

DELTA=*n*
specifies the coefficient to use to control the extrapolation weight, where $0 < n < 1$. By default, DELTA=0.9999.

MAXITER=*n*
specifies the maximum number of iterations to perform. By default, MAXITER=500.

OVERSAMP=*n*

OVERSAMPLING=*n*
specifies the size of oversampling. By default, OVERSAMP=10.

SUBITER=*n*
specifies the number of subspace iterations to perform. By default, SUBITER=4.

TOL=*n*

TOLERANCE=*n*
specifies the convergence criterion. By default, TOL=1E–7.

UPDATES=*n*
specifies the number of updates on the factor matrices **W** and **H** at each iteration. By default, UPDATES=10.

By default, METHOD=APG. If you specify the IMPUTE statement, only the METHOD=APG option is supported.

MISSING=MEAN | NONE
specifies how to handle observations that have missing values. You can specify the following values:

MEAN
imputes missing values to the mean of the corresponding variable.

NONE
excludes all observations that have missing values from the analysis.

By default, MISSING=NONE. If you specify the IMPUTE statement, the MISSING= option is ignored.

NOSCALE
suppresses scaling of the numeric variables to analyze to between 0 and 1.

OUTH=*libref.data-table*

OUTPUTH=*libref.data-table*
creates an output data table to contain the factor matrix **H**. The data table contains the following variables:

- the BY variables, if any
- _Index_, a variable that contains observation numbers starting from 1
- the variables that are analyzed (that is, those in the VAR statement); or, if there is no VAR statement, all numeric variables not listed in any other statement

libref.data-table is a two-level name, where *libref* refers to the library, and *data-table* specifies the name of the output data table. For more information about this two-level name, see the DATA= option and the section "Using CAS Sessions and CAS Engine Librefs" on page 1235.

PREFIX=*name*

specifies a prefix for naming the columns in the factor matrix $\mathbf{W}$. By default, the *names* are Comp1, Comp2, ..., Comp$n$. If you specify PREFIX=Abc, the components are named Abc1, Abc2, Abc3, and so on. The number of characters in the prefix plus the number of digits that are required in order to designate the variables should not exceed the current name length that is defined by the VALIDVARNAME= system option.

RANK=*number*

specifies the target rank of the low-dimensional factor matrices to compute. You must specify the *number*, which must be an integer greater than 0.

REG=L1<(*l1-options***)> | L2<(***l2-options***)>**

REGULARIZATION=L1<(*l1-options***)> | L2<(***l2-options***)>**

PENALTY=L1<(*l1-options***)> | L2<(***l2-options***)>**

specifies the regularization method to use. You can specify the following values:

L1<(*l1-options***)>**

requests the L_1-norm regularization method. You can specify the following *l1-options*:

ALPHA=*n*

specifies the regularization weight of the factor matrix $\mathbf{W}$, where $n \geq 0$. By default, ALPHA=1.

BETA=*n*

specifies the regularization weight of the factor matrix $\mathbf{H}$, where $n \geq 0$. By default, BETA=1.

L2<(*l2-options***)>**

requests the L_2-norm regularization method. You can specify the following *l2-options*:

ALPHA=*n*

specifies the regularization weight of the factor matrix $\mathbf{W}$, where $n \geq 0$. By default, ALPHA=1. This option is ignored if you specify the LCURVE *l2-option*.

BETA=*n*

specifies the regularization weight of the factor matrix $\mathbf{H}$, where $n \geq 0$. By default, BETA=1. This option is ignored if you specify the LCURVE *l2-option*.

LCURVE

uses the L-curve approach to perform L_2-norm regularization. This option is ignored if you specify the METHOD=RANDOM option, the STOP=OBJFUNC option, or the IMPUTE statement.

SEED=*number*

specifies an integer as the seed value for pseudorandom number generation. PROC NMF uses random values to initialize the factor matrices and the random projection matrices. If you do not specify this option or if *number* is less than or equal to 0, the seed is generated by reading the time of day from the computer's clock. By default, SEED=0.

STOP=PROJGRAD | OBJFUNC
STOPMEASURE=PROJGRAD | OBJFUNC
specifies the stopping criterion. You can specify the following values:

OBJFUNC
uses the objective function.

PROJGRAD
uses the projected gradient.

By default, STOP=PROJGRAD. If you specify METHOD=RANDOM, only STOP=PROJGRAD is supported.

BY Statement

BY *variables* **;**

You can specify a BY statement in PROC NMF to obtain separate analyses of observations in groups that are defined by the values of the BY variables. If you specify more than one BY statement, only the last one specified is used. For more information, see the discussion of BY-group processing in the "Grouping Data" section of *SAS Programmers Guide: Essentials*.

DISPLAY Statement

DISPLAY < *table-list* > < / *options* > **;**

The DISPLAY statement enables you to specify a list of display tables to display or exclude. This statement is similar to the ODS SELECT, ODS EXCLUDE, and ODS TRACE statements. However, the DISPLAY statement can improve performance when a large number of tables could be generated (such as in BY-group processing). The procedure processes the DISPLAY statement on a CAS server and thus sends only a subset of ODS tables to the SAS client. Because ODS statements are processed on a SAS client, first all the generated display tables are sent to the client, and then the client creates a subset.

If you use both DISPLAY and ODS statements together, the DISPLAY statement takes precedence over the ODS statements. Note that the ODS EXCLUDE statement processes tables that are sent to the client after they have been filtered by the DISPLAY statement. In some cases, it might appear that the ODS EXCLUDE statement is taking precedence because it can further filter the tables. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

You can specify the *table-list* as a list of table names, paths, partial pathnames, and regular expressions.

The table names that you can specify are listed in the section "ODS Table Names" on page 1255. A path is a table name that is prefixed with dot-separated grouping information. For example, a SelectionSummary table that a procedure produces during a selection routine might have the path *Bygroup1.Summary.SelectionSummary*. A partial pathname does not include all groups; for example, *SelectionSummary* and *Summary.SelectionSummary* are partial pathnames for *Bygroup1.Summary.SelectionSummary*.

When you specify a table name or partial pathname, all display tables whose paths end in the specified name are selected for display or exclusion. For example, both *SelectionSummary* and *Summary.SelectionSummary* select *Bygroup1.Summary.SelectionSummary*.

A regular expression is enclosed in forward slashes (/). For example, specifying "/tions/" selects all pathnames that contain the substring "tions"; in particular, the *Bygroup1.Summary.SelectionSummary* table is selected. Specifying "!/tions/" selects all pathnames that do not contain the substring "tions"; in particular, the *Bygroup1.Summary.SelectionSummary* table is not selected.

You can specify the following *options* after a slash (/):

CASESENSITIVE
performs a case-sensitive comparison of table names in the *table-list* to display table names when tables are subsetted for display. To preserve case, you must enclose table names in the *table-list* in quotation marks.

EXCLUDE
displays all display tables except those that you specify in the *table-list*.

EXCLUDEALL
suppresses display of all tables. This option takes precedence over the other options.

TRACE
displays the display table names, labels, and paths.

DISPLAYOUT Statement

DISPLAYOUT *table-spec-list* < / *options* > **;**

The DISPLAYOUT statement enables you to create output tables from your displayed output. This statement is similar to the ODS OUTPUT statement. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

The *table-spec-list* specifies a list of output tables to create. Each entry in the list has either a *key*=*value* format or a *key* format:

key=*value*	specifies *key* as the ODS table name, path, or partial pathname, and specifies *value* as the output table name.
key	specifies *key* as the ODS table name and also as the output table name.

The ODS table names that you can specify are listed in the section "ODS Table Names" on page 1255. You cannot specify the ODS table named OutputCasTables in the *table-spec-list*.

Table names and partial pathnames are discussed under the DISPLAY statement. The DISPLAYOUT statement does not support regular expressions.

You can specify the following *options* after a slash (/):

INCLUDEALL
creates output tables for all display tables. The name of the created output table is the same as the name of the corresponding display table. If you specify this option, the *table-spec-list* specification is ignored.

NOREPLACE
does not replace any existing output table of the same name.

REPEATED
replicates all output tables on all nodes.

IMPUTE Statement

IMPUTE *impute-options* **;**

The IMPUTE statement invokes low-rank matrix completion by using nonnegative matrix factorization to recover missing entries of the input data table.

You must specify exactly one of the following *impute-options*:

OUT=NONE
suppresses creation of an output data table that contains imputation results.

OUT=*libref.data-table* **<** *out-options* **>**
creates an output data table that contains imputation results. The data table contains all variables in the VAR statement (if there is no VAR statement, it contains all numeric variables not listed in any other statement), with missing values being replaced by the imputed values. You must specify OUT=*libref.data-table* before the *out-options* in the IMPUTE statement.

libref.data-table is a two-level name, where *libref* refers to the library, and *data-table* specifies the name of the output data table. For more information about this two-level name, see the DATA= option and the section "Using CAS Sessions and CAS Engine Librefs" on page 1235.

You can specify the following *out-options*:

COPYVAR=*variable*
COPYVARS=(*variables***)**
transfers one or more *variables* from the input data table to the output data table.

IMPUTEDROWSONLY
keeps only the rows that contain the imputed values in the output data table.

PREDONLY
sets the observed values to missing values in the output data table.

If you specify both the IMPUTEDROWSONLY and PREDONLY options, only the rows that contain the imputed values are kept in the output data table, and the observed values in those rows are set to missing.

OUTPUT Statement

OUTPUT OUT=*libref.data-table*
< **COPYVARS=***(variables)* >
< *keyword* < **=***prefix* > > . . . < *keyword* < **=***prefix* > > **;**

The OUTPUT statement creates a data table that contains observationwise statistics, which are computed after PROC NMF fits the model. If you do not specify a *keyword*, then only values of the factor matrix **W** are included.

The variables in the input data table are *not* included in the output data table, in order to avoid data duplication for large data tables; however, variables that you specify in the COPYVARS= option are included.

You must specify the following option:

OUT=*libref.data-table*
names the output data table for PROC NMF to use. You must specify this option before any other options. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to where the data table is to be stored, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 1235.

data-table specifies the name of the output data table.

You can also specify the following syntax elements:

COPYVAR=*variable*
COPYVARS=(*variables***)**
transfers one or more *variables* from the input data table to the output data table.

keyword **< =***prefix* **>**
specifies a statistic to include in the output data table and optionally a *prefix* for naming the output variables. If you do not provide a *prefix*, PROC NMF assigns a default prefix that is based on the type of statistic requested.

You can specify the following *keywords* to add statistics to the output data table:

COMP
COMPONENT
requests the source values for each column in the factor matrix **W**. The default prefix is the one that you specify in the PREFIX= option in the PROC NMF statement.

The output variables that contain the requested statistic are named as follows, according to the *keyword* that you specify:

- The *keyword* COMP defines an output variable for each column in the factor matrix **W**, so the variables that correspond to each successive component are named by appending a number to the prefix. For example, if you specify RANK=3 in the PROC NMF statement, then COMP=C produces the variables C1, C2, and C3.

VAR Statement

VAR *variables* **;**

The VAR statement lists the numeric variables to analyze. If you omit this statement, all numeric variables that are not specified in other statements are analyzed.

Details: NMF Procedure

Computing Matrix Factorization

Given a nonnegative data matrix $\mathbf{X}$, nonnegative matrix factorization (NMF) attempts to find two low-rank nonnegative matrices $\mathbf{W}$ and $\mathbf{H}$ such that $\mathbf{X}$ is approximated by the product of $\mathbf{W}$ and $\mathbf{H}$. Formally, NMF can be modeled by the following minimization problem,

$$\underset{\mathbf{W},\mathbf{H}}{\text{minimize}}\ f(\mathbf{W},\mathbf{H}) = \frac{1}{2}\|\mathbf{X}-\mathbf{W}\mathbf{H}\|_F^2, \quad \text{subject to } \mathbf{W}\in\mathbb{R}_+^{n\times r},\ \mathbf{H}\in\mathbb{R}_+^{r\times v}$$

where $\mathbf{X}\in\mathbb{R}_+^{n\times v}$ is the given $n\times v$ nonnegative data matrix; $\|\cdot\|_F$ is the matrix Frobenius norm; and $r<\min\{n,v\}$, which is predetermined.

The NMF procedure implements two methods for performing nonnegative matrix factorization: the alternating proximal gradient (APG) method (Guan et al. 2012; Xu and Yin 2013) and the compressed alternating proximal gradient method using random projections (RANDOM) (Tepper and Sapiro 2016; Yahaya et al. 2018).

APG Method

The alternating proximal gradient (APG) method alternatively applies the proximal gradient with Nesterov-type acceleration technique to minimize the objective function f with respect to $\mathbf{W}$ and $\mathbf{H}$, updating one matrix at a time, while keeping the other matrix fixed. At each iteration, $\mathbf{W}$ and $\mathbf{H}$ are updated by

$$\begin{aligned}
\alpha_t &= \frac{1}{2}\left(1+\sqrt{1+4\alpha_{t-1}^2}\right)\\
\mathbf{W}^t &\leftarrow \mathbf{W}^t + \min\left\{\frac{\alpha_{t-1}-1}{\alpha_t}, \delta\sqrt{\frac{L_{\mathbf{W}}^{t-1}}{L_{\mathbf{W}}^t}}\right\}(\mathbf{W}^t-\mathbf{W}^{t-1})\\
\mathbf{H}^t &\leftarrow \mathbf{H}^t + \min\left\{\frac{\alpha_{t-1}-1}{\alpha_t}, \delta\sqrt{\frac{L_{\mathbf{H}}^{t-1}}{L_{\mathbf{H}}^t}}\right\}(\mathbf{H}^t-\mathbf{H}^{t-1})\\
\mathbf{W}^{t+1} &= \max\left\{0, \mathbf{W}^t - \nabla_{\mathbf{W}} f(\mathbf{W}^t,\mathbf{H}^t)/L_{\mathbf{W}}^t\right\}\\
\mathbf{H}^{t+1} &= \max\left\{0, \mathbf{H}^t - \nabla_{\mathbf{H}} f(\mathbf{W}^{t+1},\mathbf{H}^t)/L_{\mathbf{H}}^t\right\}
\end{aligned}$$

where t is the iteration counter; $\alpha_{-1}=1$; $0<\delta<1$ is a coefficient that controls the extrapolation weight; $L_{\mathbf{W}}^t = \|\mathbf{H}^t{\mathbf{H}^t}^T\|_2$; and $L_{\mathbf{H}}^t = \|{\mathbf{W}^{t+1}}^T\mathbf{W}^{t+1}\|_2$. $L_{\mathbf{W}}$ and $L_{\mathbf{H}}$ are Lipschitz constants of the gradient functions $\nabla_{\mathbf{W}} f$ and $\nabla_{\mathbf{H}} f$, respectively.

The APG method implements two types of stopping criterion to check the convergence of the iterative algorithm: projected gradient and objective function. For the criterion that uses the projected gradient, the APG method stops the iteration after the norm of the projected gradient satisfies the condition

$$\left\| \left[\nabla^P_{\mathbf{W}} f(\mathbf{W}^t, \mathbf{H}^t), \nabla^P_{\mathbf{H}} f(\mathbf{W}^t, \mathbf{H}^t)^T \right] \right\|_F < \epsilon \left\| \left[\nabla^P_{\mathbf{W}} f(\mathbf{W}^0, \mathbf{H}^0), \nabla^P_{\mathbf{H}} f(\mathbf{W}^0, \mathbf{H}^0)^T \right] \right\|_F$$

where ϵ is a tolerance, $\mathbf{W}^0$ and $\mathbf{H}^0$ are initial values, and $\nabla^P_{\mathbf{W}} f(\mathbf{W}, \mathbf{H})$ and $\nabla^P_{\mathbf{H}} f(\mathbf{W}, \mathbf{H})$ are the projected gradient that is defined respectively by

$$\nabla^P_{\mathbf{W}} f(\mathbf{W}, \mathbf{H})_{ij} = \begin{cases} \nabla_{\mathbf{W}} f(\mathbf{W}, \mathbf{H})_{ij} & \text{if } \mathbf{W}_{ij} > 0 \\ \min\left\{0, \nabla_{\mathbf{W}} f(\mathbf{W}, \mathbf{H})_{ij}\right\} & \text{if } \mathbf{W}_{ij} = 0 \end{cases}$$

$$\nabla^P_{\mathbf{H}} f(\mathbf{W}, \mathbf{H})_{ij} = \begin{cases} \nabla_{\mathbf{H}} f(\mathbf{W}, \mathbf{H})_{ij} & \text{if } \mathbf{H}_{ij} > 0 \\ \min\left\{0, \nabla_{\mathbf{H}} f(\mathbf{W}, \mathbf{H})_{ij}\right\} & \text{if } \mathbf{H}_{ij} = 0 \end{cases}$$

The projected gradient criterion measures how close the computed solution $(\mathbf{W}^t, \mathbf{H}^t)$ is to a stationary point.

For the stopping criterion that uses the objective function, the APG method stops the iteration after the following condition is satisfied:

$$\left| \frac{f(\mathbf{W}^{t-1}, \mathbf{H}^{t-1}) - f(\mathbf{W}^t, \mathbf{H}^t)}{f(\mathbf{W}^0, \mathbf{H}^0) - f(\mathbf{W}^t, \mathbf{H}^t)} \right| < \epsilon$$

RANDOM Method

The compressed alternating proximal gradient method using random projections (RANDOM) performs NMF by using a two-step process. The first step constructs two approximate basis matrices with orthonormal columns for the range of the input data matrix $\mathbf{X}$: $\mathbf{L} \in \mathbb{R}^{n \times (r+r_o)}$ and $\mathbf{R} \in \mathbb{R}^{v \times (r+r_o)}$, where r_o is the oversampling parameter and $r + r_o \leq \min\{n, v\}$. The construction is done by using the randomized subspace iterations. The initial basis matrix is a random projection of the data matrix $\mathbf{X}$: $\mathbf{L}^0 = \mathbf{X}\boldsymbol{\Omega}_l$ and $\mathbf{R}^0 = \mathbf{X}^T\boldsymbol{\Omega}_r$, where $\boldsymbol{\Omega}_l$ and $\boldsymbol{\Omega}_r$ are random Gaussian projection matrices. Each iteration ($t = 0, 1, 2, \ldots, k$) then consists of calculating the subspace defined by the QR factorization of the random projection and orthonormalization steps,

$$\begin{aligned} \mathbf{L}^t &\leftarrow \mathrm{QR}(\mathbf{L}^t) & \mathbf{R}^t &\leftarrow \mathrm{QR}(\mathbf{R}^t) \\ \mathbf{Q}_l &\leftarrow \mathrm{QR}(\mathbf{X}^T\mathbf{L}^t) & \mathbf{Q}_r &\leftarrow \mathrm{QR}(\mathbf{X}\mathbf{R}^{\mathrm{t}}) \\ \mathbf{L}^{t+1} &= \mathbf{X}\mathbf{Q}_l & \mathbf{R}^{t+1} &= \mathbf{X}^T\mathbf{Q}_r \end{aligned}$$

where QR() is the reduced QR factorization operation that returns a matrix with orthonormal columns. When $t = k$, $\mathbf{L}^t$ and $\mathbf{R}^t$ are the desired orthonormal basis matrices.

The second step computes two compressed data matrices, $\mathbf{X}_l$ and $\mathbf{X}_r$, by using $\mathbf{L}$ and $\mathbf{R}$: $\mathbf{X}_l = \mathbf{L}^T\mathbf{X}$,

$\mathbf{X}_r = \mathbf{XR}$. $\mathbf{W}$ and $\mathbf{H}$ are then computed by the APG method with slight modifications at each iteration:

$$
\begin{aligned}
\alpha_t &= \frac{1}{2}\left(1+\sqrt{1+4\alpha_{t-1}^2}\right) \\
\mathbf{W}^t &\leftarrow \mathbf{W}^t + \min\left\{\frac{\alpha_{t-1}-1}{\alpha_t}, \delta\sqrt{\frac{L_{\mathbf{W}}^{t-1}}{L_{\mathbf{W}}^t}}\right\}(\mathbf{W}^t - \mathbf{W}^{t-1}) \\
\mathbf{H}^t &\leftarrow \mathbf{H}^t + \min\left\{\frac{\alpha_{t-1}-1}{\alpha_t}, \delta\sqrt{\frac{L_{\mathbf{H}}^{t-1}}{L_{\mathbf{H}}^t}}\right\}(\mathbf{H}^t - \mathbf{H}^{t-1}) \\
\mathbf{H}_r^t &= \mathbf{H}^t\mathbf{R} \\
\mathbf{W}^{t+1} &= \max\left\{0, \mathbf{W}^t - \nabla_{\mathbf{W}} f_1(\mathbf{W}^t, \mathbf{H}_r^t)/L_{\mathbf{W}}^t\right\} \\
\mathbf{W}_l^{t+1} &= \mathbf{L}^T\mathbf{W}^{t+1} \\
\mathbf{H}^{t+1} &= \max\left\{0, \mathbf{H}^t - \nabla_{\mathbf{H}} f_2(\mathbf{W}_l^{t+1}, \mathbf{H}^t)/L_{\mathbf{H}}^t\right\}
\end{aligned}
$$

where $f_1 = \frac{1}{2}\|\mathbf{X}_r - \mathbf{WH}_r\|_F^2$, $f_2 = \frac{1}{2}\|\mathbf{X}_l - \mathbf{W}_l\mathbf{H}\|_F^2$, $L_{\mathbf{W}}^t = \|\mathbf{H}_r^t{\mathbf{H}_r^t}^T\|_2$, and $L_{\mathbf{H}}^t = \|{\mathbf{W}_l^{t+1}}^T\mathbf{W}_l^{t+1}\|_2$.

The RANDOM method uses the projected gradient as the only stopping criterion, which stops the iteration after the norm of the projected gradient satisfies the following condition:

$$
\left\|\left[\nabla_{\mathbf{W}}^P f_1(\mathbf{W}^t, \mathbf{H}_r^t), \nabla_{\mathbf{H}}^P f_2(\mathbf{W}_l^t, \mathbf{H}^t)^T\right]\right\|_F < \epsilon\left\|\left[\nabla_{\mathbf{W}}^P f_1(\mathbf{W}^0, \mathbf{H}_r^0), \nabla_{\mathbf{H}}^P f_2(\mathbf{W}_l^0, \mathbf{H}^0)^T\right]\right\|_F
$$

Regularization

The NMF procedure implements two methods of performing regularized nonnegative matrix factorization: L_1-norm regularization and L_2-norm regularization.

L_1-norm regularization is usually applied in NMF to control the solution sparsity. L_1-norm-regularized NMF is modeled by the following minimization problem:

$$
\underset{\mathbf{W},\mathbf{H}}{\text{minimize}}\ f(\mathbf{W},\mathbf{H}) = \frac{1}{2}\|\mathbf{X}-\mathbf{WH}\|_F^2 + \alpha\|\mathbf{W}\|_1 + \beta\|\mathbf{H}\|_1, \quad \text{subject to } \mathbf{W}\in\mathbb{R}_+^{n\times r},\ \mathbf{H}\in\mathbb{R}_+^{r\times v}
$$

where $\|\cdot\|_1$ is the L_1-norm, and α and β are the regularization weights of the factor matrices $\mathbf{W}$ and $\mathbf{H}$, respectively. Because both $\mathbf{W}$ and $\mathbf{H}$ are nonnegative matrices, $\|\mathbf{W}\|_1 = \Sigma_{ij}\mathbf{W}_{ij}$ and $\|\mathbf{H}\|_1 = \Sigma_{ij}\mathbf{H}_{ij}$.

L_2-norm regularization, which is also called Tikhonov regularization, is usually applied in NMF to control the solution smoothness. L_2-norm-regularized NMF is modeled by the following minimization problem:

$$
\underset{\mathbf{W},\mathbf{H}}{\text{minimize}}\ f(\mathbf{W},\mathbf{H}) = \frac{1}{2}\|\mathbf{X}-\mathbf{WH}\|_F^2 + \frac{\alpha}{2}\|\mathbf{W}\|_F^2 + \frac{\beta}{2}\|\mathbf{H}\|_F^2, \quad \text{subject to } \mathbf{W}\in\mathbb{R}_+^{n\times r},\ \mathbf{H}\in\mathbb{R}_+^{r\times v}
$$

where α and β are the regularization weights of the factor matrices $\mathbf{W}$ and $\mathbf{H}$, respectively.

If you request L_2-norm regularization and specify REG=L2(LCURVE), PROC NMF calculates and updates the regularization weights at each iteration by using the L-curve approach to the L_2-norm-regularized NMF

with the following formulas (Mirzal 2014):

$$\alpha^{t+1} \leftarrow \frac{\|\mathbf{X} - \mathbf{W}^t\mathbf{H}^t\|_F^2}{nv\|\mathbf{W}^t\|_F^2}$$
$$\beta^{t+1} \leftarrow \frac{\|\mathbf{X} - \mathbf{W}^t\mathbf{H}^t\|_F^2}{nv\|\mathbf{H}^t\|_F^2}$$

Matrix Completion

If you specify the IMPUTE statement, the NMF procedure performs low-rank matrix completion with nonnegative matrix factorization, in order to recover missing entries of a low-rank matrix that is incompletely sampled (Nguyen, Kim, and Shim 2019; Xu and Yin 2013).

Low-rank matrix completion with NMF is modeled by the following minimization problem:

$$\underset{\mathbf{X},\mathbf{W},\mathbf{H}\geq\mathbf{0}}{\text{minimize}}\ f(\mathbf{X},\mathbf{W},\mathbf{H}) = \frac{1}{2}\|\mathbf{X} - \mathbf{W}\mathbf{H}\|_F^2, \quad \text{subject to } \mathcal{P}_\Omega(\mathbf{X}) = \mathcal{P}_\Omega(\mathbf{A})$$

where $\mathbf{A}$ is the observed low-rank nonnegative data matrix with missing entries; $\mathbf{X}$ is the low-rank nonnegative matrix that estimates $\mathbf{A}$; Ω is the index set of the observed entries of $\mathbf{A}$; and $\mathcal{P}_\Omega(\cdot)$ is the sampling operator, where $\mathcal{P}_\Omega(\mathbf{A})$ is defined as

$$[\mathcal{P}_\Omega(\mathbf{A})]_{ij} = \begin{cases} \mathbf{A}_{ij} & (i,j) \in \Omega \\ 0 & \text{otherwise} \end{cases}$$

The NMF procedure implements low-rank matrix completion by using the alternating proximal gradient (APG) method with modifications (Guan et al. 2012; Xu and Yin 2013). The modified APG method initializes the estimated matrix $\mathbf{X}$ by replacing missing values with zero in the observed data matrix $\mathbf{A}$: $\mathbf{X}^0 = \mathcal{P}_\Omega(\mathbf{A})$. At each iteration, $\mathbf{X}$ is then computed by using $\mathbf{A}$ and the latest update of $\mathbf{W}$ and $\mathbf{H}$:

$$\mathbf{X}^{t+1} = \mathcal{P}_\Omega(\mathbf{A}) + \mathcal{P}_{\Omega^c}(\mathbf{W}^{t+1}\mathbf{H}^{t+1})$$

where Ω^c is the complement of Ω and $\mathcal{P}_{\Omega^c}(\cdot)$ is the sampling operator, with $\mathcal{P}_{\Omega^c}(\mathbf{X})$ defined as

$$[\mathcal{P}_{\Omega^c}(\mathbf{X})]_{ij} = \begin{cases} 0 & (i,j) \in \Omega \\ \mathbf{X}_{ij} & \text{otherwise} \end{cases}$$

To invoke the modified APG method, you specify the METHOD=APG option in the PROC NMF statement. The METHOD=RANDOM option is not supported for low-rank matrix completion.

The NMF procedure also performs low-rank matrix completion with regularized nonnegative matrix factorization. You can specify the REG= option in the PROC NMF statement to use either the L_1-norm or L_2-norm regularization method. If the observed data matrix contains a large proportion of missing entries, you can use the regularization method to mitigate the impact of the initial factor matrices $\mathbf{W}$ and $\mathbf{H}$ on the imputation results.

Initial Values

The NMF procedure initializes the factor matrices $\mathbf{W}$ and $\mathbf{H}$ by using random values that are drawn from the uniform distribution in the interval of [0, 1]. If you specify METHOD=RANDOM, PROC NMF generates the random projection matrices by using random values that are drawn from the standard Gaussian distribution, where negative values are set to positive. You can use the SEED= option to set the seed value for the random number generator. Changing the seed value changes the initial factor matrices and the random projection matrices, resulting in different estimated factor matrices.

The initial factor matrix $\mathbf{W}$ and the random projection matrix $\mathbf{\Omega}_r$ are generated according to how the data are distributed. Changing the number of compute nodes and the number of threads that are used changes the initial distribution of data. Thus, you might get a different initial factor matrix $\mathbf{W}$ and a different random projection matrix $\mathbf{\Omega}_r$, resulting in different estimated factor matrices.

Missing Values

If you specify MISSING=NONE in the PROC NMF statement, which is the default, observations that have missing values for any variable in the VAR statement are omitted from the analysis, and they are given missing values for values of the factor matrix $\mathbf{W}$ in the OUT= data table that is specified in the OUTPUT statement. If you specify MISSING=MEAN, observations that have missing values for any variable in the VAR statement are not omitted from the analysis, and the missing values are imputed to the mean of the corresponding variable.

If you specify the IMPUTE statement, the NMF procedure performs low-rank matrix completion and the MISSING= option is ignored. Observations that have missing values for all variables in the VAR statement are omitted from the analysis, and they are set to missing in the recovered matrix $\mathbf{X}$ in the OUT= data table that you specify in the IMPUTE statement.

Displayed Output

The following sections describe the output that PROC NMF produces. The output is organized into various tables, which are discussed in their order of appearance.

Model Information

The "Model Information" table displays basic information about the model, including the input data table, the number of VAR statement variables, the target rank of factor matrices, the factorization method, and the stopping criterion used in the computation. It also displays information about the factorization method, including the maximum number of iterations, the number of matrix updates at each iteration, the convergence tolerance, the random number seed, whether to scale input data or not, the way to handle the missing values, and the coefficient for extrapolation weight. If you specify METHOD=RANDOM in the PROC NMF statement, the table also displays information about the oversampling size and the number of subspace iterations.

Regularization Information

If you specify the REG= option in the PROC NMF statement, the NMF procedure produces a "Regularization Information" table to display the regularization method and the regularization weight values.

Number of Observations

The "Number of Observations" table displays the number of observations that are read from the input data table, the number of observations that are used in the analysis, and the number of observations that have missing values. If you specify the IMPUTE statement, the table also displays the number of observations that have missing values for all variables in the VAR statement.

Imputation Summary

If you specify the IMPUTE statement, the NMF procedure produces an "Imputation Summary" table to display the degree of freedom of the recovered data matrix, the number of observed values of the data matrix, the number of imputed values, and the proportion of imputed values. The degree of freedom of an $n \times v$ matrix with rank r is $(n + v)r - r^2$, which is a lower bound of the number of observed values to ensure the exact recovery of the rank r matrix.

Iteration Results

The "Iteration Results" table displays the matrix factorization accuracy information, which includes the number of iterations, the relative error at which the iteration stops, and the stopping criterion. If you specify STOP=PROJGRAD in the PROC NMF statement, the table displays the initial and final projected gradient norms. If you specify STOP=OBJFUNC, the table displays the initial and final objective function values. The table also displays the sparsity of the factor matrices $\mathbf{W}$ and $\mathbf{H}$. The sparsity of a matrix is the proportion of the number of zero-valued elements to the total number of elements in the matrix.

Iteration Details

If you specify the ITDETAILS option in the PROC NMF statement, the NMF procedure produces an "Iteration Details" table to display the matrix factorization accuracy information for each iteration, including the iteration step, relative error, and projected gradient norm (if you specify STOP=PROJGRAD in the PROC NMF statement) or objective function value (if you specify STOP=OBJFUNC). If you specify REG=L2(LCURVE) in the PROC NMF statement, the table also displays the regularization weight values for each iteration.

Timing Information

The "Timing" table displays the elapsed time of each main task of the procedure.

OutputCasTables Table

The OutputCasTables table is a special table that has information about each CAS table that is created during a CAS action execution. The information for each CAS table consists of the CAS table name, the caslib in which the table resides, and the number of columns and rows in the CAS table. Because this table is not a typical ODS table that contains analytical results, you cannot include it in the *table-spec-list* in the DISPLAYOUT statement.

ODS Table Names

Each table that the NMF procedure creates has a name associated with it. You must use this name to refer to the table when you use the DISPLAY statement, the DISPLAYOUT statement, or the Output Delivery System (ODS) statement. These names are listed in Table 23.2.

Table 23.2 ODS Tables Produced by PROC NMF

Table Name	Description	Required Statement / Option
ImputationSummary	Summary of imputation results	IMPUTE statement
IterDetail	Accuracy information for each iteration	PROC NMF statement / ITDETAILS option
IterResult	Iteration results	Default output
ModelInfo	Model information	Default output
NObs	Number of observations read and used	Default output
OutputCasTables	See the section "OutputCasTables Table" on page 1254	DISPLAYOUT statement, OUTPUT statement, PROC NMF statement / OUTH= option, or IMPUTE statement / OUT= option
RegInfo	Regularization information	PROC NMF statement / REG= option
Timing	Absolute and relative times of tasks that are performed by the procedure	Default output

Examples: NMF Procedure

The examples in this chapter are available in the GitHub repository located at `https://github.com/sascommunities/doc-supplement-casstat`.

NOTE: Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections "Using CAS Sessions and CAS Engine Librefs" on page 7 and "Loading a SAS Data Set onto a CAS Server" on page 7 in Chapter 2, "Shared Concepts."

Example 23.1: Discovering Topics from Documents

(View the complete code for this example (nmfex1.sas) in the example repository.)

This example illustrates how you can use the NMF procedure to discover topics (main themes) from contents of the documents. The following DATA step creates the data table mylib.ex1Data, which contains 16 observations that have two variables, in your CAS session. The variable text contains the input documents, and the variable did contains the IDs of the documents. Each row in the data table represents a document for analysis.

```
data mylib.ex1Data;
   infile datalines delimiter='|' missover;
   length text $150;
   input text$ did;
   datalines;
      Reduces the cost of maintenance. Improves revenue forecast.        | 1
      Analytics holds the key to unlocking big data.                     | 2
      The cost of updates between different environments is eliminated.  | 3
      Ensures easy deployment in the cloud or on-site.                   | 4
      Organizations are turning to SAS for business analytics.           | 5
      This removes concerns about maintenance and hidden costs.          | 6
      Service-oriented and cloud-ready for many cloud infrastructures.   | 7
      Easily apply machine learning and data mining techniques to data.  | 8
      SAS Viya will address data analysis, modeling and learning.        | 9
      Helps customers reduce cost and make better decisions faster.      | 10
      Simple, powerful architecture ensures easy deployment in the cloud.| 11
      SAS is helping industries glean insights from data.                | 12
      Solve complex business problems faster than ever.                  | 13
      Shatter the barriers associated with data volume with SAS Viya.    | 14
      Casual business users, data scientists and application developers. | 15
      Serves as the basis for innovation causing revenue growth.         | 16
run;
```

These statements assume that your CAS engine libref is named mylib, but you can substitute any appropriately defined CAS engine libref.

In order to use PROC NMF to perform topic discovery, you first need to obtain a term-by-document matrix from the mylib.ex1Data data table. You can use PROC TEXTMINE to generate a data table that contains the term-by-document matrix. For more information about the TEXTMINE procedure, see Chapter 41, "TEXTMINE Procedure" (*SAS Viya: Machine Learning Procedures*).

Many text mining applications benefit from the use of a stop list, which contains a set of commonly used terms in a language. Those terms are commonly eliminated because they can be distracting and noninformative and can incur additional memory overhead. The following DATA step creates the stop list for use in PROC TEXTMINE to eliminate noisy, noninformative terms:

```
data mylib.en_stoplist;
   length Term $16;
   input Term;
   datalines;
about
and
are
```

```
as
between
for
from
in
is
of
or
than
the
this
to
with
;
```

The following statements invoke PROC TEXTMINE to run on the mylib.ex1Data data table and specify that all terms in the input document collection, except for those on the stop list, are to be kept for generating the term-by-document matrix. The summary information about the terms in the document collection is stored in a data table named mylib.terms. The term-by-document matrix is stored in a data table named mylib.termdoc.

```
proc textmine data=mylib.ex1Data;
   doc_id     did;
   variables  text;
   parse      stop      = mylib.en_stoplist
              outterms  = mylib.terms
              outparent = mylib.termdoc
              reducef   = 1;
run;
```

The mylib.termdoc data table that PROC TEXTMINE generates contains a compressed representation of the sparse term-by-document matrix, which is stored in the coordinate list (COO) format. In this format, the matrix is represented as a set of triples $(i,\ j,\ x)$, where x is an entry in the matrix and i and j denote its row and column indices, respectively. The NMF procedure does not support input data in COO format. The following statements convert the mylib.termdoc data table that is stored in COO format to a data table named mylib.termdoc2 that is stored in dense format. The mylib.termdoc2 data table contains 18 columns: ID (ID of the terms), Term, and DOC1, DOC2, . . ., DOC16 (which correspond to the 16 documents). Each row in the data table contains the term counts of a certain term in every document.

```
data terms; set mylib.terms; keep Term Key;
   rename Key=_termnum_;
run;
proc sort data=terms; by _termnum_; run;

data termdoc; set mylib.termdoc; run;
proc sort data=termdoc; by _termnum_ _document_; run;

data termdoc1; merge terms termdoc; by _termnum_;
   if missing(_count_) then delete;
run;

proc sql noprint;
   select max(_document_) into :max_docid from termdoc1;
run;
%let maxid = &max_docid;
```

```
data mylib.termdoc2;
   length id 8;
   array docs{&maxid} doc1 - doc&maxid;
   retain doc1 - doc&maxid;
   set termdoc1;
   by _termnum_;
   id = _termnum_;
   if First._termnum_ then do;
      do i= 1 to &maxid;
         docs{i} = 0;
      end;
   end;
   docs{_document_} = _count_;
   if Last._termnum_;
   drop i _termnum_ _document_ _count_;
run;
```

You can specify RANK=5 in PROC NMF to derive five topics from the mylib.termdoc2 data table. The following statements invoke the NMF procedure to run on this data table and output the factor matrices **W** and **H** to the output data tables mylib.W and mylib.H, respectively:

```
proc nmf data=mylib.termdoc2 seed=789 rank=5 outh=mylib.H;
   var doc1-doc&maxid;
   output out=mylib.W comp=Topic copyvar=Term;
run;
```

Because the mylib.termdoc2 data table is treated as a term-by-document matrix, **W** is considered a features matrix, where each column in **W** is a feature (topic) vector, and **H** is considered a weights matrix, where each column in **H** represents the topic membership values in a document. As the example in the Getting Started section shows, nonnegative matrix factorization would group together into a topic the terms that are somewhat coherent or probably related. Using the output data table mylib.W, you can obtain the most important terms (that is, the terms that have the largest cell values) in each topic. The following PROC CAS statements use the output data table mylib.W and the **table.fetch** action to obtain the top 10 terms for one of the five discovered topics, Topic3:

```
proc cas;
   action table.fetch /
      table='W'
      fetchVars={'Term', 'Topic3'}
      sortby={{name='Topic3' order='descending'}}
      to=10;
run;
quit;
```

Output 23.1.1 displays the top 10 terms for the topic Topic3 and the cell value for each term, which defines the significance of a term in the topic.

Output 23.1.1 Top 10 Terms for Topic3

Results from table.fetch

Selected Rows from Table W		
Index	Term	Topic3
1	application developer	0.1868321986
2	application	0.1868321986
3	casual	0.1868321986
4	data scientist	0.1868321986
5	scientist	0.1868321986
6	developer	0.1868321986
7	user	0.1868321986
8	business user	0.1868321986
9	business	0.1306168732
10	data	0.0698839337

You can also use the output data table mylib.W and the **table.fetch** action to generate the results table "Topics," which contains the five discovered topics with the top 10 terms (sorted by descending cell values) to characterize each topic. The following PROC CAS statements generate and display the "Topics" table:

```
proc cas;
   topic=${Topic1-Topic5};
   cols='_Index_' + topic;
   coltypes=${integer, varchar, varchar, varchar, varchar, varchar};
   Topics=newtable('Topics', cols, coltypes);

   t={};
   do i=1 to 5;
      table.fetch result=r /
         table='W'
         fetchVars={'Term'}
         sortby={{name=topic[i] order='descending'}}
         to=10;
      t[i]=r.Fetch[, 'Term'];
   end;

   row={};
   do j=1 to 10;
      row[1]=j;
      do i=1 to 5; row[i+1]=t[i][j]; end;
      addrow(Topics, row);
   end;

   print Topics;
run;
quit;
```

Output 23.1.2 shows the contents of the "Topics" results table.

Output 23.1.2 Topics with Top 10 Terms

Topics: Results

Index	Topic1	Topic2	Topic3	Topic4	Topic5
1	reduce	technique	application developer	easy deployment	viya
2	fast	apply	application	deployment	data analysis
3	cost	machine learning	casual	easy	learn
4	decision	easily	data scientist	ensure	analysis
5	customer	mine	scientist	cloud	will
6	make	machine	developer	powerful	address
7	well	learning	user	powerful architecture	model
8	help	data	business user	architecture	sas
9	revenue	unlock	business	simple	data
10	cost	hold	data	on-site	associate

Example 23.2: Making Recommendations Using Matrix Completion

(View the complete code for this example (nmfex2.sas) in the example repository.)

This example illustrates how you can use the NMF procedure to build a simple recommender system that aims at predicting the preference (or rating) of a user for an item and making recommendations based on the prediction. The data in this example are derived from the MovieLens data set, which was developed by the GroupLens project at the University of Minnesota and is available at `http://grouplens.org/datasets/movielens`. This example uses the MovieLens 100K version. You can download the compressed archive file from the website at `http://files.grouplens.org/datasets/movielens/ml-100k.zip` and use any third-party unzip tool to extract all the files in the archive to the destination directory of your choice.[1]

The file that contains the movie ratings is *u.data*, which lists four columns: user ID, item ID (each item is a movie), rating, and timestamp. The data set is very sparse because most combinations of users and movies are not rated. Assuming that the file is located in the directory */path/to/your/directory*, the following statements invoke the PYTHON procedure to load the data from the directory, store it in dense matrix format, and transfer it to a SAS data table named mylib.ratings in your CAS session. Here, the **`SAS.df2sd`** callback method is used in PROC PYTHON to transfer data from a Python Pandas DataFrame to a SAS data set; the CAS engine libref is named mylib, but you can substitute any appropriately defined CAS engine libref. The mylib.ratings data table contains the following columns: UserID and M1, M2, M3, etc. (which correspond to all the movies). Each row in the data table contains the ratings that a certain user gave to the movies. If the user did not rate a movie, the corresponding rating is missing.

[1] Disclaimer: SAS may reference other websites or content or resources for use at Customer's sole discretion. SAS has no control over any websites or resources that are provided by companies or persons other than SAS. Customer acknowledges and agrees that SAS is not responsible for the availability or use of any such external sites or resources, and does not endorse any advertising, products, or other materials on or available from such websites or resources. Customer acknowledges and agrees that SAS is not liable for any loss or damage that may be incurred by Customer or its end users as a result of the availability or use of those external sites or resources, or as a result of any reliance placed by Customer or its end users on the completeness, accuracy, or existence of any advertising, products, or other materials on, or available from, such websites or resources.

```
proc python;
   submit;

import pandas as pd
import numpy as np

# load data from the file
colname = ['userID', 'movieID', 'rating']
colpick = [0, 1, 2]
df = pd.read_csv('u.data', delimiter='\t', usecols=colpick, names=colname)

# store data in dense matrix format
nrow = max(df.loc[:, 'userID'])
ncol = max(df.loc[:, 'movieID']) + 1
mat = np.full((nrow, ncol), np.nan)

for i in range(0, nrow):
   mat[i, 0] = i+1

for idx, rowSeries in df.iterrows():
   val = rowSeries.values
   mat[val[0]-1, val[1]] = val[2]

# transfer data to a SAS data table
cols = ['UserID'] + ['M%d' %i for i in range(1, ncol)]
matdf = pd.DataFrame(mat, columns=cols)

SAS.df2sd(matdf, 'mylib.ratings')

   endsubmit;
run;
```

You specify the IMPUTE statement in the NMF procedure to enable low-rank matrix completion to recover the missing entries in the mylib.ratings data table. The following statements invoke the NMF procedure for this data table and output the imputation results to the output data table mylib.outX. Because the file that contains the movie genres, *u.genre*, lists 19 genres, the PROC NMF statement specifies RANK=19 to compute 19 feature vectors during the factorization. The mylib.ratings data table is very sparse, so to ensure convergence for low-rank matrix completion and mitigate the impact of the initial factor matrices on the imputation results, the PROC NMF statement specifies 600 as the maximum number of iterations for the APG method by using the MAXITER= suboption and requests the L_2-norm regularization method by using the REG= option. The IMPUTEDROWSONLY and PREDONLY options in the IMPUTE statement keep only the rows that contain the imputed values in the output data table mylib.outX and set the observed values in those rows as missing values.

```
proc nmf data=mylib.ratings rank=19 seed=6789
         method=apg(maxiter=600) reg=L2(alpha=5 beta=5);
   var m:;
   impute out=mylib.outX imputedRowsOnly predOnly copyvar=UserID;
run;
```

The following statements invoke PROC PYTHON to fetch from the mylib.outX data table the first 10 observations that contain the predicted ratings of the first 10 users (the **SAS.sd2df** callback method is used

in PROC PYTHON to transfer data from a SAS data set to a Python Pandas DataFrame); load information about the movies from the file *u.item* that is located in the directory */path/to/your/directory*; produce the top 10 recommended movies for the 9th user; and generate a table that contains the top 5 recommended movies for each of the first 10 users:

```
proc python;
   submit;

import pandas as pd
import numpy as np
import csv

# fetch the first 10 observations
df = SAS.sd2df('mylib.outX(obs=10)')

# load information about the movies
movieDict = {}
csvFile = csv.reader(open('u.item', encoding='latin-1'), delimiter='|')
for row in csvFile:
   key = 'M' + row[0]
   movieDict[key] = row[1]

# top 10 recommended movies for a single user
row = 8
uid = df.iloc[row, 0]
rating = df.iloc[row, 1:].sort_values(ascending=False, inplace=False)
colUid = [uid]*10
colRank = np.arange(1, 11).tolist()
colMid = rating.index.tolist()[0:10]
colRate = rating.values.tolist()[0:10]
colTitle = []
for i in range(0, 10):
   colTitle.append(movieDict[rating.index[i]])

cols = ['UserID', 'Rank', 'MovieID', 'Title', 'PredictedRating']
topRating = pd.DataFrame(list(zip(colUid, colRank, colMid, colTitle, colRate)),
                         columns=cols)
SAS.df2sd(topRating, 'topRating')

# top 5 recommended movies for each of the 10 users
movies = []
for idx, rowSeries in df.iterrows():
   uid = rowSeries.pop('UserID')
   rowSeries.sort_values(ascending=False, inplace=True)
   row = [uid]
   for i in range(0, 5):
      row.append(movieDict[rowSeries.index[i]])
   movies.append(row)

cols = ['UserID', '_1_', '_2_', '_3_', '_4_', '_5_']
topRecom = pd.DataFrame(movies, columns=cols)
SAS.df2sd(topRecom, 'topRecom')
```

```
   endsubmit;
run;
```

The following statements print the top 10 recommended movies along with the predicted ratings for the 9th user, as shown in Output 23.2.1:

```
proc print data=topRating;
run;
```

Output 23.2.1 Top 10 Recommended Movies with Predicted Ratings for Single User

Obs	UserID	Rank	MovieID	Title	PredictedRating
1	9	1	M313	Titanic (1997)	4.08736
2	9	2	M272	Good Will Hunting (1997)	4.07087
3	9	3	M302	L.A. Confidential (1997)	4.03031
4	9	4	M127	Godfather, The (1972)	3.99586
5	9	5	M318	Schindler's List (1993)	3.97899
6	9	6	M64	Shawshank Redemption, The (1994)	3.92072
7	9	7	M100	Fargo (1996)	3.89836
8	9	8	M515	Boot, Das (1981)	3.87869
9	9	9	M269	Full Monty, The (1997)	3.84054
10	9	10	M315	Apt Pupil (1998)	3.80159

The following statements print the top 5 recommended movies (sorted by descending predicted ratings) for each of the 10 users, as shown in Output 23.2.2:

```
proc print data=topRecom;
run;
```

Output 23.2.2 Top 5 Recommended Movies for 10 Users

Obs	UserID	_1_	_2_	_3_	_4_	_5_
1	1	Close Shave, A (1995)	Casablanca (1942)	Secrets & Lies (1996)	L.A. Confidential (1997)	Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb (1963)
2	2	Schindler's List (1993)	Shawshank Redemption, The (1994)	Boot, Das (1981)	Close Shave, A (1995)	Casablanca (1942)
3	3	Star Wars (1977)	Titanic (1997)	Godfather, The (1972)	Full Monty, The (1997)	Fargo (1996)
4	4	Titanic (1997)	Good Will Hunting (1997)	L.A. Confidential (1997)	Full Monty, The (1997)	Apt Pupil (1998)
5	5	L.A. Confidential (1997)	Godfather, The (1972)	Usual Suspects, The (1995)	Secrets & Lies (1996)	Pulp Fiction (1994)
6	6	Rear Window (1954)	Chinatown (1974)	Manchurian Candidate, The (1962)	Clockwork Orange, A (1971)	Wallace & Gromit: The Best of Aardman Animation (1996)
7	7	Wrong Trousers, The (1993)	Close Shave, A (1995)	Wallace & Gromit: The Best of Aardman Animation (1996)	Philadelphia Story, The (1940)	Some Folks Call It a Sling Blade (1993)
8	8	Titanic (1997)	Shawshank Redemption, The (1994)	Good Will Hunting (1997)	Schindler's List (1993)	Fargo (1996)
9	9	Titanic (1997)	Good Will Hunting (1997)	L.A. Confidential (1997)	Godfather, The (1972)	Schindler's List (1993)
10	10	Schindler's List (1993)	Close Shave, A (1995)	Good Will Hunting (1997)	Boot, Das (1981)	To Kill a Mockingbird (1962)

References

Guan, N., Tao, D., Luo, Z., and Yuan, B. (2012). "NeNMF: An Optimal Gradient Method for Nonnegative Matrix Factorization." *IEEE Transactions on Signal Processing* 60:2882–2898.

Mirzal, A. (2014). "Nonparametric Tikhonov Regularized NMF and Its Application in Cancer Clustering." *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 11:1208–1217.

Nguyen, L. T., Kim, J., and Shim, B. (2019). "Low-Rank Matrix Completion: A Contemporary Survey." *IEEE Access* 7:94215–94237.

Tepper, M., and Sapiro, G. (2016). "Compressed Nonnegative Matrix Factorization Is Fast and Accurate." *IEEE Transactions on Signal Processing* 64:2269–2283.

Xu, Y., and Yin, W. (2013). "A Block Coordinate Descent Method for Regularized Multiconvex Optimization with Applications to Nonnegative Tensor Factorization and Completion." *SIAM Journal on Imaging Sciences* 6:1758–1789.

Yahaya, F., Puigt, M., Delmaire, G., and Roussel, G. (2018). "Faster-Than-Fast NMF Using Random Projections and Nesterov Iterations." In *Proceedings of iTWIST: International Traveling Workshop on Interactions between Low-Complexity Data Models and Sensing Techniques*. Marseille, France.

Chapter 24
PARTITION Procedure

Contents

Overview: PARTITION Procedure **1266**
Using CAS Sessions and CAS Engine Librefs 1266
Getting Started: PARTITION Procedure **1267**
Syntax: PARTITION Procedure **1268**
PROC PARTITION Statement 1268
BY Statement 1270
DISPLAY Statement 1270
DISPLAYOUT Statement 1271
FREQ Statement 1272
OUTPUT Statement 1272
TARGET Statement 1273
Details: PARTITION Procedure **1274**
Folds 1274
Training, Validation, and Test Data 1274
Classification Level 1274
Displayed Output 1275
ODS Table Names 1275
Examples: PARTITION Procedure **1276**
Example 24.1: Simple Random Sampling 1277
Example 24.2: Stratified Sampling 1278
Example 24.3: Oversampling 1280
Example 24.4: *k*-Fold Partitioning 1281
Example 24.5: Segment-Stratified Sampling for Each Target 1283
Example 24.6: Segment-Stratified Sampling for Each Target Using the FREQ Statement 1285

Overview: PARTITION Procedure

The PARTITION procedure performs sampling in SAS Viya. It performs simple random sampling, stratified sampling, oversampling, or *k*-fold partitioning to produce a table that contains a subset of the observations or that contains partitioned observations.

The PARTITION procedure creates the following:

- one output data table, which contains the subset or partitioned data table
- one summary table, which contains the numbers of observations and variables in the output data table
- one frequency table, which contains the frequency information for the population and sample or for each fold for *k*-fold partitioning
- one partition indicator map table, which contains the mapping of each partition indicator to each target. This table is created when the TARGET statement is specified.
- one sample frequency map table, which contains the mapping of each sample frequency to each target. This table is created when both the TARGET statement and the FREQ statement are specified.

Using CAS Sessions and CAS Engine Librefs

SAS Cloud Analytic Services (CAS) is the analytic server and associated cloud services in SAS Viya. This section describes how to create a CAS session and set up a CAS engine libref that you can use to connect to the CAS session. It assumes that you have a CAS server already available; contact your system administrator if you need help starting and terminating a server. This CAS server is identified by specifying the host on which it runs and the port on which it listens for communications. To simplify your interactions with this CAS server, the host information and port information for the server are stored as SAS option values that are retrieved automatically whenever this CAS server needs to be accessed. You can examine the host and port values for the server at your site by using the following statements:

```
proc options option=(CASHOST CASPORT);
run;
```

In addition to starting a CAS server, your system administrator might also have created a CAS session and a CAS engine libref for your use. You can define your own sessions and CAS engine librefs that connect to the CAS server as shown in the following statements:

```
cas mysess;
libname mylib cas sessref=mysess;
```

The CAS statement creates the CAS session named mysess, and the LIBNAME statement creates the mylib CAS engine libref that you use to connect to this session. It is not necessary to explicitly name the CASHOST and CASPORT of the CAS server in the CAS statement, because these values are retrieved from the corresponding SAS option values.

If you have created the mysess session, you can terminate it by using the TERMINATE option in the CAS statement as follows:

```
cas mysess terminate;
```

For more information about the CAS and LIBNAME statements, see the section "Introduction to Shared Concepts" on page 6 in Chapter 2, "Shared Concepts."

Getting Started: PARTITION Procedure

(View the complete code for this example at https://github.com/sascommunities/doc-supplement-casstat/tree/main/Examples/partition_gs.sas.)

NOTE: Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections "Using CAS Sessions and CAS Engine Librefs" on page 7 and "Loading a SAS Data Set onto a CAS Server" on page 7 in Chapter 2, "Shared Concepts."

This example performs stratified partitioning of 5,960 fictitious mortgages, with the BY variable BAD used as the stratum. The input data table mylib.hmeq includes information about the fictitious mortgages. Each observation represents an applicant for a home equity loan, and all applicants have an existing mortgage.

You can load the sampsio.hmeq data set into your CAS session by naming your CAS engine libref in the first statement of the following DATA step. This DATA step assumes that your CAS engine libref is named mylib, but you can substitute any appropriately defined CAS engine libref.

```
data mylib.hmeq;
   set sampsio.hmeq;
run;
```

The following statements perform the partitioning:

```
proc partition data=mylib.hmeq samppct=10 samppct2=20 seed=1234 partind;
   by BAD;
   output out=mylib.out1 copyvars=(BAD loan derog mortdue value yoj
          delinq clage ninq clno debtinc);
run;
```

The SAMPPCT=10 option requests that 10% of the input data be included in the training partition, and the SAMPPCT2=20 option requests that 20% of the input data be included in the testing partition. The SEED= option specifies 1234 as the random seed to be used in the partitioning process. The PARTIND option requests that the output data table, mylib.out1, include an indicator that shows whether each observation is selected to a partition (1 for training or 2 for testing) or not (0). The binary BY variable BAD indicates whether an applicant eventually defaulted or was ever seriously delinquent. The BY statement triggers stratified sampling, which enables you to sample each subpopulation in the BY variable (stratum) independently. The OUTPUT statement creates a new data table to contain the variables from the input data table that are listed in the COPYVARS= option and the partition indicator. The displayed output includes a frequency table (Figure 24.1) that shows the frequency of observations in each level of BAD.

Figure 24.1 Frequency Information Table

The PARTITION Procedure

Stratified Sampling Frequency				
Index	BAD	Number of Obs	Sample Size 1	Sample Size 2
0	0	4771	477	954
1	1	1189	119	238

Syntax: PARTITION Procedure

The following statements are available in the PARTITION procedure:

PROC PARTITION < *options* > **;**
BY *variable* < *variable . . . variable* > **;**
DISPLAY < *table-list* > < */options* > **;**
DISPLAYOUT *table-spec-list* < */options* > **;**
FREQ *variable* **;**
OUTPUT OUT=*libref.data-table* < *options* > **;**
TARGET *variable* < / **LEVEL=NOMINAL** | **INTERVAL** > **;**

The BY statement is required for stratified sampling and oversampling. The OUTPUT statement is required for all sampling methods. The FREQ statement is supported in simple random sampling, stratified sampling, and oversampling.

PROC PARTITION Statement

PROC PARTITION < *options* > **;**

The PROC PARTITION statement invokes the procedure.

You can specify the following *options* for all types of sampling:

APPLYROWORDER
uses a data distribution and row order as determined by a previous **`partition`** action call. For more information, see the section "The APPLYROWORDER Option" on page 107 in Chapter 2, "Shared Concepts."

DATA=*libref.data-table*
names the input data table for PROC PARTITION to use. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to the data, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 1266.

data-table specifies the name of the input data table.

NTHREADS=*number-of-threads*
specifies the number of threads to be used. The default is the CPU count on each node.

PARTIND
adds to the output data table a partition indicator, _PartInd_, which indicates whether an observation is selected to a partition (1 or 2) or not (0).

SEED=*random-seed*
specifies an integer that is used to start the pseudorandom number generator. If you do not specify a seed or you specify a value less than or equal to 0, the seed is generated from reading the time of day from the computer's clock. The SEED= option enables you to reproduce the same sample output.

You can specify the following *options* only for simple random sampling and stratified sampling:

FIXEDOBS=*fixed-observations*
specifies the number of samples. In simple random sampling, this is the total number of samples. In stratified sampling, this is the number of samples from each stratum. When you specify this option, the SAMPPCT= option and SAMPPCT2= option are ignored. When you specify this option along with the PARTIND option, the sampled observations have a _PartInd_ of 1.

SAMPPCT=*sample-percentage*
specifies what percentage of the data is to be sampled, where *sample-percentage* is a positive number less than or equal to 100. For example, SAMPPCT=50.5 samples 50.5% of data. When you specify this option along with the PARTIND option, the percentage of the sample whose _PartInd_ is 1 is specified in *sample-percentage*, and the percentage of the sample whose _PartInd_ is 0 is 100 minus *sample-percentage*.

SAMPPCT2=*sample-percentage-2*
partitions the input data into three parts when specified along with the SAMPPCT= and PARTIND options. The percentage of the sample whose _PartInd_ is 1 is specified in the SAMPPCT= option, the percentage of the sample whose _PartInd_ is 2 is specified in the SAMPPCT2= option, and the percentage of the sample whose _PartInd_ is 0 is 100 minus the sum of the values of the SAMPPCT= and SAMPPCT2= options. The sum of the *sample-percentages* that you specify in the SAMPPCT= and SAMPPCT2= options must be a positive number less than or equal to 100.

The following formulas are used to calculate sample sizes for partitions:

$$n = \mathrm{round}((\textit{sample-percentage} + \textit{sample-percentage-2}) \times N)$$

$$n_1 = \mathrm{round}((\textit{sample-percentage}/(\textit{sample-percentage} + \textit{sample-percentage-2})) \times n$$

$$n_2 = n - n_1$$

where N is the number of observations in the input data, n is the total number of observations in the partitions, n_1 is the number of observations in the first sample, and n_2 is the number of observations in the second sample.

You can specify the following *options* only for oversampling:

EVENT=*"rare-event-level"*
specifies the level (category) that corresponds to a rare event of the variable that is specified in the BY statement. If you have a format applied to the variable, you should specify the formatted event level in this option. If you specify this option, PROC PARTITION uses an oversampling technique to adjust the class distribution of the data, and the following two options are required.

EVENTPROP=*event-proportion*
specifies the proportion of rare events that you want in the sample, where *event-proportion* is a positive number less than or equal to 1. For example, EVENTPROP=0.3 specifies that 30% of the sample are rare events and 70% are not rare.

SAMPPCTEVT=*sample-event-percentage*
specifies the sample percentage from the rare event level, where *sample-event-percentage* is a positive number less than or equal to 100. For example, SAMPPCTEVT=50.5 specifies that you want to sample 50.5% of the rare events.

You can specify the following *option* only for *k*-fold partitioning:

KFOLD=*k*
specifies the number of folds you want to partition the data into, where *k* is a positive integer greater than or equal to 2.

BY Statement

BY *variable* < *variable . . . variable* > **;**

The BY statement specifies classification variables to use for stratification. There is no limit on the BY variables for stratified sampling or *k*-fold partitioning, but only one BY variable is allowed for oversampling. If the TARGET statement is specified, each target variable is combined with the BY variables for stratification in order to perform stratified partitioning.

DISPLAY Statement

DISPLAY < *table-list* > < / *options* > **;**

The DISPLAY statement enables you to specify a list of display tables to display or exclude. This statement is similar to the ODS SELECT, ODS EXCLUDE, and ODS TRACE statements. However, the DISPLAY statement can improve performance when a large number of tables could be generated (such as in BY-group processing). The procedure processes the DISPLAY statement on a CAS server and thus sends only a subset of ODS tables to the SAS client. Because ODS statements are processed on a SAS client, first all the generated display tables are sent to the client, and then the client creates a subset.

If you use both DISPLAY and ODS statements together, the DISPLAY statement takes precedence over the ODS statements. Note that the ODS EXCLUDE statement processes tables that are sent to the client after they have been filtered by the DISPLAY statement. In some cases, it might appear that the ODS EXCLUDE statement is taking precedence because it can further filter the tables. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

You can specify the *table-list* as a list of table names, paths, partial pathnames, and regular expressions.

The table names that you can specify are listed in the section "ODS Table Names" on page 1275. A path is a table name that is prefixed with dot-separated grouping information. For example, a SelectionSummary table that a procedure produces during a selection routine might have the path *Bygroup1.Summary.SelectionSummary*. A partial pathname does not include all groups; for example, *SelectionSummary* and *Summary.SelectionSummary* are partial pathnames for *Bygroup1.Summary.SelectionSummary*.

When you specify a table name or partial pathname, all display tables whose paths end in the specified name are selected for display or exclusion. For example, both *SelectionSummary* and *Summary.SelectionSummary* select *Bygroup1.Summary.SelectionSummary*.

A regular expression is enclosed in forward slashes (/). For example, specifying "/tions/" selects all pathnames that contain the substring "tions"; in particular, the *Bygroup1.Summary.SelectionSummary* table is selected. Specifying "!/tions/" selects all pathnames that do not contain the substring "tions"; in particular, the *Bygroup1.Summary.SelectionSummary* table is not selected.

You can specify the following *options* after a slash (/):

CASESENSITIVE
performs a case-sensitive comparison of table names in the *table-list* to display table names when tables are subsetted for display. To preserve case, you must enclose table names in the *table-list* in quotation marks.

EXCLUDE
displays all display tables except those that you specify in the *table-list*.

EXCLUDEALL
suppresses display of all tables. This option takes precedence over the other options.

TRACE
displays the display table names, labels, and paths.

DISPLAYOUT Statement

DISPLAYOUT *table-spec-list* < / *options* > **;**

The DISPLAYOUT statement enables you to create output tables from your displayed output. This statement is similar to the ODS OUTPUT statement. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

The *table-spec-list* specifies a list of output tables to create. Each entry in the list has either a *key*=*value* format or a *key* format:

key=*value*	specifies *key* as the ODS table name, path, or partial pathname, and specifies *value* as the output table name.
key	specifies *key* as the ODS table name and also as the output table name.

The ODS table names that you can specify are listed in the section "ODS Table Names" on page 1275. You cannot specify the ODS table named OutputCasTables in the *table-spec-list*.

Table names and partial pathnames are discussed under the DISPLAY statement. The DISPLAYOUT statement does not support regular expressions.

You can specify the following *options* after a slash (/):

INCLUDEALL
creates output tables for all display tables. The name of the created output table is the same as the name of the corresponding display table. If you specify this option, the *table-spec-list* specification is ignored.

NOREPLACE
does not replace any existing output table of the same name.

REPEATED
replicates all output tables on all nodes.

FREQ Statement

FREQ *variable* **;**

The FREQ statement specifies a numeric variable in the data set that contains the frequency of occurrence of each observation. In simple random sampling, stratified sampling, and oversampling, the PARTITION procedure treats each observation as if it appears f times, where f is the value of the FREQ statement variable for the observation. If f is not an integer, it is truncated to an integer. If f is less than 1 or missing, the observation is not used for sampling or partitioning. When you omit this statement, each observation is assigned a frequency of 1.

OUTPUT Statement

OUTPUT OUT=*libref.data-table* < *options* > **;**

The OUTPUT statement creates a data table to contain the sampled observations and stratification information for the sample.

You must specify the following option:

OUT=*libref.data-table*
names the output data table for PROC PARTITION to use. You must specify this option before any other options. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to where the data table is to be stored, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 1266.

data-table specifies the name of the output data table.

This table includes variables that are specified in the COPYVARS= option. In the K-fold partition case, the output includes one more column, _Fold_. In other cases, if you also specify the PARTIND option in the PROC PARTITION statement, the output includes one more column, _PartInd_. In the oversampling case, an additional column, _Freq_, is provided. It is calculated as the ratio of target level's proportion in the population to its proportion in the sample. _Fold_, _PartInd_ and _Freq_ are reserved names. If they exist in the input data table and you specify them in the COPYVARS= option, you need to use the FOLDNAME=, PARTINDNAME= and FREQNAME= options to change the generated columns' names.

You can also specify the following *options*:

COPYVAR=*variable*

COPYVARS=(*variables***)**

lists one or more variables from the input data table to be transferred to the output data table.

If the COPYVARS= option is not specified, all the variables in the input data table are transferred to the output data table.

FOLDNAME=*fold-name*

renames the output data table's _Fold_ column (which is generated when *k*-fold partitioning is requested) to the specified *fold-name*.

FREQNAME=*freq-name*

renames the output data table's _Freq_ column (which is generated when oversampling is requested) to the specified *freq-name*.

PARTINDNAME=*partition-indicator-name*

renames the generated partition indicator, _PartInd_, in the output data table to the specified *partition-indicator-name.*

SAMPLEFREQNAME=*sample-frequency-name*

renames the generated sample frequency variable, _SampleFreq_, in the output data table by using the specified *sample-frequency-name.*

TARGET Statement

TARGET *variable* < / **LEVEL=NOMINAL** | **INTERVAL** > **;**

If BY (segment) variables are present, the TARGET statement performs segment-stratified partitioning for each target variable as follows: each nominal target variable is combined with the BY variables for stratification; for each interval target variable, only the BY variables are used for stratification. If BY (segment) variables are not present, the TARGET statement performs stratified partitioning for each nominal target variable and simple random sampling for each interval target variable. If target variables or BY variables have lots of levels, the memory consumption can be huge and an insufficient memory issue might occur. In this case, decreasing the number of threads by using NTHREADS= option in the PROC PARTITION statement might help because memory consumption depends on the number of threads.

You can specify the following option:

LEVEL=NOMINAL | INTERVAL

specifies the level of measurement. You can specify the following values:

NOMINAL specifies that the level of measurement of the variables is nominal.

INTERVAL specifies that the level of measurement of the variables is interval.

By default, LEVEL=INTERVAL for numeric variables and LEVEL=NOMINAL for categorical variables.

Details: PARTITION Procedure

Folds

You can use the PARTITION procedure to partition the input data table into equal-size disjoint subsets for *k*-fold cross validation. When you specify the number of folds *k* in the KFOLD= option, the _Fold_ column in the output data table shows which fold each observation is selected to. _Fold_ takes values between 1 and *k*.

Training, Validation, and Test Data

You can use the PARTITION procedure to partition the input data table into disjoint subsets for model training, validation, and testing. When you specify the percentages for training and testing partitions in the SAMPPCT= and SAMPPCT2= options, the _PartInd_ indicator shows whether each observation is selected to the training partition (1) or the testing partition (2), respectively. The remaining observations are assigned 0 for _PartInd_ and belong to the validation partition.

Classification Level

For classification variables, a classification level is an observed value that is distinct after beginning and ending white space have been removed. For example, the values "**`MyLevel`**" and "**`MYLEVEL`**" are treated as different levels in the data table. But," a" and "a " are treated as the same level.

Displayed Output

The following sections describe the output that PROC PARTITION produces. The output is organized into various tables, which are discussed in the order of their appearance.

Frequency Information Table

For simple random sampling, the "Simple Random Sampling Frequency" table lists the number of observations in the input data table and in the sample output data table.

For stratified sampling, the "Stratified Sampling Frequency" table lists the respective frequency in each stratum for the input data and the sample. If one BY variable is specified, each level of the BY variable represents a stratum; if two BY variables are specified, a combination of the levels of two BY variables represents a stratum.

For oversampling, the "Oversampling Frequency" table lists the number of observations in each stratum for the input data and the sample.

For *k*-fold partitioning, the "*k*-Fold Partition Frequency" table lists the input data size and fold sizes. Depending on whether you specify "options VALIDVARNAME=V7;" or "options VALIDVARNAME=ANY;", the names of the NFold variables in the frequency table KFOLDFreq are different. If you specify "options VALIDVARNAME=V7;", the names of the NFold variables in the frequency table KFOLDFreq are "NFold_i", where i is the fold number. If you specify "options VALIDVARNAME=ANY;", the names are "NFold i", where i is the fold number.

OutputCasTables Table

The OutputCasTables table is a special table that has information about each CAS table that is created during a CAS action execution. The information for each CAS table consists of the CAS table name, the caslib in which the table resides, and the number of columns and rows in the CAS table. Because this table is not a typical ODS table that contains analytical results, you cannot include it in the *table-spec-list* in the DISPLAYOUT statement.

ODS Table Names

Each table that the PARTITION procedure creates has a name associated with it. You must use this name to refer to the table when you use the DISPLAY statement or ODS statements. These names are listed in Table 24.1.

Table 24.1 ODS Tables Produced by PROC PARTITION

Table Name	Description	Statements	Option
KFOLDFreq	Frequency table of input data size and fold sizes	PROC PARTITION	KFOLD=
OutputCasTables	See the section "OutputCasTables Table" on page 1275	OUTPUT	OUT=

Table 24.1 *continued*

Table Name	Description	Statements	Option
OVERFreq	Frequency table of input data size and sample size in different levels of the BY variable for the oversampling level of the BY variable (this table is produced for oversampling)	PROC PARTITION	EVENT =
PartIndMap	Table that maps each partition indicator to each target (this table is produced for segment-stratified sampling for each target variable)	TARGET	Default
SampleFreqMap	Table that maps each sample frequency to each target	TARGET, FREQ	Default
SRSFreq	Frequency table of input data size and sample size (this table is produced for simple random sampling)	PROC PARTITION	Default
STRAFreq	Frequency table of input data size and sample size in different stratification levels that are defined by BY variables (this table is produced for stratified sampling)	BY	Default

Examples: PARTITION Procedure

The examples in this chapter are available in the GitHub repository located at `https://github.com/sascommunities/doc-supplement-casstat`.

NOTE: Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections "Using CAS Sessions and CAS Engine Librefs" on page 7 and "Loading a SAS Data Set onto a CAS Server" on page 7 in Chapter 2, "Shared Concepts."

Example 24.1: Simple Random Sampling

(View the complete code for this example (partition_ex1.sas) in the example repository.)

This example demonstrates how to use PROC PARTITION to perform simple random sampling on the mylib.hmeq data table.

You can load the sampsio.hmeq data set into your CAS session by naming your CAS engine libref in the first statement of the following DATA step. This DATA step assumes that your CAS engine libref is named mylib, but you can substitute any appropriately defined CAS engine libref.

```
data mylib.hmeq;
   set sampsio.hmeq;
run;
```

The following statements perform the sampling:

```
proc partition data=mylib.hmeq samppct=10 seed=10 nthreads=1;
   output out=mylib.out2 copyvars=(job reason loan value delinq derog);
   display 'SRSFreq';
run;

proc print data=mylib.out2(obs=20);
run;
```

The SAMPPCT=10 option requests that 10% of the input data be sampled. The OUTPUT statement requests that the sampled data be stored in a table named mylib.out2, and the COPYVARS= option lists the variables to be copied from mylib.hmeq to mylib.out2. The DISPLAY statement requests that the SRSFreq ODS table be displayed.

Output 24.1.1 shows the number of observations in the mylib.hmeq data table and the number of samples in the mylib.out2 data table .

Output 24.1.1 Frequency Information Table

The PARTITION Procedure

Simple Random Sampling Frequency	
Number of Obs	Number of Samples
5960	596

Output 24.1.2 shows the sample data, which are stored in the mylib.out2 data table.

Output 24.1.2 Sample Data

Obs	JOB	REASON	LOAN	VALUE	DELINQ	DEROG
1	Other	HomeImp	2900	67996	0	3
2	Other	HomeImp	3000	33000	1	0
3	Office	HomeImp	3000	56400	0	0
4	Office	HomeImp	3600	63989	0	0
5	Office	HomeImp	4000	61777	0	0
6	Other	HomeImp	4000	16020	1	0
7	Mgr	HomeImp	4000	71865	2	.
8	ProfExe	DebtCon	4100	63797	4	0
9	Other	HomeImp	4300	79852	0	0
10			4400	60250	.	.
11	ProfExe	DebtCon	4500	87000	4	0
12	Self	HomeImp	5000	53448	0	0
13	Other	HomeImp	5000	50830	0	0
14		DebtCon	5000	77908	.	.
15	Other	HomeImp	5000	46281	0	0
16	ProfExe	HomeImp	5100	115341	0	0
17	ProfExe	HomeImp	5100	121382	0	0
18	Other	HomeImp	5200	77000	1	0
19	Office	HomeImp	5300	50673	.	.
20			5400	75254	.	.

Example 24.2: Stratified Sampling

(View the complete code for this example (partition_ex2.sas) in the example repository.)

This example demonstrates how to use PROC PARTITION to perform stratified sampling to partition the data; it uses the same data table as is used in Example 24.1.

You can load the sampsio.hmeq data set into your CAS session by naming your CAS engine libref in the first statement of the following DATA step. This DATA step assumes that your CAS engine libref is named mylib, but you can substitute any appropriately defined CAS engine libref.

```
data mylib.hmeq;
   set sampsio.hmeq;
run;
```

The following statements perform the partitioning:

```
proc partition data=mylib.hmeq samppct=10 samppct2=20 seed=10 partind nthreads=3;
   by BAD;
   output out=mylib.out3 copyvars=(job reason loan value delinq derog);
run;

proc print data=mylib.out3(obs=20);
run;
```

The SAMPPCT=10 option requests that 10% of the input data be included in the training partition, and the SAMPPCT2=20 option requests that 20% of the input data be included in the testing partition. The SEED=

option specifies 10 as the random seed to be used in the partitioning process. The PARTIND option requests that the output data table, mylib.out3, include an indicator that shows whether each observation is selected to a partition (1 for training or 2 for testing) or not (0). The OUTPUT statement requests that the sampled data be stored in a table named mylib.out3, and the COPYVARS= option lists the variables to be copied from mylib.hmeq to mylib.out3.

Output 24.2.1 shows the frequency information for each level of the BY variable BAD in the mylib.out3 data table.

Output 24.2.1 Frequency Information Table

The PARTITION Procedure

Stratified Sampling Frequency

Index	BAD	Number of Obs	Sample Size 1	Sample Size 2
0	0	4771	477	954
1	1	1189	119	238

Output 24.2.2 shows the first 20 output sample observations in mylib.out3; the _PartInd_ column shows which partition the observation is selected for (1 for training or 2 for testing) or none (0).

Output 24.2.2 Sample Output with Partition Indicator

Obs	JOB	REASON	LOAN	VALUE	DELINQ	DEROG	_PartInd_
1	Other	HomeImp	1100	39025	0	0	1
2	Other	HomeImp	1300	68400	2	0	0
3	Other	HomeImp	1500	16700	0	0	2
4			1500	.	.	.	2
5	Office	HomeImp	1700	112000	0	0	0
6	Other	HomeImp	1700	40320	0	0	0
7	Other	HomeImp	1800	57037	2	3	0
8	Other	HomeImp	1800	43034	0	0	0
9	Other	HomeImp	2000	46740	2	0	0
10	Sales	HomeImp	2000	62250	0	0	0
11			2000	.	.	.	0
12	Office	HomeImp	2000	29800	1	0	2
13	Other	HomeImp	2000	55000	0	0	0
14	Mgr		2000	87400	0	0	2
15	Other	HomeImp	2100	83850	1	0	2
16	Other	HomeImp	2200	34687	1	0	1
17	Mgr	HomeImp	2200	102600	6	2	0
18			2200	.	.	.	0
19	Other	HomeImp	2300	40150	0	0	0
20	Office	HomeImp	2300	120953	0	0	0

Example 24.3: Oversampling

(View the complete code for this example (partition_ex3.sas) in the example repository.)

This example demonstrates how to use PROC PARTITION to perform oversampling; it uses the same data table as in Example 24.1.

You can load the sampsio.hmeq data set into your CAS session by naming your CAS engine libref in the first statement of the following DATA step. This DATA step assumes that your CAS engine libref is named mylib, but you can substitute any appropriately defined CAS engine libref.

```
data mylib.hmeq;
   set sampsio.hmeq;
run;
```

The following statements perform oversampling:

```
proc partition data=mylib.hmeq samppctevt=90 eventprop=0.5
               event="1" seed=10 nthreads=1;
   by BAD;
   ods output OVERFREQ=outFreq;
   output out=mylib.out4 copyvars=(job loan value delinq derog)
          freqname=_Freq2_;
run;

proc print data=mylib.out4(obs=20);
run;
```

The EVENTPROP=0.5 option specifies that 50% of the sample are rare events. The SAMPPCTEVT=90 option requests that 90% of the rare events be sampled. The EVENT="1" option specifies that the second level of the variable BAD corresponds to a rare event. The OUTPUT statement requests that the sampled data be stored in a table named mylib.out4, specifies the variables to be transferred from the input data table, requests that the _Freq_ column be renamed to _Freq2_.

Output 24.3.1 shows the number of observations in the sample and in each level of the BY variable BAD in the mylib.out4 data table.

Output 24.3.1 Frequency Information Table

The PARTITION Procedure

Oversampling Frequency

Index	BAD	Number of Obs	Number of Samples
0	0	4771	1070
1	1	1189	1070

Output 24.3.2 shows the first 20 output sample observations in the mylib.out4 data table; the _Freq2_ column shows the ratio of target level's proportion in the population to its proportion in the sample.

Output 24.3.2 Output Data Table

Obs	JOB	LOAN	VALUE	DELINQ	DEROG	_Freq2_
1	Other	1100	39025	0	0	0.39899
2	Other	1300	68400	2	0	0.39899
3	Other	1500	16700	0	0	0.39899
4		1500	.	.	.	0.39899
5	Other	1700	40320	0	0	0.39899
6	Other	1800	57037	2	3	0.39899
7	Other	1800	43034	0	0	0.39899
8	Other	2000	46740	2	0	0.39899
9	Sales	2000	62250	0	0	0.39899
10		2000	.	.	.	0.39899
11	Office	2000	29800	1	0	0.39899
12	Other	2000	55000	0	0	0.39899
13	Mgr	2000	87400	0	0	1.60101
14	Other	2100	83850	1	0	0.39899
15	Other	2200	34687	1	0	0.39899
16	Mgr	2200	102600	6	2	0.39899
17		2200	.	.	.	0.39899
18	Other	2300	40150	0	0	0.39899
19	Other	2300	46200	1	0	0.39899
20	Mgr	2400	40800	0	0	0.39899

Example 24.4: *k*-Fold Partitioning

(View the complete code for this example (partition_ex4.sas) in the example repository.)

This example demonstrates how to use PROC PARTITION to perform stratified *k*-fold partitioning to partition the data; it uses the same data table as is used in Example 24.1.

You can load the sampsio.hmeq data set into your CAS session by naming your CAS engine libref in the first statement of the following DATA step. This DATA step assumes that your CAS engine libref is named mylib, but you can substitute any appropriately defined CAS engine libref.

```
data mylib.hmeq;
   set sampsio.hmeq;
run;
```

The following statements perform the partitioning:

```
proc partition data=mylib.hmeq kfold=10 seed=10 nthreads=3;
   by BAD;
   output out=mylib.out5 copyvars=(job reason loan value delinq derog);
run;

proc print data=mylib.out5(obs=20);
run;
```

The KFOLD=10 option requests that the input data be partitioned to 10 equal-size folds within each level of the BY variable BAD. The SEED= option specifies 10 as the random seed to be used in the partitioning

process. The OUTPUT statement requests that the sampled data be stored in a table named mylib.out5, and the COPYVARS= option lists the variables to be copied from mylib.hmeq to mylib.out5. The output data table, mylib.out5, includes an indicator that shows which fold each observation is selected to.

Output 24.4.1 shows the frequency information for each level of the BY variable BAD in the mylib.out5 data table.

Output 24.4.1 Frequency Information Table

The PARTITION Procedure

k-Fold Partition Frequency

Index	BAD	Number of Obs	Fold Size 1	Fold Size 2	Fold Size 3	Fold Size 4	Fold Size 5	Fold Size 6	Fold Size 7	Fold Size 8	Fold Size 9	Fold Size 10
0	0	4771	478	477	477	477	477	477	477	477	477	477
1	1	1189	119	118	119	119	119	119	119	119	119	119

Output 24.4.2 shows the first 20 output sample observations in mylib.out5; the _Fold_ column shows which fold the observation is selected to.

Output 24.4.2 *k*-Fold Partitioning Output Data Table

Obs	JOB	REASON	LOAN	VALUE	DELINQ	DEROG	_Fold_
1	Other	HomeImp	1100	39025	0	0	1
2	Other	HomeImp	1300	68400	2	0	5
3	Other	HomeImp	1500	16700	0	0	3
4			1500	.	.	.	3
5	Office	HomeImp	1700	112000	0	0	10
6	Other	HomeImp	1700	40320	0	0	7
7	Other	HomeImp	1800	57037	2	3	9
8	Other	HomeImp	1800	43034	0	0	6
9	Other	HomeImp	2000	46740	2	0	5
10	Sales	HomeImp	2000	62250	0	0	4
11			2000	.	.	.	6
12	Office	HomeImp	2000	29800	1	0	2
13	Other	HomeImp	2000	55000	0	0	4
14	Mgr		2000	87400	0	0	2
15	Other	HomeImp	2100	83850	1	0	3
16	Other	HomeImp	2200	34687	1	0	1
17	Mgr	HomeImp	2200	102600	6	2	4
18			2200	.	.	.	5
19	Other	HomeImp	2300	40150	0	0	4
20	Office	HomeImp	2300	120953	0	0	4

Example 24.5: Segment-Stratified Sampling for Each Target

(View the complete code for this example (partition_ex5.sas) in the example repository.)

This example demonstrates how to use PROC PARTITION to perform segment-stratified partitioning for each target; it uses the same data table as is used in Example 24.1.

You can load the sampsio.hmeq data set into your CAS session by naming your CAS engine libref in the first statement of the following DATA step. This DATA step assumes that your CAS engine libref is named mylib, but you can substitute any appropriately defined CAS engine libref.

```
data mylib.hmeq;
   set sampsio.hmeq;
run;
```

The following statements perform the partitioning:

```
proc partition data=mylib.hmeq samppct=10 samppct2=20 seed=10 nthreads=3;
   by bad;
   target job reason;
   output out=mylib.out3 copyvars=(job reason loan value delinq derog);
run;

proc print data=mylib.out3(obs=20);
run;
```

The SAMPPCT=10 option requests that 10% of the input data be included in the training partition, and the SAMPPCT2=20 option requests that 20% of the input data be included in the testing partition. The SEED= option specifies 10 as the random seed to be used in the partitioning process. The BY statement requests that the BY variable bad be used as the segmentation variable. The TARGET statement requests that stratified sampling be performed for each target variable job and reason in each segment level. The OUTPUT statement requests that the sampled data be stored in a table named mylib.out3, and the COPYVARS= option lists the variables to be copied from mylib.hmeq to mylib.out3. The output data table, mylib.out3, includes two indicators that show whether each observation is selected to a partition (1 for training, 2 for testing, or 0 for not selected) for each target variable in each segment level in the BY variable bad.

Output 24.5.1 shows the frequency information for each level combination of the BY variable bad and each target variable in the mylib.out3 data table.

Output 24.5.1 Frequency Information Table

The PARTITION Procedure

Stratified Sampling Frequency					
Target Name	Target Level	BAD	Number of Obs	Sample Size 1	Sample Size 2
JOB		0	256	26	51
JOB	Mgr	0	588	59	117
JOB	Office	0	823	82	165
JOB	Other	0	1834	183	367
JOB	ProfExe	0	1064	106	213
JOB	Sales	0	71	7	14
JOB	Self	0	135	14	27
JOB		1	23	2	5
JOB	Mgr	1	179	18	36
JOB	Office	1	125	13	25
JOB	Other	1	554	55	111
JOB	ProfExe	1	212	21	43
JOB	Sales	1	38	4	7
JOB	Self	1	58	6	11
REASON		0	204	20	41
REASON	DebtCon	0	3183	318	637
REASON	HomeImp	0	1384	138	277
REASON		1	48	5	9
REASON	DebtCon	1	745	75	149
REASON	HomeImp	1	396	40	79

Output 24.5.2 shows the map of each partition indicator to each target variable.

Output 24.5.2 Partition Indicator Map

Partition Indicator Map	
Partition Indicator Name	Target Name
PartInd1	JOB
PartInd2	REASON

Output 24.5.3 shows the first 20 output sample observations in mylib.out3. The _PartInd1_ column shows which partition the observation is selected for (1 for training, 2 for testing, or 0 for not selected) for the stratification that is defined by the level combinations of the target variable job and the BY variable bad, and the _PartInd2_ column shows which partition the observation is selected for (1 for training, 2 for testing, or 0 for not selected) for the stratification that is defined by the level combinations of the target variable reason and the BY variable bad.

Output 24.5.3 Sample Output with Partition Indicator

Obs	JOB	REASON	LOAN	VALUE	DELINQ	DEROG	_PartInd1_	_PartInd2_
1	Other	HomeImp	1100	39025	0	0	2	0
2	Other	HomeImp	1300	68400	2	0	0	0
3	Other	HomeImp	1500	16700	0	0	0	1
4			1500	.	.	.	0	0
5	Office	HomeImp	1700	112000	0	0	0	0
6	Other	HomeImp	1700	40320	0	0	0	0
7	Other	HomeImp	1800	57037	2	3	0	0
8	Other	HomeImp	1800	43034	0	0	0	0
9	Other	HomeImp	2000	46740	2	0	0	0
10	Sales	HomeImp	2000	62250	0	0	0	2
11			2000	.	.	.	0	0
12	Office	HomeImp	2000	29800	1	0	0	1
13	Other	HomeImp	2000	55000	0	0	1	0
14	Mgr		2000	87400	0	0	1	2
15	Other	HomeImp	2100	83850	1	0	0	0
16	Other	HomeImp	2200	34687	1	0	0	0
17	Mgr	HomeImp	2200	102600	6	2	0	0
18			2200	.	.	.	0	2
19	Other	HomeImp	2300	40150	0	0	1	0
20	Office	HomeImp	2300	120953	0	0	0	0

Example 24.6: Segment-Stratified Sampling for Each Target Using the FREQ Statement

(View the complete code for this example (partition_ex6.sas) in the example repository.)

This example demonstrates how to use PROC PARTITION to perform segment-stratified partitioning for each target with the FREQ statement. It uses the same data table as in Example 24.1.

You can load the sampsio.hmeq data set into your CAS session by naming your CAS engine libref in the first statement of the following DATA step. This DATA step assumes that your CAS engine libref is named mylib, but you can substitute any appropriately defined CAS engine libref.

```
data mylib.hmeq;
   set sampsio.hmeq;
   id = _N_;
   freqVar = 10;
run;
```

Note that the frequency variable freqVar is set to 10 for all the observations in the hmeq data set in the CAS engine libref mylib.

The following statements perform the partitioning:

```
proc partition data=mylib.hmeq samppct=10 samppct2=20 seed=10;
   by bad;
   target job reason;
   freq freqVar;
   output out=mylib.out6 copyvars=(job reason loan value delinq derog);
run;

proc print data=mylib.out6(obs=20);
run;
```

The SAMPPCT=10 option requests that 10% of the input data be included in the training partition, and the SAMPPCT2=20 option requests that 20% of the input data be included in the testing partition. The SEED= option specifies 10 as the random seed to be used in the partitioning process. The BY statement requests that the BY variable bad be used as the segmentation variable. The TARGET statement requests that stratified sampling be performed for each target variable job and reason in each segment level. The FREQ statement requests that the variable freqVar be used as the frequency variable. In this example, the procedure treats each observation as if it appears 10 times. The OUTPUT statement requests that the sampled data be stored in a table named mylib.out6, and the COPYVARS= option lists the variables to be copied from mylib.hmeq to mylib.out6. The output data table, mylib.out6, also includes one partition indicator variable that shows whether each observation is selected for a partition (1 for training, 2 for testing, or 0 for not selected), as well as two sample frequency variables (one for each target variable in each segment level in the BY variable bad).

Output 24.6.1 shows the frequency information for each level combination of the BY variable bad and each target variable in the mylib.out6 data table.

Output 24.6.1 Frequency Information Table

The PARTITION Procedure

Stratified Sampling Frequency

Target Name	Target Level	BAD	Number of Obs	Sample Size 1	Sample Size 2
JOB		0	2560	256	512
JOB	Mgr	0	5880	588	1176
JOB	Office	0	8230	823	1646
JOB	Other	0	18340	1834	3668
JOB	ProfExe	0	10640	1064	2128
JOB	Sales	0	710	71	142
JOB	Self	0	1350	135	270
JOB		1	230	23	46
JOB	Mgr	1	1790	179	358
JOB	Office	1	1250	125	250
JOB	Other	1	5540	554	1108
JOB	ProfExe	1	2120	212	424
JOB	Sales	1	380	38	76
JOB	Self	1	580	58	116
REASON		0	2040	204	408
REASON	DebtCon	0	31830	3183	6366
REASON	HomeImp	0	13840	1384	2768
REASON		1	480	48	96
REASON	DebtCon	1	7450	745	1490
REASON	HomeImp	1	3960	396	792

Output 24.6.2 shows the map of each sample frequency to each target variable.

Output 24.6.2 Sample Frequency Map

Sample Frequency Map

Sample Frequency Name	Target Name
SampleFreq1	JOB
SampleFreq2	REASON

Output 24.6.3 shows the first 20 output sample observations in mylib.out6. Each observation is duplicated by three rows in the output, with the _PartInd_ column showing which partition each observation is selected for (1 for training, 2 for testing, or 0 for not selected). The _SampleFreq1_ column shows the sample frequency for each partition for the target variable job. The _SampleFreq2_ column shows the sample frequency for each partition for the target variable reason. Note that the rows for which all the _SampleFreq_ columns have a value of 0 are not included in the output.

Output 24.6.3 Sample Output with Partition Indicator

Obs	JOB	REASON	LOAN	VALUE	DELINQ	DEROG	_PartInd_	_SampleFreq1_	_SampleFreq2_
1	Other	HomeImp	1100	39025	0	0	0	8	7
2	Other	HomeImp	1100	39025	0	0	1	1	2
3	Other	HomeImp	1100	39025	0	0	2	1	1
4	Other	HomeImp	1300	68400	2	0	0	8	8
5	Other	HomeImp	1300	68400	2	0	1	1	1
6	Other	HomeImp	1300	68400	2	0	2	1	1
7	Other	HomeImp	1500	16700	0	0	0	9	6
8	Other	HomeImp	1500	16700	0	0	1	0	1
9	Other	HomeImp	1500	16700	0	0	2	1	3
10			1500	.	.	.	0	9	6
11			1500	.	.	.	1	1	1
12			1500	.	.	.	2	0	3
13	Office	HomeImp	1700	112000	0	0	0	8	8
14	Office	HomeImp	1700	112000	0	0	1	0	1
15	Office	HomeImp	1700	112000	0	0	2	2	1
16	Other	HomeImp	1700	40320	0	0	0	9	7
17	Other	HomeImp	1700	40320	0	0	1	0	1
18	Other	HomeImp	1700	40320	0	0	2	1	2
19	Other	HomeImp	1800	57037	2	3	0	8	8
20	Other	HomeImp	1800	57037	2	3	1	2	1

Chapter 25
PCA Procedure

Contents

Overview: PCA Procedure **1290**
PROC PCA Features 1290
PROC PCA Compared with Other SAS Procedures 1291
PROC PCA Compared with the HPPRINCOMP Procedure 1291
PROC PCA Compared with the PRINCOMP Procedure 1292
Using CAS Sessions and CAS Engine Librefs 1293
Getting Started: PCA Procedure **1294**
Syntax: PCA Procedure **1298**
PROC PCA Statement 1298
BY Statement 1304
CODE Statement 1305
DISPLAY Statement 1305
DISPLAYOUT Statement 1306
FREQ Statement 1307
OUTPUT Statement 1307
PARTIAL Statement 1309
STORE Statement 1309
VAR Statement 1310
WEIGHT Statement 1310
Details: PCA Procedure **1310**
Computing Principal Components 1310
Eigenvalue Decomposition 1311
NIPALS 1312
ITERGS 1312
RANDOM 1312
Missing Values 1313
OUTSTAT= Data Table 1313
Displayed Output 1315
Model Information 1315
Number of Variables 1315
Number of Observations 1315
Simple Statistics 1315
Centering and Scaling Information 1315
Explained Variation of Variables 1315
Correlation Matrix 1316
Regression Statistics 1316

Regression Coefficients 1316
Partial Correlation Matrix 1316
Total Variance 1316
Eigenvalues 1316
Eigenvectors 1316
Loadings 1316
Timing Information 1317
OutputCasTables Table 1317
ODS Table Names 1317
ODS Graphics 1318
Examples: PCA Procedure **1318**
Example 25.1: Analyzing Mean Temperatures of US Cities 1318
Example 25.2: Extracting Principal Components with NIPALS 1320
Example 25.3: Approximating Principal Components with the RANDOM Method . . 1325
References **1328**

Overview: PCA Procedure

The PCA procedure performs principal component analysis in SAS Viya.

Principal component analysis is a multivariate technique for examining relationships among several quantitative variables. It provides an optimal way to reduce dimensionality by projecting the data onto a lower-dimensional orthogonal subspace that explains as much variation as possible in those variables. The choice between using factor analysis and using principal component analysis depends in part on your research objectives. You should use the PCA procedure if you are interested in summarizing data and detecting linear relationships. You can use principal component analysis to reduce the number of variables in regression, clustering, and so on.

PROC PCA Features

The PCA procedure does the following:

- provides a PARTIAL statement for analyzing a partial correlation or covariance matrix
- provides a FREQ statement for grouped analysis
- provides a WEIGHT statement for weighted analysis
- provides a CODE statement for scoring new data
- produces an output data table that contains principal component scores and other observationwise statistics
- produces an output data table that contains means, standard deviations, number of observations, correlations or covariances, eigenvalues, and eigenvectors

- produces a scree plot of eigenvalues and proportion variance explained and a pattern profile plot

The PCA procedure implements the following methods:

- eigenvalue decomposition, which uses the correlation or covariance of the data matrix and calculates all the principal components simultaneously
- the nonlinear iterative partial least squares (NIPALS) method, which uses the data matrix and extracts the principal components successively
- the iterative method based on the Gram-Schmidt orthogonalization (ITERGS) of Andrecut (2009), which uses the data matrix and extracts the principal components successively. The algorithm applies reorthogonalization correction to both the scores and the loadings at each iteration step.
- the iterative method based on random projection (RANDOM) of Halko, Martinsson, and Tropp (2011), which uses the data matrix and extracts all requested principal components simultaneously. Because of the nature of the algorithm, the principal components are an approximation of the actual principal components.

Because the PCA procedure runs on CAS, it also does the following:

- enables you to run on a cluster of machines that distribute the data and the computations
- enables you to run in single-machine mode on CAS
- exploits all the available cores and concurrent threads. For information about how PROC PCA uses threads, see the section "Multithreading" on page 102 in Chapter 2, "Shared Concepts."

PROC PCA Compared with Other SAS Procedures

The PCA procedure provides functionality to perform principal component analysis that is comparable to that of the HPPRINCOMP and PRINCOMP procedures in SAS/STAT software.

PROC PCA Compared with the HPPRINCOMP Procedure

The functionality of the PCA procedure closely resembles that of the HPPRINCOMP procedure, which is a high-performance procedure. The PCA procedure is the next generation of the HPPRINCOMP procedure, and it was developed specifically for SAS Viya. Both procedures are designed to run on a cluster of machines that distribute the data and the computations.

The PCA procedure and the HPPRINCOMP procedure have the similarities and differences shown in Table 25.1.

Table 25.1 Comparison of PROC PCA and PROC HPPRINCOMP

Feature	PROC PCA	PROC HPPRINCOMP
Supports various methods to extract principal components by using the METHOD= option in the PROC statement	Yes	Yes
Supports BY statement	Yes	Yes
Supports DISPLAY and DISPLAYOUT statements	Yes	No
Supports CODE and OUTPUT statements	Yes	Yes
Supports ID statement	No	Yes
Supports PLOTS= option	Yes, with limited number of options	No
Includes input variables in output data table	Not by default, in order to avoid data duplication for large data tables. To include input variables, specify them in the COPYVARS= option in the OUTPUT statement.	Not by default, in order to avoid data duplication for large data sets. To include input variables, specify them in the ID statement.
Threading	Specifically designed for CAS; executes on multiple threads	Primarily designed for a distributed environment; executes on multiple threads

PROC PCA Compared with the PRINCOMP Procedure

The PCA procedure and the PRINCOMP procedure have the similarities and differences shown in Table 25.2.

Table 25.2 Comparison of PROC PCA and PROC PRINCOMP

Feature	PROC PCA	PROC PRINCOMP
Supports various methods to extract principal components by using the METHOD= option in the PROC statement	Yes	No, supports only eigenvalue decomposition
Supports BY statement	Yes	Yes
Supports DISPLAY and DISPLAYOUT statements	Yes	No
Supports CODE and OUTPUT statements	Yes	No
Supports ID statement	No	Yes
Supports PLOTS= option	Limited number of options	Full set of options

Table 25.2 *continued*

Feature	PROC PCA	PROC PRINCOMP
Accepts input data	Only CAS input data tables	Ordinary SAS data sets and certain types of special SAS data sets
Includes input variables in output data table	Not by default, in order to avoid data duplication for large data tables. To include input variables, specify them in the COPYVARS= option in the OUTPUT statement.	Yes
Threading	Specifically designed for CAS; executes on multiple threads	Executes on a single thread

Using CAS Sessions and CAS Engine Librefs

SAS Cloud Analytic Services (CAS) is the analytic server and associated cloud services in SAS Viya. This section describes how to create a CAS session and set up a CAS engine libref that you can use to connect to the CAS session. It assumes that you have a CAS server already available; contact your system administrator if you need help starting and terminating a server. This CAS server is identified by specifying the host on which it runs and the port on which it listens for communications. To simplify your interactions with this CAS server, the host information and port information for the server are stored as SAS option values that are retrieved automatically whenever this CAS server needs to be accessed. You can examine the host and port values for the server at your site by using the following statements:

```
proc options option=(CASHOST CASPORT);
run;
```

In addition to starting a CAS server, your system administrator might also have created a CAS session and a CAS engine libref for your use. You can define your own sessions and CAS engine librefs that connect to the CAS server as shown in the following statements:

```
cas mysess;
libname mylib cas sessref=mysess;
```

The CAS statement creates the CAS session named mysess, and the LIBNAME statement creates the mylib CAS engine libref that you use to connect to this session. It is not necessary to explicitly name the CASHOST and CASPORT of the CAS server in the CAS statement, because these values are retrieved from the corresponding SAS option values.

If you have created the mysess session, you can terminate it by using the TERMINATE option in the CAS statement as follows:

```
cas mysess terminate;
```

For more information about the CAS and LIBNAME statements, see the section "Introduction to Shared Concepts" on page 6 in Chapter 2, "Shared Concepts."

Getting Started: PCA Procedure

(View the complete code for this example at https://github.com/sascommunities/doc-supplement-casstat/tree/main/Examples/pcags.sas.)

NOTE: Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections "Using CAS Sessions and CAS Engine Librefs" on page 7 and "Loading a SAS Data Set onto a CAS Server" on page 7 in Chapter 2, "Shared Concepts."

The following DATA step creates the Crime data table, which provides crime rates per 100,000 people in seven categories for each of the 50 US states in 1977, in your CAS session:

```
data mylib.Crime;
   title 'Crime Rates per 100,000 Population by State';
   input State $1-15 Murder Rape Robbery Assault
         Burglary Larceny Auto_Theft;
   datalines;
Alabama        14.2 25.2  96.8 278.3 1135.5 1881.9 280.7
Alaska         10.8 51.6  96.8 284.0 1331.7 3369.8 753.3
Arizona         9.5 34.2 138.2 312.3 2346.1 4467.4 439.5
Arkansas        8.8 27.6  83.2 203.4  972.6 1862.1 183.4
California     11.5 49.4 287.0 358.0 2139.4 3499.8 663.5
Colorado        6.3 42.0 170.7 292.9 1935.2 3903.2 477.1
Connecticut     4.2 16.8 129.5 131.8 1346.0 2620.7 593.2
Delaware        6.0 24.9 157.0 194.2 1682.6 3678.4 467.0
Florida        10.2 39.6 187.9 449.1 1859.9 3840.5 351.4
Georgia        11.7 31.1 140.5 256.5 1351.1 2170.2 297.9
Hawaii          7.2 25.5 128.0  64.1 1911.5 3920.4 489.4
Idaho           5.5 19.4  39.6 172.5 1050.8 2599.6 237.6
Illinois        9.9 21.8 211.3 209.0 1085.0 2828.5 528.6
Indiana         7.4 26.5 123.2 153.5 1086.2 2498.7 377.4
Iowa            2.3 10.6  41.2  89.8  812.5 2685.1 219.9
Kansas          6.6 22.0 100.7 180.5 1270.4 2739.3 244.3
Kentucky       10.1 19.1  81.1 123.3  872.2 1662.1 245.4
Louisiana      15.5 30.9 142.9 335.5 1165.5 2469.9 337.7
Maine           2.4 13.5  38.7 170.0 1253.1 2350.7 246.9
Maryland        8.0 34.8 292.1 358.9 1400.0 3177.7 428.5
Massachusetts   3.1 20.8 169.1 231.6 1532.2 2311.3 1140.1
Michigan        9.3 38.9 261.9 274.6 1522.7 3159.0 545.5
Minnesota       2.7 19.5  85.9  85.8 1134.7 2559.3 343.1
Mississippi    14.3 19.6  65.7 189.1  915.6 1239.9 144.4
Missouri        9.6 28.3 189.0 233.5 1318.3 2424.2 378.4
Montana         5.4 16.7  39.2 156.8  804.9 2773.2 309.2
Nebraska        3.9 18.1  64.7 112.7  760.0 2316.1 249.1
```

```
Nevada          15.8 49.1 323.1 355.0 2453.1 4212.6 559.2
New Hampshire    3.2 10.7  23.2  76.0 1041.7 2343.9 293.4
New Jersey       5.6 21.0 180.4 185.1 1435.8 2774.5 511.5
New Mexico       8.8 39.1 109.6 343.4 1418.7 3008.6 259.5
New York        10.7 29.4 472.6 319.1 1728.0 2782.0 745.8
North Carolina  10.6 17.0  61.3 318.3 1154.1 2037.8 192.1
North Dakota     0.9  9.0  13.3  43.8  446.1 1843.0 144.7
Ohio             7.8 27.3 190.5 181.1 1216.0 2696.8 400.4
Oklahoma         8.6 29.2  73.8 205.0 1288.2 2228.1 326.8
Oregon           4.9 39.9 124.1 286.9 1636.4 3506.1 388.9
Pennsylvania     5.6 19.0 130.3 128.0  877.5 1624.1 333.2
Rhode Island     3.6 10.5  86.5 201.0 1489.5 2844.1 791.4
South Carolina  11.9 33.0 105.9 485.3 1613.6 2342.4 245.1
South Dakota     2.0 13.5  17.9 155.7  570.5 1704.4 147.5
Tennessee       10.1 29.7 145.8 203.9 1259.7 1776.5 314.0
Texas           13.3 33.8 152.4 208.2 1603.1 2988.7 397.6
Utah             3.5 20.3  68.8 147.3 1171.6 3004.6 334.5
Vermont          1.4 15.9  30.8 101.2 1348.2 2201.0 265.2
Virginia         9.0 23.3  92.1 165.7  986.2 2521.2 226.7
Washington       4.3 39.6 106.2 224.8 1605.6 3386.9 360.3
West Virginia    6.0 13.2  42.2    .  597.4 1341.7 163.3
Wisconsin        2.8 12.9  52.2  63.7  846.9 2614.2 220.7
Wyoming           .  21.9  39.7 173.9  811.6 2772.2 282.0
;
```

These statements assume that your CAS engine libref is named mylib, but you can substitute any appropriately defined CAS engine libref.

The following statements invoke the PCA procedure, which requests a principal component analysis of the data and produces Figure 25.1 through Figure 25.4:

```
proc pca data=mylib.Crime;
run;
```

Figure 25.1 displays the "Model Information," "Number of Variables," "Number of Observations," and "Simple Statistics" tables.

The "Model Information" table identifies the data source and shows that the principal component extraction method is eigenvalue decomposition, which is the default.

The "Number of Variables" table indicates that there are seven variables to be analyzed and seven principal components to be computed. By default, if you omit the VAR statement, all numeric variables that are not listed in other statements are used in the analysis.

The "Number of Observations" table shows that of the 50 observations in the input data, only 48 observations are used in the analysis, because some observations have incomplete data.

The "Simple Statistics" table displays the mean and standard deviation of the analysis variables.

Figure 25.1 Model Information and Simple Statistics

Crime Rates per 100,000 Population by State

The PCA Procedure

Model Information	
Data Source	CRIME
Component Extraction Method	Eigenvalue Decomposition

Number of Variables	7
Number of Principal Components	7

Number of Observations Read	50
Number of Observations Used	48

Simple Statistics

Variable	Mean	Standard Deviation
Murder	7.51667	3.93059
Rape	26.07500	10.81304
Robbery	127.55625	88.49374
Assault	214.58750	100.64360
Burglary	1316.37917	423.31261
Larceny	2696.88542	714.75023
Auto_Theft	383.97917	194.37033

Figure 25.2 displays the "Correlation Matrix" table. By default, the PROC PCA statement requests that principal components be computed from the correlation matrix, so the total variance is equal to the number of variables, 7.

Figure 25.2 Correlation Matrix Table

Correlation Matrix

Variable	Murder	Rape	Robbery	Assault	Burglary	Larceny	Auto_Theft
Murder	1.0000	0.6000	0.4768	0.6485	0.3778	0.0925	0.0555
Rape	0.6000	1.0000	0.5817	0.7316	0.7038	0.6009	0.3282
Robbery	0.4768	0.5817	1.0000	0.5452	0.6200	0.4371	0.5787
Assault	0.6485	0.7316	0.5452	1.0000	0.6082	0.3791	0.2520
Burglary	0.3778	0.7038	0.6200	0.6082	1.0000	0.7932	0.5390
Larceny	0.0925	0.6009	0.4371	0.3791	0.7932	1.0000	0.4246
Auto_Theft	0.0555	0.3282	0.5787	0.2520	0.5390	0.4246	1.0000

Figure 25.3 displays the "Eigenvalues" table. The first principal component accounts for about 57.8% of the total variance, the second principal component accounts for about 18.1%, and the third principal component accounts for about 10.7%. Note that the eigenvalues sum to the total variance.

The eigenvalues indicate that two or three components provide a good summary of the data: two components account for 76% of the total variance, and three components account for 87%. Subsequent components account for less than 5% each.

Figure 25.3 Eigenvalues Table

Eigenvalues of the Correlation Matrix				
	Eigenvalue	Difference	Proportion	Cumulative
1	4.045824	2.781795	0.5780	0.5780
2	1.264030	0.516529	0.1806	0.7586
3	0.747500	0.421175	0.1068	0.8653
4	0.326325	0.061119	0.0466	0.9120
5	0.265207	0.036843	0.0379	0.9498
6	0.228364	0.105613	0.0326	0.9825
7	0.122750		0.0175	1.0000

Figure 25.4 displays the "Eigenvectors" table. From the eigenvectors matrix, you can represent the first principal component, Prin1, as a linear combination of the original variables:

$$\begin{aligned}\text{Prin1} = &-0.30289 \times (\text{Murder}) \\ &- 0.43410 \times (\text{Rape}) \\ &- 0.39705 \times (\text{Robbery}) \\ &\vdots \\ &- 0.28834 \times (\text{Auto\_Theft})\end{aligned}$$

Similarly, the second principal component, Prin2, is

$$\begin{aligned}\text{Prin2} = &\ 0.61893 \times (\text{Murder}) \\ &+ 0.17053 \times (\text{Rape}) \\ &- 0.04713 \times (\text{Robbery}) \\ &\vdots \\ &- 0.50400 \times (\text{Auto\_Theft})\end{aligned}$$

where the variables are standardized.

Figure 25.4 Eigenvectors Table

Eigenvectors							
Variable	**Prin1**	**Prin2**	**Prin3**	**Prin4**	**Prin5**	**Prin6**	**Prin7**
Murder	-0.30289	0.61893	0.17353	-0.23308	-0.54896	0.26371	-0.26428
Rape	-0.43410	0.17053	-0.23539	0.06540	-0.18075	-0.78232	0.27946
Robbery	-0.39705	-0.04713	0.49208	-0.57470	0.50808	-0.09452	0.02497
Assault	-0.39622	0.35142	-0.05343	0.61743	0.51525	0.17395	-0.19921
Burglary	-0.44164	-0.20861	-0.22454	-0.02750	-0.11273	0.52340	0.65085
Larceny	-0.35634	-0.40570	-0.53681	-0.23231	-0.02172	0.04085	-0.60346
Auto_Theft	-0.28834	-0.50400	0.57524	0.41853	-0.35939	-0.06024	-0.15487

The first component is a measure of the overall crime rate, because the first eigenvector shows approximately equal loadings on all variables. The second eigenvector has high negative loadings on the variables Auto_Theft and Larceny and high positive loadings on the variables Murder and Assault. There is also a small negative loading on the variable Burglary and a small positive loading on the variable Rape. This component seems to measure the preponderance of property crime compared to violent crime. The interpretation of the third component is not obvious.

Syntax: PCA Procedure

The following statements are available in the PCA procedure:

```
PROC PCA < options > ;
    BY variables ;
    CODE < options > ;
    DISPLAY < table-list > < / options > ;
    DISPLAYOUT table-spec-list < / options > ;
    FREQ variable ;
    OUTPUT OUT=libref.data-table
           < COPYVARS=(variables) >
           < keyword < =prefix > >. . . < keyword < =prefix > > ;
    PARTIAL variables ;
    STORE < OUT= >libref.data-table ;
    VAR variables ;
    WEIGHT variable ;
```

The rest of this section provides detailed syntax information about each of the preceding statements, beginning with the PROC PCA statement. The remaining statements are described in alphabetical order.

PROC PCA Statement

```
PROC PCA < options > ;
```

The PROC PCA statement invokes the PCA procedure. Optionally, it also identifies the input and output data sets, specifies the analyses to be performed, and controls displayed output. Table 25.3 summarizes the options available in the PROC PCA statement.

Table 25.3 PROC PCA Statement Options

Option	Description
Specify Data Sets	
DATA=	Specifies the input data table
OUTSTAT=	Specifies the output data table that contains various statistics
Specify Details of Analysis	
COV	Computes the principal components from the covariance matrix

Table 25.3 *continued*

Option	Description
GPU	Uses graphics processing units (GPUs) to perform model computation
METHOD=	Specifies the principal component extraction method to be used
N=	Specifies the number of principal components to be computed
NOINT	Omits the intercept from the model
PARPREFIX=	Specifies a prefix for naming the residual variables
PREFIX=	Specifies a prefix for naming the principal components
SINGULAR=	Specifies the singularity criterion
STD	Standardizes the principal component scores
VARDEF=	Specifies the divisor to use for calculating variances and standard deviations
Specify ODS Graphics Details	
PLOTS=	Specifies options that control the details of the plots

The following list provides details about these *options*.

COVARIANCE

COV

computes the principal components from the covariance matrix. If you omit this option, the correlation matrix is analyzed. The COV option causes variables that have large variances to be more strongly associated with components that have large eigenvalues, and it causes variables that have small variances to be more strongly associated with components that have small eigenvalues. You should not specify this option unless the units in which the variables are measured are comparable or the variables are standardized in some way.

DATA=*libref.data-table*

names the input data table for PROC PCA to use. The default is the most recently created data table. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to the data, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 1293.

data-table specifies the name of the input data table.

GPU

enables PROC PCA to use graphics processing units (GPUs) to perform model computation on GPU-equipped machines (one GPU is used on each machine). Currently the eigenvalue decomposition (EIG) method, the nonlinear iterative partial least squares (NIPALS) method, and the iterative method based on random projection (RANDOM) support this option. If you specify any other principal component extraction method or specify the BY statement or PARTIAL statement, the GPU option is ignored. If no GPU is available, the CPU is used to perform model computation.

PROC PCA uses CUDA 11.8, and the NVIDIA V100 and P100 GPUs are supported.[1]

METHOD=EIG | ITERGS< (*iter-options*) > | NIPALS< (*nipals-options*) > | RANDOM< (*rand-options*) >
specifies the principal component extraction method to be used. You can specify the following values:

EIG
requests eigenvalue decomposition.

ITERGS< (*iter-options*) >
requests the iterative method based on Gram-Schmidt orthogonalization (ITERGS) of Andrecut (2009). You can specify the following *iter-options*.

EPSILON=*n*
specifies the convergence criterion for the iterative method. By default, EPSILON=1E–12.

MAXITER=*n*
specifies the maximum number of iterations for the iterative method. By default, MAXITER=500.

NOCENTER
suppresses centering of the numeric variables to be analyzed. This option is useful if the analysis variables are already centered and scaled.

NOSCALE
suppresses scaling of the numeric variables to be analyzed. This option is useful if the analysis variables are already centered and scaled.

NIPALS< (*nipals-options*) >
requests the nonlinear iterative partial least squares (NIPALS) method. You can specify the following *nipals-options*.

EPSILON=*n*
specifies the convergence criterion for the iterative method. By default, EPSILON=1E–12.

MAXITER=*n*
specifies the maximum number of iterations for the iterative method. By default, MAXITER=500.

NOCENTER
suppresses centering of the numeric variables to be analyzed. This option is useful if the analysis variables are already centered and scaled.

NOSCALE
suppresses scaling of the numeric variables to be analyzed. This option is useful if the analysis variables are already centered and scaled.

[1] The version information and corresponding GPU architectures depend on your SAS Viya version. Make sure that you are consulting the correct version of the documentation for your specific SAS Viya version.

RANDOM< (*rand-options*) >
requests the iterative method based on random projection (RANDOM) of Halko, Martinsson, and Tropp (2011). You can specify the following *rand-options*.

NITER=*n*
specifies the number of iterations. By default, NITER=1.

NOCENTER
suppresses centering of the numeric variables to be analyzed. This option is useful if the analysis variables are already centered and scaled.

NOSCALE
suppresses scaling of the numeric variables to be analyzed. This option is useful if the analysis variables are already centered and scaled.

SEED=*number*
specifies an integer as the seed value for pseudorandom number generation. PROC PCA uses random values to initialize the Gaussian random projection matrix. If you do not specify this option or if *number* is less than or equal to 0, the seed is generated by reading the time of day from the computer's clock. By default, SEED=0.

By default, METHOD=EIG. If you specify METHOD=ITERGS, METHOD=NIPALS, or METHOD=RANDOM, the following options in the PROC PCA statement are not supported: COV, NOINT, OUTSTAT=, PARPREFIX=, SINGULAR=, and STD.

N=*number*
specifies the number of principal components to be computed. If you specify METHOD=ITERGS, METHOD=NIPALS, or METHOD=EIG, the default is the number of variables. If you specify METHOD=RANDOM, the default is 1. The *number* must be an integer greater than or equal to 0. If N=0, the default is used.

NOINT
omits the intercept from the model. In other words, this option requests that the covariance or correlation matrix not be corrected for the mean. When you specify this option, the covariance matrix and, hence, the standard deviations are not corrected for the mean. If you want to obtain the standard deviations corrected for the mean, you can obtain them by using a procedure such as PROC MEANS in Base SAS.

OUTSTAT=*libref.data-table*
creates an output data table to contain means, standard deviations, number of observations, correlations or covariances, eigenvalues, and eigenvectors. *libref.data-table* is a two-level name, where *libref* refers to the library, and *data-table* specifies the name of the output data table. For more information about this two-level name, see the DATA= option and the section "Using CAS Sessions and CAS Engine Librefs" on page 1293.

If you specify the COV option, the output data table contains a covariance matrix, and the corresponding _TYPE_ variable value is `'COV'` or `'UCOV'`, depending on the value of the NOINT option; otherwise, the output data table contains a correlation matrix, and the corresponding _TYPE_ variable value is `'CORR'` or `'UCORR'`, depending on the value of the NOINT option. If you specify the PARTIAL statement, the OUTSTAT= data table also contains R-squares.

For more information about the OUTSTAT= data table, see the section "OUTSTAT= Data Table" on page 1313.

PLOTS <(*global-plot-options*)> <= *plot-request* <(*options*)> >

PLOTS <(*global-plot-options*)> <= (*plot-request* <(*options*)> <... *plot-request* <(*options*)> >) >

controls the plots that are produced through ODS Graphics. When you specify only one *plot-request*, you can omit the parentheses around it. Some examples follow:

```
plots
plots=none
plots=(eigen profile)
plots(unpack)=scree
plots=pattern(ncomp=3 circles=25 50 75 1)
```

ODS Graphics must be enabled before you can request plots. For example:

```
ods graphics on;
proc pca plots=all;
   var x1--x10;
run;
ods graphics off;
```

You must specify the PLOTS option in order for the PCA procedure to produce plots. If you do not specify a *plot-request*, PROC PCA produces the scree plot by default.

You can specify the following *global-plot-options*:

UNPACKPANEL

UNPACK

suppresses paneling in the scree plot. By default, multiple plots can appear in an output panel. Specify UNPACKPANEL if you want each plot to appear in a separate panel. You can specify PLOTS(UNPACKPANEL) to unpack the default plots. You can also specify UNPACKPANEL as a suboption of the SCREE option (such as PLOTS=SCREE(UNPACKPANEL)).

You can specify the following *plot-requests*:

ALL

produces all appropriate plots. You can specify other options along with ALL; for example, to request all plots and unpack only the scree plot, specify PLOTS=(ALL SCREE(UNPACKPANEL)).

EIGEN <(UNPACKPANEL)>

EIGENVALUE <(UNPACKPANEL)>

SCREE <(UNPACKPANEL)>

produces the scree plot of eigenvalues and proportion variance explained. By default, both plots appear in the same panel. Specify PLOTS= SCREE(UNPACKPANEL) if you want each plot to appear in a separate panel.

NONE
suppresses the display of all plots.

PATTERN < (*pattern-options*) >
produces the pairwise component pattern plots. Each variable is plotted as an observation whose coordinates are correlations between the variable and the two corresponding components in the plot. Use the NCOMP= option (for example, PLOTS=PATTERN(NCOMP=3)) as described in the following list to control the number of plots to display.

You can specify the following *pattern-options*:

CIRCLES < =*number-list* >
plots the variance percentage circles. For each number c ($0 < c \leq 1$) that you specify, a ($c \times 100\%$) variance circle is displayed. For each number c ($c > 1$) that you specify, a $c\%$ variance circle is displayed. You can specify either CIRCLES=0.05 1 or CIRCLES=5 100 to display 5% and 100% variance circles. PLOTS=PATTERN(CIRCLES) and PLOTS=PATTERN(VECTOR) both display a unit circle (100% variance). By default, no circle is displayed when you specify PLOTS=PATTERN.

FLIP
interchanges (flips) the X-axis and Y-axis dimensions of the component pattern plots.

NCOMP=*n*
specifies the number of components n (≥ 2) to be plotted. The default is 5 or the total number of components m (≥ 2), whichever is smaller. If $n > m$, NCOMP=m is used. Be aware that the number of plots ($n \times (n-1)/2$) that are produced grows quadratically when *n* increases.

VECTOR
plots the pattern in a vector form.

PATTERNPROFILE

PROFILE
produces the pattern profile plot. Each component has its own profile. The Y-axis value represents the correlation between the variable (corresponding to the X-axis value) and the profiled principal component.

PREFIX=*name*
specifies a prefix for naming the principal components. By default, the *names* are Prin1, Prin2, ..., Prin*n*. If you specify PREFIX=Abc, the components are named Abc1, Abc2, Abc3, and so on. The number of characters in the prefix plus the number of digits required to designate the variables should not exceed the current name length that is defined by the VALIDVARNAME= system option.

PARPREFIX=*name*

PPREFIX=*name*

RPREFIX=*name*
specifies a prefix for naming the residual variables in the OUTSTAT= data table. By default, the prefix is R_. The number of characters in the prefix plus the maximum length of the variable names should not exceed the current name length that is defined by the VALIDVARNAME= system option.

SINGULAR=*p*

SING=*p*

specifies the singularity criterion, where $0 < p < 1$. If a variable in a PARTIAL statement has an R-square as large as $1 - p$ when predicted from the variables listed before it in the statement, the variable is assigned a standardized coefficient of 0. By default, SINGULAR=1E–8.

STANDARD

STD

standardizes the principal component scores in the OUT= data table in the OUTPUT statement to unit variance. If you omit this option, the scores have a variance equal to the corresponding eigenvalue. Note that this option has no effect on the eigenvalues themselves.

VARDEF=DF | N | WDF | WEIGHT | WGT

specifies the divisor to be used to calculate variances and standard deviations. By default, VARDEF=DF. The following table displays the values and associated divisors:

Value	Divisor	Formula	
DF	Error degrees of freedom	$n - i$	(before partialing)
		$n - p - i$	(after partialing)
N	Number of observations	n	
WEIGHT | WGT	Sum of weights	$\sum_{j=1}^{n} w_j$	
WDF	Sum of weights minus one	$\left(\sum_{j=1}^{n} w_j\right) - i$	(before partialing)
		$\left(\sum_{j=1}^{n} w_j\right) - p - i$	(after partialing)

In the formulas for VARDEF=DF and VARDEF=WDF, p is the number of degrees of freedom of the variables in the PARTIAL statement, and i is 0 if you specify the NOINT option and 1 otherwise.

BY Statement

BY *variables* **;**

You can specify a BY statement in PROC PCA to obtain separate analyses of observations in groups that are defined by the values of the BY variables. If you specify more than one BY statement, only the last one specified is used. For more information, see the discussion of BY-group processing in the "Grouping Data" section of *SAS Programmers Guide: Essentials*.

CODE Statement

CODE < *options* > **;**

The CODE statement writes SAS DATA step code for computing the principal component scores either to a file, to a catalog entry, or to a data table. To score new data, you can then include the file in a DATA step, or you can specify the table in the **runCodeTable** action in the **dataStep** action set (for more information, see *SAS Viya: System Programming Guide*).

The CODE statement is not supported when you specify the PARTIAL statement. If you specify more than one CODE statement, only the first one specified is used.

Table 25.4 summarizes the *options* available in the CODE statement.

Table 25.4 CODE Statement Options

Option	Description
COMMENT	Adds comments to the generated code
FILE=	Names the file in which to save the generated code
FORMATWIDTH=	Specifies the numeric format width for the regression coefficients
INDENTSIZE=	Specifies the number of spaces to indent the generated code
LABELID=	Specifies a number used to construct names and labels
LINESIZE=	Specifies the line size for the generated code
NOTRIM	Compares formatted values, including blank padding
OUT=	Names an output table in which to save the generated code

For more information about the syntax of the CODE statement, see the section "CODE Statement" on page 13 in Chapter 2, "Shared Concepts."

DISPLAY Statement

DISPLAY < *table-list* > < / *options* > **;**

The DISPLAY statement enables you to specify a list of display tables to display or exclude. This statement is similar to the ODS SELECT, ODS EXCLUDE, and ODS TRACE statements. However, the DISPLAY statement can improve performance when a large number of tables could be generated (such as in BY-group processing). The procedure processes the DISPLAY statement on a CAS server and thus sends only a subset of ODS tables to the SAS client. Because ODS statements are processed on a SAS client, first all the generated display tables are sent to the client, and then the client creates a subset.

If you use both DISPLAY and ODS statements together, the DISPLAY statement takes precedence over the ODS statements. Note that the ODS EXCLUDE statement processes tables that are sent to the client after they have been filtered by the DISPLAY statement. In some cases, it might appear that the ODS EXCLUDE statement is taking precedence because it can further filter the tables. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

You can specify the *table-list* as a list of table names, paths, partial pathnames, and regular expressions.

The table names that you can specify are listed in the section "ODS Table Names" on page 1317. A path is a table name that is prefixed with dot-separated grouping information. For example, a SelectionSummary table that a procedure produces during a selection routine might have the path *Bygroup1.Summary.SelectionSummary*. A partial pathname does not include all groups; for example, *SelectionSummary* and *Summary.SelectionSummary* are partial pathnames for *Bygroup1.Summary.SelectionSummary*.

When you specify a table name or partial pathname, all display tables whose paths end in the specified name are selected for display or exclusion. For example, both *SelectionSummary* and *Summary.SelectionSummary* select *Bygroup1.Summary.SelectionSummary*.

A regular expression is enclosed in forward slashes (/). For example, specifying "/tions/" selects all pathnames that contain the substring "tions"; in particular, the *Bygroup1.Summary.SelectionSummary* table is selected. Specifying "!/tions/" selects all pathnames that do not contain the substring "tions"; in particular, the *Bygroup1.Summary.SelectionSummary* table is not selected.

You can specify the following *options* after a slash (/):

CASESENSITIVE
performs a case-sensitive comparison of table names in the *table-list* to display table names when tables are subsetted for display. To preserve case, you must enclose table names in the *table-list* in quotation marks.

EXCLUDE
displays all display tables except those that you specify in the *table-list*.

EXCLUDEALL
suppresses display of all tables. This option takes precedence over the other options.

TRACE
displays the display table names, labels, and paths.

DISPLAYOUT Statement

DISPLAYOUT *table-spec-list* < / *options* > **;**

The DISPLAYOUT statement enables you to create output tables from your displayed output. This statement is similar to the ODS OUTPUT statement. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

The *table-spec-list* specifies a list of output tables to create. Each entry in the list has either a *key*=*value* format or a *key* format:

key=*value* specifies *key* as the ODS table name, path, or partial pathname, and specifies *value* as the output table name.

key specifies *key* as the ODS table name and also as the output table name.

The ODS table names that you can specify are listed in the section "ODS Table Names" on page 1317. You cannot specify the ODS table named OutputCasTables in the *table-spec-list*.

Table names and partial pathnames are discussed under the DISPLAY statement. The DISPLAYOUT statement does not support regular expressions.

You can specify the following *options* after a slash (/):

INCLUDEALL
creates output tables for all display tables. The name of the created output table is the same as the name of the corresponding display table. If you specify this option, the *table-spec-list* specification is ignored.

NOREPLACE
does not replace any existing output table of the same name.

REPEATED
replicates all output tables on all nodes.

FREQ Statement

FREQ *variable* **;**

The *variable* in the FREQ statement identifies a numeric variable in the input data table that contains the frequency of occurrence of each observation. PROC PCA treats each observation as if it appears f times, where f is the value of the FREQ *variable* for the observation. If f is not an integer, it is truncated to an integer. If f is less than 1 or missing, the observation is not used in the analysis. When you omit the FREQ statement, each observation is assigned a frequency of 1.

The FREQ statement is not supported if you specify METHOD=ITERGS, METHOD=NIPALS, or METHOD=RANDOM in the PROC PCA statement.

OUTPUT Statement

OUTPUT OUT=*libref.data-table*
< **COPYVARS=***(variables)* >
< *keyword* < **=***prefix* > >. . . < *keyword* < **=***prefix* > > **;**

The OUTPUT statement creates a data table that contains observationwise statistics, which are computed after PROC PCA fits the model. If you do not specify a *keyword*, then only the principal component scores are included.

The variables in the input data table are *not* included in the output data table, in order to avoid data duplication for large data tables; however, variables that you specify in the COPYVARS= option are included.

You must specify the following option:

OUT=*libref.data-table*
names the output data table for PROC PCA to use. You must specify this option before any other options. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to where the data table is to be stored, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 1293.

data-table specifies the name of the output data table.

You can also specify the following syntax elements:

COPYVAR=*variable*

COPYVARS=(*variables***)**

transfers one or more *variables* from the input data table to the output data table.

keyword **< =***prefix* **>**

specifies a statistic to include in the output data table and optionally a *prefix* for naming the output variables. If you do not provide a *prefix*, the PCA procedure assigns a default prefix based on the type of statistic requested. For example, for the VAR statement variables x1 and x2, RESIDUAL produces two residual value variables, R_x1 and R_x2.

You can specify the following *keywords* to add statistics to the OUTPUT data table:

H

requests the approximate leverage. The default prefix is H.

STD

requests standardized (centered and scaled) VAR statement variable values for each VAR statement variable. The default prefix is Std.

STDSSE

requests the sum of squares of residuals for standardized VAR statement variables. The default prefix is StdSSE.

TSQUARE

T2

requests scaled sum of squares of score values. The default prefix is TSquare.

RESIDUAL

RESID

R

requests residuals for each VAR statement variable. The default prefix is R.

SCORE

requests principal component scores for each principal component. The default prefix is Score.

If you specify METHOD=EIG, the only valid *keywords* are RESIDUAL (if you also specify the PARTIAL statement; PROC PCA computes the residuals by predicting the VAR statement variables from the PARTIAL statement variables) and SCORE. Other *keywords* are ignored. The output variables that contain the principal component scores have mean 0 and a variance equal to the corresponding eigenvalue, unless you specify the STANDARD option to standardize the scores to unit variance. Also, if you specify the COV option, PROC PCA computes the principal component scores from the corrected variables or (if the NOINT option is specified) uncorrected variables rather than from the standardized variables.

The output variables that contain the requested statistic are named as follows, according to the *keyword* that you specify:

- The *keywords* RESIDUAL and STD define an output variable for each VAR statement variable, so the variables that correspond to each VAR statement variable are named by appending the name of the VAR statement variable to the prefix. For example, if the model has the VAR statement variables x1 and x2, then RESIDUAL=R produces the variables R_x1 and R_x2.
- The *keyword* SCORE defines an output variable for each principal component, so the variables that correspond to each successive component are named by appending the component number to the prefix. For example, if the model has three principal components, then SCORE=T produces the variables T1, T2, and T3.
- The *keywords* H, STDSSE, and TSQUARE each define a single output variable, so the variable name matches the prefix.

PARTIAL Statement

PARTIAL *variables* **;**

If you want to analyze a partial correlation or covariance matrix, use the PARTIAL statement to specify the names of the numeric variables to be partialed out. The PCA procedure computes the principal components of the residuals by predicting the VAR statement variables from the PARTIAL statement variables. If you request an OUTSTAT= data table, the residual variables are named by prefixing either the characters R_ (by default) or the string specified in the PARPREFIX= option to the VAR statement variable names.

The PARTIAL statement is not supported if you specify METHOD=ITERGS, METHOD=NIPALS, or METHOD=RANDOM in the PROC PCA statement.

STORE Statement

STORE < **OUT=** >*libref.data-table* **;**

The STORE statement saves the context and results of the statistical analysis. The resulting analytic store has a binary file format that you cannot modify. An example of analytic store technology is to perform a time-consuming analysis and to store its results by using the STORE statement. Later you can perform specific statistical analysis tasks that are based on the saved results of the previous analysis, without having to fit the model again. You can process the contents of the analytic store by using the ASTORE procedure or the **score** action in the **aStore** action set. For more information, see Chapter 4, "ASTORE Procedure" (*SAS Viya: Machine Learning Procedures*).

libref.data-table is a two-level name, where *libref* refers to the library, and *data-table* specifies the name of the output data table. For more information about this two-level name, see the section "Using CAS Sessions and CAS Engine Librefs" on page 1293.

The STORE statement is ignored if you specify the BY statement or the PARTIAL statement.

VAR Statement

VAR *variables* **;**

The VAR statement lists the numeric variables to be analyzed. If you omit the VAR statement, all numeric variables that are not specified in other statements are analyzed.

WEIGHT Statement

WEIGHT *variable* **;**

The *variable* in the WEIGHT statement is used as a weight to perform a weighted analysis of the data. Observations that have nonpositive or missing weights are not included in the analysis. The values of the WEIGHT variable can be nonintegral and are not truncated. When you omit the WEIGHT statement, all observations that are used in the analysis are assigned a weight of 1.

The WEIGHT statement is not supported if you specify METHOD=ITERGS, METHOD=NIPALS, or METHOD=RANDOM in the PROC PCA statement.

Details: PCA Procedure

Computing Principal Components

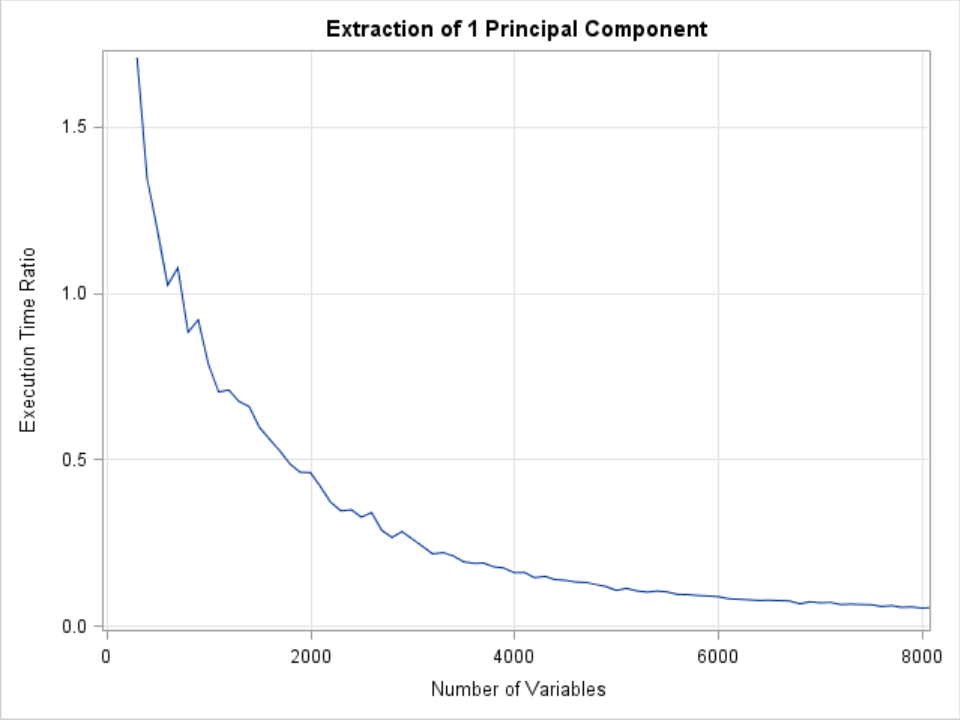
The PCA procedure implements several methods of calculating principal components: eigenvalue decomposition (EIG), the nonlinear iterative partial least squares (NIPALS) method, the iterative method based on the Gram-Schmidt orthogonalization (ITERGS) of Andrecut (2009), and the iterative method based on random projection (RANDOM) of Halko, Martinsson, and Tropp (2011). The EIG, NIPALS, and ITERGS methods provide exact principal component solutions, whereas the RANDOM method gives approximate principal components.

Each method has computational strengths; Table 25.5 provides guidelines for selecting an efficient (fast) method. The guidelines include the concept of a "small to moderate" and "moderate to large" number of variables. This concept depends on the number of data set observations, the number of principal components required, the PCA method parameter settings, and the grid computing environment. As a simple illustration of the efficiency relationship between the EIG and RANDOM methods, consider a scenario in which one principal component is extracted from a data set that consists of 1,000,000 observations that are distributed across a grid of 16 servers. Figure 25.5 depicts execution time for RANDOM divided by the execution time of EIG (labeled "Execution Time Ratio") as a function of the number of data set variables. Where execution time ratio is greater than 1, EIG is more efficient than RANDOM. For this scenario, "small to moderate" means approximately 700 variables or less and "moderate to large" means anything more than 700 variables.

Table 25.5 Guidelines for PCA Method Selection

Method	Efficient in These Scenarios
EIG	Small to moderate number of variables, and/or computing all principal components
NIPALS	Computing only the first few principal components
ITERGS	Computing only the first few principal components
RANDOM	Moderate to large number of variables

Figure 25.5 Illustration of Efficiency Crossover Point for EIG and RANDOM Methods



Eigenvalue Decomposition

Let $\mathbf{X}$ be a centered and scaled data matrix that has k numeric variables. The eigenvalue decomposition method bases the component extraction on the eigenvalue decomposition of the covariance matrix $\mathbf{X}'\mathbf{X}$, which extracts all the k principal components simultaneously. Each principal component is a linear combination of the original variables, and each component is orthogonal, with coefficients equal to the eigenvectors of the covariance matrix $\mathbf{X}'\mathbf{X}$. The eigenvectors are usually normalized to have unit length. The principal components are sorted by descending order of the eigenvalues, which are equal to the variances of the components.

NIPALS

The nonlinear iterative partial least squares (NIPALS) method extracts the principal components successively based on the data matrix $\mathbf{X}$. The NIPALS method starts by calculating the loadings, $\mathbf{p}$, as $\mathbf{p}' = (\mathbf{t}'\mathbf{t})^{-1}\mathbf{t}'\mathbf{X}$, where $\mathbf{t}$ is the score vector. It then calculates an improved score vector, $\mathbf{t} = \mathbf{X}\mathbf{p}$. The method iteratively computes the improved $\mathbf{p}$ and $\mathbf{t}$ until convergence is reached.

This process accounts for how the first principal component is extracted. The second component is extracted in the same way, by replacing $\mathbf{X}$ with the residual from the first component: $\mathbf{E} = \mathbf{X} - \mathbf{t}\mathbf{p}'$.

For large data matrices or matrices that have a high degree of column collinearity, the NIPALS method suffers from loss of orthogonality because of the machine-precision errors that accumulate at each iteration step. In practice, the NIPALS method is used to extract only the first few principal components.

ITERGS

The iterative method based on Gram-Schmidt orthogonalization (ITERGS) of Andrecut (2009) overcomes the issue of loss of orthogonality in the NIPALS method by applying Gram-Schmidt reorthogonalization correction to both the loadings and the scores at each iteration step:

$$\begin{aligned} \mathbf{p}_c &= \mathbf{p} - \mathbf{P}_k\mathbf{P}_k'\mathbf{p} \\ \mathbf{t}_c &= \mathbf{t} - \mathbf{T}_k\mathbf{T}_k'\mathbf{t} \end{aligned}$$

Here, $\mathbf{p}_c$ and $\mathbf{t}_c$ are the corrected loading vector and score vector, respectively. $\mathbf{P}_k$ is the matrix that is formed by using the first k loadings. $\mathbf{T}_k$ is the matrix that is formed by using the first k scores.

The ITERGS method stabilizes the iterative process at the cost of increased computational effort.

RANDOM

The iterative method based on random projection (RANDOM) of Halko, Martinsson, and Tropp (2011) computes approximate scores and loadings by using a two-step process. The first step calculates an approximate basis for the range of $\mathbf{X}$ (that is, $\mathbf{X} \approx \mathbf{Q}\mathbf{Q}^T\mathbf{X}$), where $\mathbf{Q}$ has orthonormal columns. The second step computes a sketch of $\mathbf{X}$ from $\mathbf{B} = \mathbf{Q}^T\mathbf{X}$. The singular value decomposition of $\mathbf{B}$ gives an equivalent representation $\tilde{\mathbf{U}}\mathbf{S}\mathbf{V}^T$, and simple manipulations of these matrices provide the approximate loadings $\mathbf{P} = \mathbf{V}$ and the scores $\mathbf{T} = \mathbf{Q}\tilde{\mathbf{U}}\mathbf{S}$.

The iterative part of this method is the construction of the orthonormal basis matrix $\mathbf{Q}$. Conceptually, you can obtain $\mathbf{Q}$ by QR factorization of $\mathbf{Y} = (\mathbf{X}\mathbf{X}^T)^i\mathbf{X}\boldsymbol{\Omega}$, where $\boldsymbol{\Omega}$ is a random Gaussian projection matrix and i is the number of iterations. Here the notion of iteration refers to multiplication of a sample matrix by $\mathbf{X}^T$ and then $\mathbf{X}$. This conceptual approach is susceptible to rounding error, so in practice you orthonormalize the columns of the sample matrix for numerical stability. Here, the initial sample matrix is the subspace defined by the QR factorization of a random projection of the data matrix $\mathbf{X}\boldsymbol{\Omega} = \mathbf{Q}_0\mathbf{R}_0$. Each iteration ($j = 1, 2, \ldots, i$) then consists of alternating projection and orthonormalization steps of

$$\begin{aligned} \tilde{\mathbf{Y}}_j &= \mathbf{X}^T\mathbf{Q}_{j-1} \\ \tilde{\mathbf{Q}}_j\tilde{\mathbf{R}}_j &= \tilde{\mathbf{Y}}_j \end{aligned}$$

and

$$\begin{aligned} \mathbf{Y}_j &= \mathbf{X}\tilde{\mathbf{Q}}_j \\ \mathbf{Q}_j\mathbf{R}_j &= \mathbf{Y}_j \end{aligned}$$

When $j = i$, $\mathbf{Q}_j$ is the desired orthonormal basis matrix.

Missing Values

Observations that have missing values for any variable in the VAR, PARTIAL, FREQ, or WEIGHT statement are omitted from the analysis and are given missing values for principal component scores in the OUT= data table that is specified in the OUTPUT statement.

OUTSTAT= Data Table

The OUTSTAT= data table contains the following variables:

- the BY variables, if any
- two new variables, _TYPE_ and _NAME_, both character variables
- the variables that are analyzed (that is, those in the VAR statement); or, if there is no VAR statement, all numeric variables not listed in any other statement; or, if there is a PARTIAL statement, the residual variables that contain the residuals computed by predicting the VAR statement variables from the PARTIAL statement variables.

Each observation in the new data table contains some type of statistic, as indicated by the _TYPE_ variable. The values of the _TYPE_ variable are as follows:

TYPE	Contents
MEAN	mean of each variable. If you specify the PARTIAL statement, this observation is omitted.
STD	standard deviations. If you specify the COV option, this observation is omitted. If you specify the PARTIAL statement, the standard deviation of a variable is computed as its root mean squared error as predicted from the PARTIAL statement variables.
USTD	uncorrected standard deviations. When you specify the NOINT option in the PROC PCA statement, the OUTSTAT= data table contains standard deviations not corrected for the mean. However, if you also specify the COV option in the PROC PCA statement, this observation is omitted.
N	number of observations on which the analysis is based. This value is the same for each variable. If you specify the PARTIAL statement and the value of the VARDEF= option is DF or unspecified, then the number of observations is decremented by the degrees of freedom for the PARTIAL statement variables.

SUMWGT	the sum of the weights of the observations. This value is the same for each variable. If you specify the PARTIAL statement and VARDEF=WDF, then the sum of the weights is decremented by the degrees of freedom for the PARTIAL statement variables. This observation is output only if the value is different from that in the observation for which _TYPE_='N'.
CORR	correlations between each variable and the variable specified by the _NAME_ variable. The number of observations for which _TYPE_='CORR' is equal to the number of variables being analyzed. If you specify the COV option, no _TYPE_='CORR' observations are produced. If you use the PARTIAL statement, then the partial correlations, not the raw correlations, are output.
UCORR	uncorrected correlation matrix. When you specify the NOINT option without the COV option in the PROC PCA statement, the OUTSTAT= data table contains a matrix of correlations not corrected for the means. However, if you also specify the COV option in the PROC PCA statement, this observation is omitted.
COV	covariances between each variable and the variable specified by the _NAME_ variable. _TYPE_='COV' observations are produced only if you specify the COV option. If you specify the PARTIAL statement, the partial covariances, not the raw covariances, are output.
UCOV	uncorrected covariance matrix. When you specify the NOINT and COV options in the PROC PCA statement, the OUTSTAT= data table contains a matrix of covariances not corrected for the means.
EIGENVAL	eigenvalues. If the N= option requests less than the maximum number of principal components, only the specified number of eigenvalues are produced, and missing values fill out the observation.
SCORE	eigenvectors. The _NAME_ variable contains the name of the corresponding principal component as constructed from the PREFIX= option. The number of observations for which _TYPE_='SCORE' equals the number of principal components computed. The eigenvectors have unit length unless you specify the STD option, in which case the unit-length eigenvectors are divided by the square roots of the eigenvalues to produce scores that have unit standard deviations. To obtain the principal component scores, if the COV option is not specified, these coefficients should be multiplied by the standardized data. For the COV option, these coefficients should be multiplied by the centered data. To center and standardize the data, you should use means that are obtained from the observation for which _TYPE_='MEAN' and standard deviations that are obtained from the observation for which _TYPE_='STD'.
USCORE	scoring coefficients to be applied without subtracting the mean from the raw variables. Observations for which _TYPE_='USCORE' are produced when you specify the NOINT option in the PROC PCA statement. To obtain the principal component scores, these coefficients should be multiplied by the data that are standardized by the uncorrected standard deviations obtained from the observation for which _TYPE_='USTD'.
RSQUARED	R-squares for each VAR statement variable as predicted by the PARTIAL statement variables.
B	regression coefficients for each VAR statement variable as predicted by the PARTIAL statement variables. This observation is produced only if you specify the COV option.

STB standardized regression coefficients for each VAR statement variable as predicted by the PARTIAL statement variables. If you specify the COV option, this observation is omitted.

Displayed Output

The following sections describe the output that PROC PCA produces. The output is organized into various tables, which are discussed in their order of appearance.

Model Information

The "Model Information" table displays basic information about the model, including the input data table and the principal component extraction method that is used in the analysis.

Number of Variables

The "Number of Variables" table displays the number of VAR statement variables, the number of PARTIAL statement variables, and the number of principal components to be extracted.

Number of Observations

The "Number of Observations" table displays the number of observations read from the input data table and the number of observations used in the analysis. If you specify a FREQ statement, the table also displays the sum of frequencies read and used.

Simple Statistics

If you specify METHOD=EIG, the PCA procedure produces a "Simple Statistics" table that displays the mean and standard deviation (std) for each variable. If you specify the NOINT option, the uncorrected standard deviation (ustd) is displayed.

Centering and Scaling Information

If you specify METHOD=ITERGS, METHOD=NIPALS, or METHOD=RANDOM, the PCA procedure produces a "Centering and Scaling Information" table that displays the centering and scaling information for each variable.

Explained Variation of Variables

If you specify METHOD=ITERGS, METHOD=NIPALS, or METHOD=RANDOM, the PCA procedure produces an "Explained Variation of Variables" table that displays the fraction of variation that is accounted for in each variable by each successive principal component.

Correlation Matrix

If you specify METHOD=EIG, the PCA procedure produces a "Correlation Matrix" table that displays the correlation matrix or, if you specify the COV option, the covariance matrix.

Regression Statistics

When you specify the PARTIAL statement, the PCA procedure produces a "Regression Statistics" table that displays the R-square and root mean squared error (RMSE) for each VAR statement variable as predicted by the PARTIAL statement variables.

Regression Coefficients

When you specify the PARTIAL statement, the PCA procedure produces a "Regression Coefficients" table that displays standardized regression coefficients or, if you specify the COV option, regression coefficients for predicting the VAR statement variables from the PARTIAL statement variables.

Partial Correlation Matrix

When you specify the PARTIAL statement, the PCA procedure produces a "Partial Correlation Matrix" table that displays the partial correlation matrix or, if you specify the COV option, the partial covariance matrix.

Total Variance

If you specify METHOD=EIG and the COV option, the PCA procedure produces a simple table that displays the total variance.

Eigenvalues

The "Eigenvalues" table displays eigenvalues of the correlation or covariance matrix (if you specify METHOD=EIG) or eigenvalues of the data matrix (if you specify METHOD=ITERGS, METHOD=NIPALS, or METHOD=RANDOM), along with the difference between successive eigenvalues, the proportion of variance explained by each eigenvalue, and the cumulative proportion of variance explained.

Eigenvectors

If you specify METHOD=EIG, the PCA procedure produces an "Eigenvectors" table that displays the eigenvectors.

Loadings

If you specify METHOD=ITERGS, METHOD=NIPALS, or METHOD=RANDOM, the PCA procedure produces a "Loadings" table that displays the loadings.

Timing Information

The "Timing" table displays the elapsed time of each main task of the procedure.

OutputCasTables Table

The OutputCasTables table is a special table that has information about each CAS table that is created during a CAS action execution. The information for each CAS table consists of the CAS table name, the caslib in which the table resides, and the number of columns and rows in the CAS table. Because this table is not a typical ODS table that contains analytical results, you cannot include it in the *table-spec-list* in the DISPLAYOUT statement.

ODS Table Names

Each table that the PCA procedure creates has a name associated with it. You must use this name to refer to the table when you use the DISPLAY statement, the DISPLAYOUT statement, or the Output Delivery System (ODS) statements. These names are listed in Table 25.6.

Table 25.6 ODS Tables Produced by PROC PCA

Table Name	Description	Required Statement / Option
CenScaleInfo	Centering and scaling information	METHOD=ITERGS | NIPALS | RANDOM
Corr	Correlation matrix	METHOD=EIG
Cov	Covariance matrix	METHOD=EIG and COV
Eigenvalues	Eigenvalues	Default output
Eigenvectors	Eigenvectors	METHOD=EIG
Loadings	Loadings	METHOD=ITERGS | NIPALS | RANDOM
ModelInfo	Model information	Default output
NObs	Number of observations read and used	Default output
NVars	Number of variables, partial variables, and principal components	Default output
OutputCasTables	See the section "OutputCasTables Table" on page 1317	DISPLAYOUT statement | OUTPUT statement / OUTSTAT=
ParCorr	Partial correlation matrix	PARTIAL statement
ParCov	Uncorrected partial covariance matrix	PARTIAL statement and COV
RegCoef	Regression coefficients	PARTIAL statement and COV
RSquareRMSE	Regression statistics: R-squares and RMSEs	PARTIAL statement
SimpleStatistics	Simple statistics	METHOD=EIG
StdRegCoef	Standardized regression coefficients	PARTIAL statement
Timing	Absolute and relative times of tasks that are performed by the procedure	Default output
TotalVariance	Total variance	METHOD=EIG and COV
Variation	Explained variation of variables	METHOD=ITERGS | NIPALS | RANDOM

ODS Graphics

You can refer to every graph produced through ODS Graphics by name. The names of the graphs that PROC PCA generates are listed in Table 25.7, along with a description of each graph and the required statements and options.

Table 25.7 Graphics Produced by PROC PCA

ODS Graph Name	Plot Description	Required Statement / Option
PatternPlot	Component pattern plot	PLOTS=PATTERN
PatternProfilePlot	Component pattern profile plot	PLOTS=PATTERNPROFILE
ScreePlot	Scree and variance plots	Default output | PLOTS=SCREE
VariancePlot	Variance proportion-explained plot	PLOTS=SCREE(UNPACKPANEL)

Examples: PCA Procedure

The examples in this chapter are available in the GitHub repository located at `https://github.com/sascommunities/doc-supplement-casstat`.

NOTE: Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections "Using CAS Sessions and CAS Engine Librefs" on page 7 and "Loading a SAS Data Set onto a CAS Server" on page 7 in Chapter 2, "Shared Concepts."

Example 25.1: Analyzing Mean Temperatures of US Cities

(View the complete code for this example (pcaex1.sas) in the example repository.)

This example analyzes mean daily temperatures of selected US cities in January and July. The following DATA step creates the data:

```
data mylib.Temperature;
   length Cityid $ 2;
   title 'Mean Temperature of Selected Cities in January and July';
   input City $1-15 January July;
   Cityid = substr(City,1,2);
   datalines;
Mobile          51.2 81.6
Phoenix         51.2 91.2
Little Rock     39.5 81.4
Sacramento      45.1 75.2
Denver          29.9 73.0

   ... more lines ...
```

```
Cheyenne        26.6 69.1
;
```

The following statements invoke the PCA procedure, which requests a principal component analysis of these data and outputs the scores to an output data table. The Cityid variable is also included in the output data table (COPYVARS= Cityid).

```
title 'Mean Temperature of Selected Cities in January and July';
proc pca data=mylib.Temperature cov;
   var July January;
   output out=mylib.Scores copyVars=Cityid;
run;
```

Output 25.1.1 displays the PROC PCA output. The standard deviation of January (11.712) is higher than the standard deviation of July (5.128). The COV option in the PROC PCA statement requests that the principal components be computed from the covariance matrix. The total variance is 163.474. The first principal component accounts for about 94% of the total variance, and the second principal component accounts for only about 6%. The eigenvalues sum to the total variance.

Note that January receives a higher loading on Prin1 because it has a higher standard deviation than July. Also note that the PCA procedure calculates the scores by using the centered variables rather than the standardized variables.

Output 25.1.1 Results of Principal Component Analysis

Mean Temperature of Selected Cities in January and July

The PCA Procedure

Model Information	
Data Source	TEMPERATURE
Component Extraction Method	Eigenvalue Decomposition

Number of Variables	2
Number of Principal Components	2

Number of Observations Read	64
Number of Observations Used	64

Simple Statistics

Variable	Mean	Standard Deviation
July	75.60781	5.12762
January	32.09531	11.71243

Covariance Matrix

Variable	July	January
July	26.29248	46.82829
January	46.82829	137.18109

Total Variance	163.47356647

Output 25.1.1 *continued*

Eigenvalues of the Covariance Matrix

	Eigenvalue	Difference	Proportion	Cumulative
1	154.310607	145.147647	0.9439	0.9439
2	9.162960		0.0561	1.0000

Eigenvectors

Variable	Prin1	Prin2
July	0.34353	-0.93914
January	0.93914	0.34353

Example 25.2: Extracting Principal Components with NIPALS

(View the complete code for this example (pcaex2.sas) in the example repository.)

This example demonstrates the NIPALS method in PROC PCA, which extracts principal components successively. The data that this example uses are from the Getting Started section; they provide crime rates per 100,000 people in seven categories for each of the 50 US states in 1977. The following DATA step generates the data:

```
data mylib.Crime;
   title 'Crime Rates per 100,000 Population by State';
   input State $1-15 Murder Rape Robbery Assault
         Burglary Larceny Auto_Theft;
   datalines;
Alabama        14.2 25.2  96.8 278.3 1135.5 1881.9 280.7
Alaska         10.8 51.6  96.8 284.0 1331.7 3369.8 753.3
Arizona         9.5 34.2 138.2 312.3 2346.1 4467.4 439.5
Arkansas        8.8 27.6  83.2 203.4  972.6 1862.1 183.4
California     11.5 49.4 287.0 358.0 2139.4 3499.8 663.5

   ... more lines ...

Wisconsin       2.8 12.9  52.2  63.7  846.9 2614.2 220.7
Wyoming           .  21.9  39.7 173.9  811.6 2772.2 282.0
;
```

The following statements use PROC PCA to extract principal components by using the NIPALS method, and produce default plots:

```
ods graphics on;

proc pca data=mylib.Crime method=nipals plots;
run;
```

Output 25.2.1 displays the PROC PCA output. The "Model Information" table shows that the NIPALS method is used to extract principal components. The "Explained Variation of Variables" table lists the fraction of variation that is accounted for in each variable by each of the seven principal components. All the variation in each variable is accounted for by seven principal components because there are only seven variables. The eigenvalues indicate that two or three components provide a good summary of the data: two components

account for 76% of the total variance, and three components account for 87%. Subsequent components account for less than 5% each.

Note that in the Getting Started section, the principal components are extracted from the same data by using the eigenvalue decomposition method; the "Eigenvalues" table that is generated there matches the one generated by the NIPALS method. Also, the eigenvectors in the "Eigenvectors" table match the loading factors in the "Loadings" table.

Output 25.2.1 Results of Principal Component Analysis Using NIPALS

Crime Rates per 100,000 Population by State

The PCA Procedure

Model Information	
Data Source	CRIME
Component Extraction Method	NIPALS

Number of Variables	7
Number of Principal Components	7

Number of Observations Read	50
Number of Observations Used	48

Centering and Scaling Information

Variable	Subtracted off	Divided by
Murder	7.51667	3.93059
Rape	26.07500	10.81304
Robbery	127.55625	88.49374
Assault	214.58750	100.64360
Burglary	1316.37917	423.31261
Larceny	2696.88542	714.75023
Auto_Theft	383.97917	194.37033

Explained Variation of Variables

Variable	Prin1	Prin2	Prin3	Prin4	Prin5	Prin6	Prin7
Murder	0.37117	0.85539	0.87790	0.89562	0.97555	0.99143	1.00000
Rape	0.76242	0.79917	0.84059	0.84199	0.85065	0.99041	1.00000
Robbery	0.63783	0.64064	0.82164	0.92942	0.99788	0.99992	1.00000
Assault	0.63517	0.79127	0.79341	0.91781	0.98822	0.99513	1.00000
Burglary	0.78913	0.84414	0.88183	0.88207	0.88544	0.94800	1.00000
Larceny	0.51373	0.72178	0.93718	0.95479	0.95492	0.95530	1.00000
Auto_Theft	0.33638	0.65746	0.90481	0.96197	0.99623	0.99706	1.00000

Output 25.2.1 *continued*

Eigenvalues

	Eigenvalue	Difference	Proportion	Cumulative
1	4.045824	2.781795	0.5780	0.5780
2	1.264030	0.516529	0.1806	0.7586
3	0.747500	0.421175	0.1068	0.8653
4	0.326325	0.061119	0.0466	0.9120
5	0.265207	0.036843	0.0379	0.9498
6	0.228364	0.105613	0.0326	0.9825
7	0.122750		0.0175	1.0000

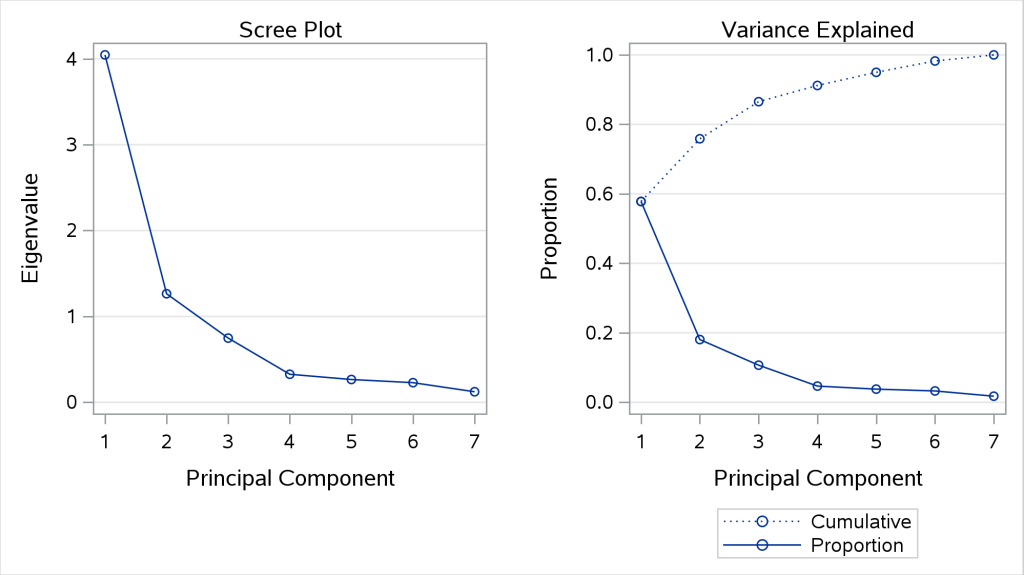
Loadings

Variable	Prin1	Prin2	Prin3	Prin4	Prin5	Prin6	Prin7
Murder	0.30289	-0.61893	0.17353	-0.23308	0.54896	-0.26371	-0.26428
Rape	0.43410	-0.17053	-0.23539	0.06540	0.18075	0.78232	0.27946
Robbery	0.39705	0.04713	0.49208	-0.57470	-0.50808	0.09452	0.02497
Assault	0.39622	-0.35142	-0.05343	0.61744	-0.51525	-0.17395	-0.19921
Burglary	0.44164	0.20861	-0.22454	-0.02750	0.11273	-0.52340	0.65085
Larceny	0.35634	0.40570	-0.53681	-0.23231	0.02172	-0.04085	-0.60346
Auto_Theft	0.28834	0.50400	0.57524	0.41853	0.35939	0.06024	-0.15487

PROC PCA produces the scree plot as shown in Output 25.2.2 by default when ODS Graphics is enabled and no plot request is specified in the PLOTS option. You can obtain more plots by specifying the plot requests in the PLOTS option in the PROC PCA statement.

The scree plot on the left shows that the eigenvalue of the first component is approximately 4 and the eigenvalue of the second component is largely decreased to under 1.5. The variance-explained plot on the right shows that the first two principal components account for nearly 80% of the total variance.

Output 25.2.2 Scree Plot from the PCA Procedure



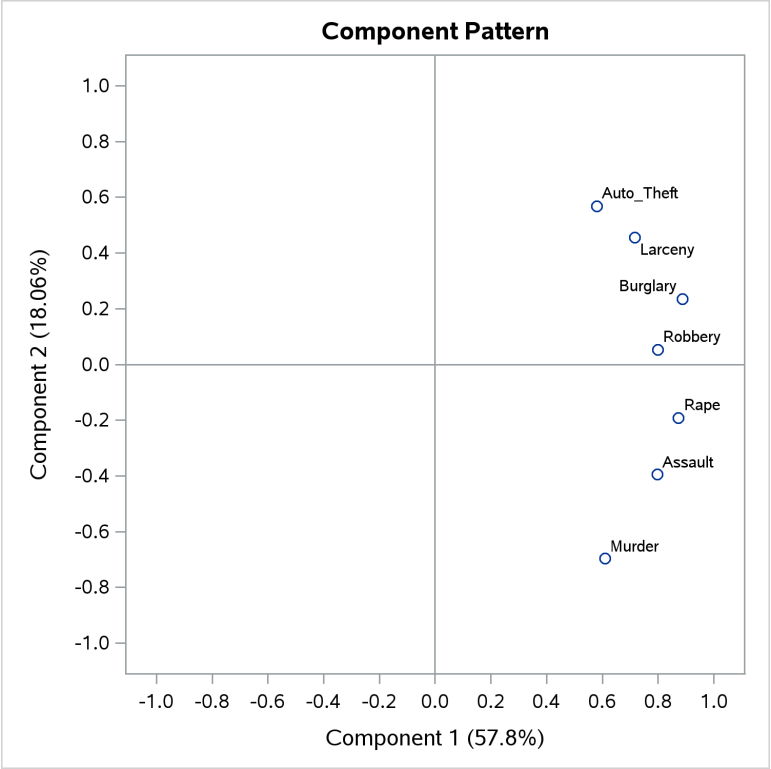
In addition to the scree plot, PROC PCA also produces the component pattern plot and the component pattern profile plot. The following statements request these plots:

```
proc pca data=mylib.Crime method=nipals
         plots=(pattern(ncomp=3) patternprofile);
run;
```

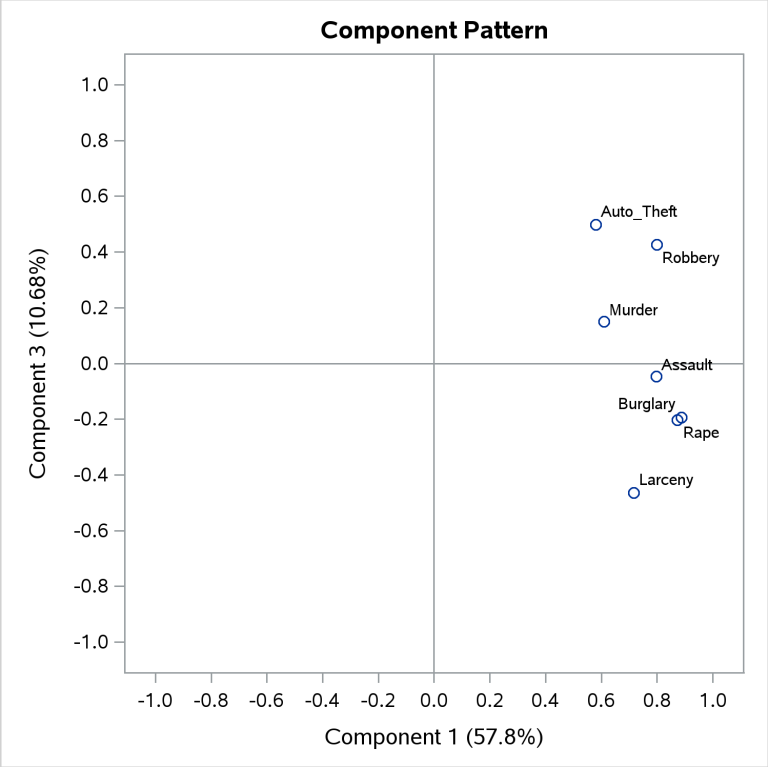
The pairwise component pattern plots are shown in Output 25.2.3 through Output 25.2.5. The pattern plots show the following:

- All variables positively and evenly correlate with the first principal component (Output 25.2.3 and Output 25.2.4).
- The variable Auto_Theft correlates highly with the second component, and the variable Murder correlates highly but negatively with the second component (Output 25.2.3).
- All the variables have low correlations (either positively or negatively) with the third component (Output 25.2.4).
- The variable Auto_Theft correlates highly with the second component, but it has low correlation with the third component; the variable Murder correlates highly but negatively with the second component, but it has very low correlation with the third component (Output 25.2.5).

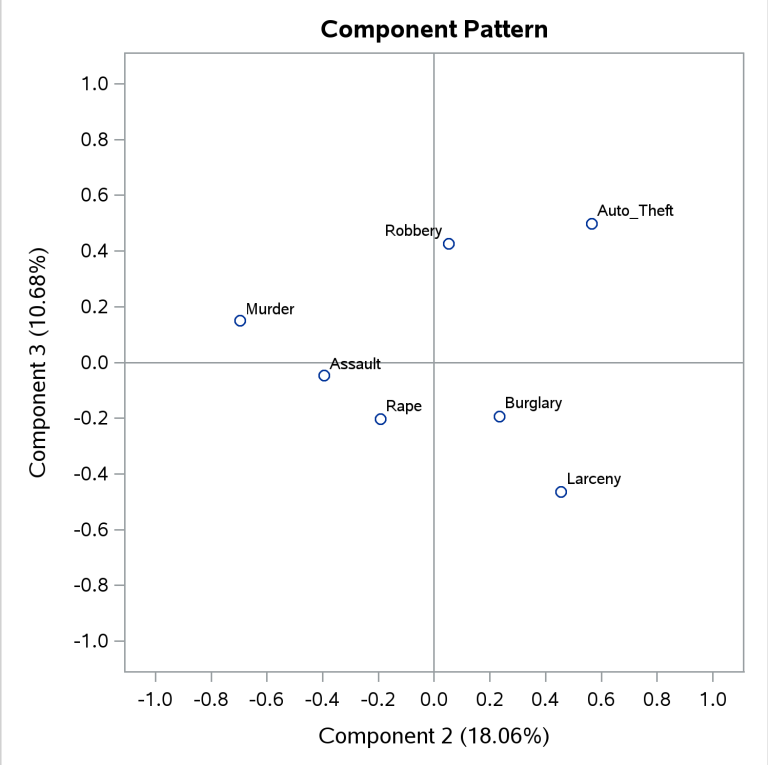
Output 25.2.3 Pattern Plot of Component 2 by Component 1



Output 25.2.4 Pattern Plot of Component 3 by Component 1

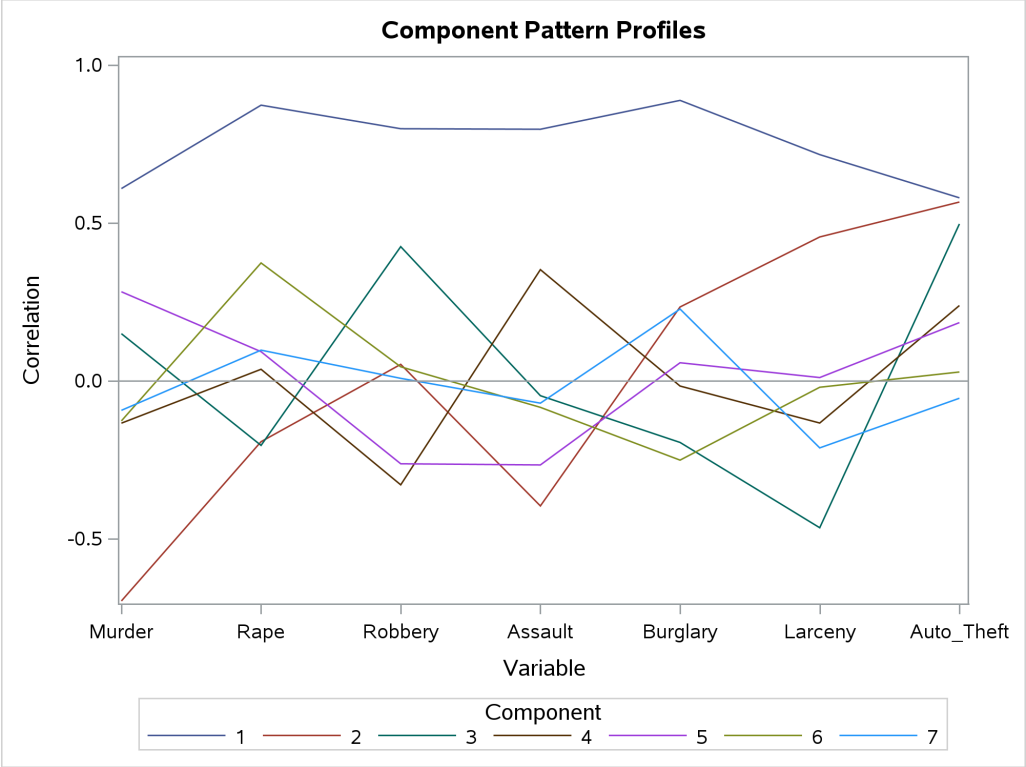


Output 25.2.5 Pattern Plot of Component 3 by Component 2



Output 25.2.6 shows a component pattern profile. As shown in the pattern plots, the nearly horizontal profile of the first component indicates that the first component is mostly correlated evenly across all variables.

Output 25.2.6 Component Pattern Profile Plot from the PCA Procedure



Example 25.3: Approximating Principal Components with the RANDOM Method

(View the complete code for this example (pcaex3.sas) in the example repository.)

This example illustrates that the iterative method based on random projection (RANDOM) of Halko, Martinsson, and Tropp (2011) provides a good approximation to the true principal components by using simulated data. The data for this example are the result of multiplying the low-rank matrices **A** and **B** and adding random noise. Matrix **A** is 10,000 rows by 50 columns, and matrix **B** is 50 rows by 2,000 columns, so the final data table consists of 10,000 observations and 2,000 variables.

The following DATA step generates the data:

```
data mylib.testdata;
   title;
   keep x:;
   drop rank number_of_obs number_of_var sigma ii idum;
   drop rv1 rv2 rsq fac row col;
   drop nobs_per_thread nextras start_obs obs j k;
   array B[50,2000];          /* dimensions: rank, number_of_var */
   array A[50];               /* dimension: rank */
   array x[2000] x1-x2000;    /* dimension: number_of_var */

   target_nthreads = min(_nthreads_,96);
   if (_threadid_ = 1) then
```

```
   put "Number of threads = " target_nthreads;
rank=50;
number_of_obs=10000;
number_of_var=2000;
sigma=0.1;

call streaminit(1);

if (_threadid_ <= target_nthreads) then do;
   ii = 0;
   idum = 0;
   do while (ii < rank * number_of_var);
      idum = mod(mod(1664525*idum,4294967296)+1013904223,4294967296);
      rv1 = 2.0*(idum/4294967296)-1.0;
      idum = mod(mod(1664525*idum,4294967296)+1013904223,4294967296);
      rv2 = 2.0*(idum/4294967296)-1.0;
      rsq = rv1*rv1+rv2*rv2;
      if ((rsq < 1.0) and (rsq ^= 0.0)) then do;
         fac = sqrt(-2.0*log(rsq)/rsq);
         row = int(ii/number_of_var)+1;
         col = mod(ii,number_of_var)+1;
         B[row,col] = rv1*fac;
         ii = ii + 1;
         if (ii < rank * number_of_var) then do;
            row = int(ii/number_of_var)+1;
            col = mod(ii,number_of_var)+1;
            B[row,col] = rv2*fac;
            ii = ii + 1;
         end;
      end;
   end;

   nobs_per_thread = int(number_of_obs /target_nthreads);
   nextras = number_of_obs - nobs_per_thread*target_nthreads;
   if (_threadid_ <= nextras) then do;
      nobs_per_thread = nobs_per_thread+1;
      start_obs = (_threadid_-1)*nobs_per_thread+1;
   end;
   else
      start_obs = nextras+(_threadid_-1)*nobs_per_thread+1;

   do obs = start_obs to (start_obs+nobs_per_thread-1);
      do j = 1 to rank;
         A[j] = rand('Normal');
      end;
      do k = 1 to number_of_var;
         x[k] = sigma*rand('Normal');
         do j = 1 to rank;
            x[k] = x[k] + A[j]*B[j,k];
         end;
      end;
      output;
   end;
end;
```

```
run;
```

The following statements use PROC PCA to extract principal components by using the RANDOM and EIG methods. The suboptions for the RANDOM method specify 1, 5, or 10 iterations. Both the EIG and RANDOM methods extract 25 principal components. The DISPLAY statement suppresses the default output except for eigenvalues, and the DISPLAYOUT statement saves the computed eigenvalues to specific output data tables.

```
proc pca data=mylib.testdata n=25 method=random(niter=1 seed=6789);
   var x:;
   display Eigenvalues;
   displayout Eigenvalues=oneiter;
run;

proc pca data=mylib.testdata n=25 method=random(niter=5 seed=6789);
   var x:;
   display Eigenvalues;
   displayout Eigenvalues=fiveiter;
run;

proc pca data=mylib.testdata n=25 method=random(niter=10 seed=6789);
   var x:;
   display Eigenvalues;
   displayout Eigenvalues=teniter;
run;

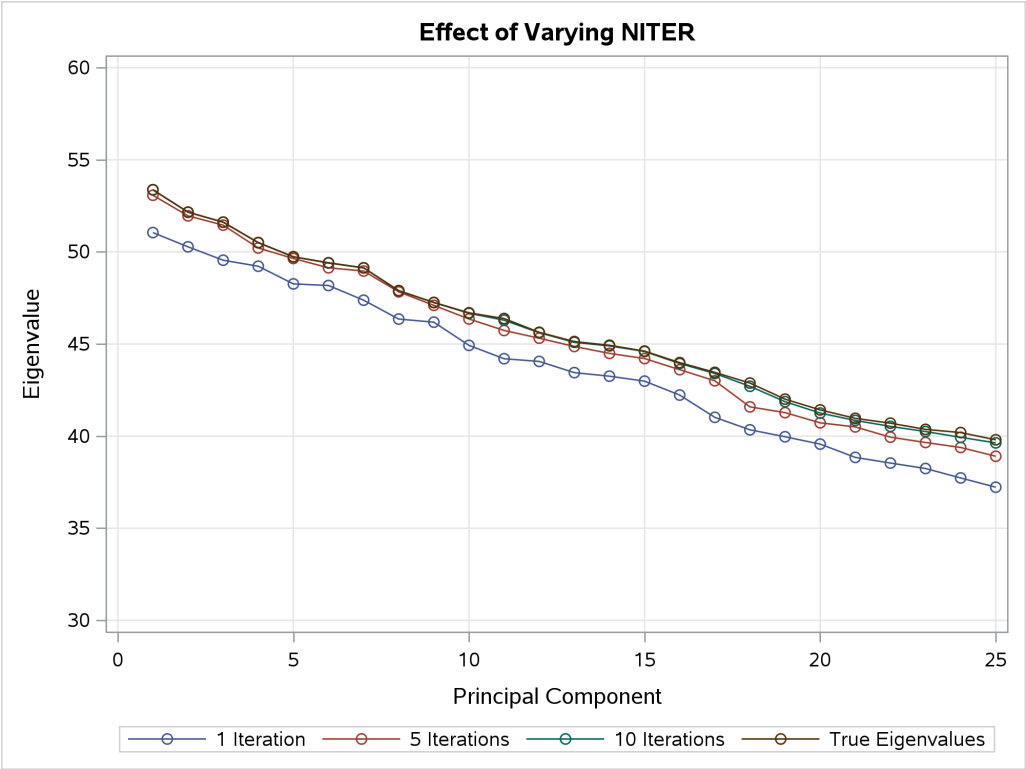
proc pca data=mylib.testdata n=25 method=eig;
   var x:;
   display Eigenvalues;
   displayout Eigenvalues=trueeig;
run;
```

To assess the accuracy of the RANDOM method, compare the estimated eigenvalues to those computed by the EIG method. Using the DATA step and PROC SGPLOT (code not shown), you can combine the estimated eigenvalue data sets with the true eigenvalue data set and plot the results. Output 25.3.1 shows this composite plot and demonstrates the following heuristic rules for the RANDOM method:

- One iteration usually provides a reasonable approximation of the true eigenvalues.
- Three to five iterations usually provide the best trade-off between approximation accuracy and computational cost.
- Beyond five iterations, the improvement in approximation accuracy is usually small compared to the computational cost.

For additional examples that show the impact of the number of iterations for both simulated and real data, see Halko, Martinsson, and Tropp (2011).

Output 25.3.1 RANDOM Method Performance with Varying Number of Iterations



NOTE: The heuristic rules provide guidance for selecting the number of iterations for data tables that are (approximately) low-rank. You might need to increase the number of iterations beyond what is recommended for data tables that are not low-rank.

References

Andrecut, M. (2009). "Parallel GPU Implementation of Iterative PCA Algorithms." *Journal of Computational Biology* 16:1593–1599.

Cooley, W. W., and Lohnes, P. R. (1971). *Multivariate Data Analysis*. New York: John Wiley & Sons.

Gnanadesikan, R. (1977). *Methods for Statistical Data Analysis of Multivariate Observations*. New York: John Wiley & Sons.

Halko, N., Martinsson, P.-G., and Tropp, J. A. (2011). "Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions." *SIAM Review* 53:217–288.

Hotelling, H. (1933). "Analysis of a Complex of Statistical Variables into Principal Components." *Journal of Educational Psychology* 24:417–441, 498–520.

Kshirsagar, A. M. (1972). *Multivariate Analysis*. New York: Marcel Dekker.

Mardia, K. V., Kent, J. T., and Bibby, J. M. (1979). *Multivariate Analysis*. London: Academic Press.

Morrison, D. F. (1976). *Multivariate Statistical Methods*. 2nd ed. New York: McGraw-Hill.

Pearson, K. (1901). "On Lines and Planes of Closest Fit to Systems of Points in Space." *Philosophical Magazine* 6:559–572.

Rao, C. R. (1964). "The Use and Interpretation of Principal Component Analysis in Applied Research." *Sankhyā, Series A* 26:329–358.

Wold, S., Esbensen, K., and Geladi, P. (1987). "Principal Component Analysis." *Chemometrics and Intelligent Laboratory Systems* 2:37–52.

Chapter 26
PHSELECT Procedure

Contents

Overview: PHSELECT Procedure **1332**
PROC PHSELECT Features 1332
Using CAS Sessions and CAS Engine Librefs 1333
Getting Started: PHSELECT Procedure **1334**
Cox Regression 1334
Syntax: PHSELECT Procedure **1339**
PROC PHSELECT Statement 1340
BY Statement 1342
CLASS Statement 1343
CODE Statement 1343
DISPLAY Statement 1345
DISPLAYOUT Statement 1346
EFFECT Statement 1347
FREQ Statement 1349
MODEL Statement 1349
OUTPUT Statement 1351
PARTITION Statement 1354
SELECTION Statement 1354
STRATA Statement 1356
WEIGHT Statement 1356
Details: PHSELECT Procedure **1356**
Missing Values 1356
Cox Proportional Hazards Regression Model 1357
Partial Likelihood for the Cox Model 1357
The LASSO Method of Model Selection 1358
Model Fit and Assessment Statistics 1358
Predicted Values and Regression Diagnostics 1359
Joint Tests and Type 3 Tests 1361
Multithreading 1362
Optimization Algorithms 1362
Displayed Output 1363
ODS Table Names 1366
ODS Graphics 1368
Examples: PHSELECT Procedure **1369**
Example 26.1: Model Selection 1369
Example 26.2: Stratified Analysis 1373
References **1375**

Overview: PHSELECT Procedure

The PHSELECT procedure fits the Cox proportional hazards regression models for survival data and performs variable selection in SAS Viya.

The models that PROC PHSELECT supports can contain main effects that consist of both continuous and classification variables and interaction effects of these variables. The models can also include constructed effects such as splines. The procedure offers a number of effect-selection methods, including stepwise methods and modern LASSO methods. It also offers extensive capabilities for customizing the model selection by using a wide variety of selection and stopping criteria, from computationally efficient significance-level-based criteria to modern, computationally intensive validation-based criteria. PROC PHSELECT also provides a variety of Cox regression diagnostics that are conditional on the selected model.

PROC PHSELECT Features

The PHSELECT procedure estimates the parameters of a Cox regression model by maximizing the partial likelihood. It also does the following:

- provides model-building syntax through the CLASS, EFFECT, and effect-based MODEL statements, which are familiar from SAS/STAT analytic procedures (in particular, the GLM, LOGISTIC, GLIMMIX, and PHREG procedures)
- enables model building (variable selection) through the SELECTION statement
- enables you to split classification effects into individual components by using the SPLIT option in the CLASS statement
- permits any degree of interaction effects that involve classification and continuous variables
- provides a WEIGHT statement for weighted analysis
- provides a FREQ statement for grouped analysis
- provides a STRATA statement for stratified analysis
- provides a CODE statement to produce SAS code that can score a new data set
- provides an OUTPUT statement to produce a data table that contains predicted values and other observationwise statistics
- uses ODS Graphics to create model selection plots as part of its output. For more information about ODS Graphics, see the section "ODS Graphics" on page 1368.

Because the PHSELECT procedure runs on CAS, it also does the following:

- enables you to run on a cluster of machines that distribute the data and the computations
- enables you to run in single-machine mode
- exploits all the available cores and concurrent threads. For information about how PROC PHSELECT uses threads, see the section "Multithreading" on page 102 in Chapter 2, "Shared Concepts."

PROC PHSELECT Compared with the PHREG Procedure

The PHSELECT procedure provides the full set of parameterizations that are available in the PHREG procedure. The PHSELECT procedure uses the GLM parameterization for the CLASS variables by default. The PHREG procedure uses the REF parameterization for the CLASS variables by default. In either procedure, you can use the PARAM= option in the CLASS statement to change the parameterization.

The PHSELECT procedure uses a quasi-Newton algorithm by default. You can choose different optimization techniques, including second-order methods that require a crossproducts matrix, by using the TECHNIQUE= option in the PROC PHSELECT statement. The PHREG procedure uses the Newton-Raphson optimization algorithm exclusively.

The PHREG procedure can fit a variety of models, but it has very limited model selection capability. The PHSELECT procedure provides a richer set of model selection options, including LASSO selection, information-criterion-based selection and stopping criteria, and validation-based criteria.

The PHSELECT procedure uses ODS Graphics to display plots that help interpret the selection process; the PHREG procedure use ODS graphics to display predicted survival curves.

The PHSELECT procedure is specifically designed to operate in SAS Viya and performs computations in multiple threads. The PHREG procedure executes in a single thread on a single machine.

Using CAS Sessions and CAS Engine Librefs

SAS Cloud Analytic Services (CAS) is the analytic server and associated cloud services in SAS Viya. This section describes how to create a CAS session and set up a CAS engine libref that you can use to connect to the CAS session. It assumes that you have a CAS server already available; contact your system administrator if you need help starting and terminating a server. This CAS server is identified by specifying the host on which it runs and the port on which it listens for communications. To simplify your interactions with this CAS server, the host information and port information for the server are stored as SAS option values that are retrieved automatically whenever this CAS server needs to be accessed. You can examine the host and port values for the server at your site by using the following statements:

```
proc options option=(CASHOST CASPORT);
run;
```

In addition to starting a CAS server, your system administrator might also have created a CAS session and a CAS engine libref for your use. You can define your own sessions and CAS engine librefs that connect to the CAS server as shown in the following statements:

```
cas mysess;
libname mylib cas sessref=mysess;
```

The CAS statement creates the CAS session named mysess, and the LIBNAME statement creates the mylib CAS engine libref that you use to connect to this session. It is not necessary to explicitly name the CASHOST and CASPORT of the CAS server in the CAS statement, because these values are retrieved from the corresponding SAS option values.

If you have created the mysess session, you can terminate it by using the TERMINATE option in the CAS statement as follows:

```
cas mysess terminate;
```

For more information about the CAS and LIBNAME statements, see the section "Introduction to Shared Concepts" on page 6 in Chapter 2, "Shared Concepts."

Getting Started: PHSELECT Procedure

Cox Regression

(View the complete code for this example at https://github.com/sascommunities/doc-supplement-casstat/tree/main/Examples/coxgs1.sas.)

NOTE: Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections "Using CAS Sessions and CAS Engine Librefs" on page 7 and "Loading a SAS Data Set onto a CAS Server" on page 7 in Chapter 2, "Shared Concepts."

The following DATA step creates the data table mylib.getStarted in your CAS session. This data table consists of 100 observations on a failure time variable (Time), an indicator variable (Status) that has two values (0 for censored observations and 1 for event observations), three classification variables (C1–C3), and four continuous variables (X1–X4). This DATA step assumes that your libref is named mylib, but you can substitute any appropriately defined libref.

```
data mylib.getStarted;
   input Time Status C1$ C2 C3$ X1-X4;
   datalines;
   53       0      Low          1    M     1.11    2.000    3.6128    12.0
   12       0      High         1    M     1.40    1.362    3.8388     8.8
   11       1      Low          1    F     1.57    1.672    3.8865     7.5
    7       1      Medium       0    M     1.04    2.000    3.7324     5.1
    2       1      Low          1    M     1.52    2.000    3.8751     9.8
   41       0      High         1    M     1.76    1.447    3.7243    12.8
    6       1      Critical     1    F     1.36    1.462    3.5441     9.0
    6       1      Critical     1    M     1.42    1.690    3.9294    10.4
   16       1      Medium       1    M     1.32    0.699    3.6990     8.8
   41       1      Medium       1    M     1.00    1.477    3.4771    10.2
    2       1      High         0    M     1.30    2.000    3.7243     5.1
   58       1      Medium       1    M     1.20    1.580    3.6990    12.1
   11       1      High         1    M     1.08    1.903    3.5051     9.6
   12       0      Critical     1    F     1.15    1.146    3.6435    11.6
   16       0      High         1    F     1.15    0.903    3.8573    13.0
   54       1      Medium       1    M     1.26    1.699    3.7243     9.0
   51       1      Low          0    M     1.57    1.041    3.4150     7.7
   67       1      Medium       1    M     1.32    1.041    3.6435    12.8
    1       1      Low          1    M     1.94    1.954    3.9868    12.0
   19       0      Medium       1    M     1.32    2.000    3.7709    13.0
    1       1      Medium       1    M     2.22    1.954    3.6628     9.4
   35       1      Medium       0    M     1.11    1.176    3.6532     7.0
```

41	1	High	1	M	1.15	1.342	3.5185	5.0
58	1	Medium	1	M	1.20	1.580	3.6990	12.1
11	1	Medium	1	M	1.11	1.279	3.8808	14.0
41	1	Medium	1	M	1.00	1.477	3.4771	10.2
19	1	High	0	M	1.26	1.929	3.7924	7.5
89	1	High	1	M	1.32	1.623	3.6532	14.0
4	0	Medium	1	F	1.95	0.778	4.0453	10.2
6	1	Critical	1	F	1.36	1.462	3.5441	9.0
57	0	Low	1	F	1.26	1.954	3.9685	12.5
5	1	High	1	M	2.24	1.663	4.9542	10.1
17	1	Medium	1	F	1.59	1.613	3.4314	11.2
77	0	Low	1	F	1.08	0.954	3.6812	14.0
66	1	High	1	M	1.45	1.820	3.7853	6.6
16	1	Medium	1	M	1.32	0.699	3.6990	8.8
8	0	Critical	1	M	1.08	1.653	3.8325	9.9
19	1	High	0	M	1.26	1.929	3.7924	7.5
37	1	High	1	F	1.60	1.204	3.9542	11.0
52	1	Medium	1	M	1.00	1.653	3.8573	10.1
13	0	High	0	F	1.66	1.792	3.6435	4.9
3	1	Medium	1	F	1.54	1.935	4.4757	6.7
51	1	Low	0	M	1.57	1.041	3.4150	7.7
2	1	High	0	M	1.30	2.000	3.7243	5.1
25	1	Medium	1	M	1.00	1.644	3.8195	12.4
11	1	Low	1	F	1.57	1.672	3.8865	7.5
19	1	Low	1	M	1.08	2.000	3.9191	14.4
9	1	Low	1	M	1.72	1.740	3.7993	8.2
6	1	Medium	1	M	1.11	1.398	3.5185	9.7
16	1	Medium	1	M	1.34	2.000	3.9345	9.0
12	0	High	1	M	1.40	1.362	3.8388	8.8
17	1	High	1	M	1.23	1.447	3.8808	10.0
17	1	High	1	M	1.23	1.447	3.8808	10.0
41	0	High	1	M	1.76	1.447	3.7243	12.8
88	1	High	1	F	1.18	1.756	3.5563	10.6
16	0	High	1	F	1.15	0.903	3.8573	13.0
4	0	Low	1	F	1.92	1.623	3.9590	10.0
7	1	Low	1	M	1.18	1.519	3.7243	11.4
19	0	Medium	1	M	1.32	1.519	3.8808	10.8
8	0	Critical	1	M	1.08	1.653	3.8325	9.9
57	0	Low	1	F	1.26	1.954	3.9685	12.5
7	1	Medium	1	M	1.98	1.568	3.3617	9.5
19	1	Low	1	M	1.08	2.000	3.9191	14.4
7	0	Low	1	F	1.53	1.881	3.5911	10.2
5	1	High	1	F	1.68	1.732	3.7324	6.5
2	1	Low	0	M	1.75	1.255	3.8062	11.3
1	1	Low	1	M	1.94	1.954	3.9868	12.0
15	1	Low	1	M	1.60	1.431	3.6902	10.6
26	1	Low	1	M	1.23	2.000	3.6021	11.2
92	1	Low	1	M	1.43	1.415	4.0755	11.0
11	1	Medium	1	M	1.30	1.820	3.7993	13.2
3	1	Medium	1	F	1.54	1.935	4.4757	6.7
66	1	High	1	M	1.45	1.820	3.7853	6.6
1	1	Medium	1	M	2.22	1.954	3.6628	9.4
11	1	Medium	1	M	1.30	1.820	3.7993	13.2
14	1	Medium	1	M	1.40	1.255	3.7243	14.6

```
   32      1      High        1    M     1.32    1.634    3.6990    10.6
   24      1      High        1    M     1.30    0.477    4.0899    14.6
   18      1      Critical    1    F     1.45    0.903    3.5682     7.5
    5      1      High        1    F     1.68    1.732    3.7324     6.5
    2      1      Low         0    M     1.75    1.255    3.8062    11.3
   26      1      Low         1    M     1.23    2.000    3.6021    11.2
   11      1      High        1    M     1.23    1.176    3.7709    12.0
   28      0      High        1    M     1.23    1.672    3.7482     7.3
   19      0      Medium      1    M     1.32    2.000    3.7709    13.0
   54      1      Medium      1    M     1.26    1.699    3.7243     9.0
    2      1      Low         1    M     1.52    2.000    3.8751     9.8
    7      0      Medium      1    M     1.11    1.857    3.7993    12.4
   19      0      Medium      1    M     1.32    1.519    3.8808    10.8
    6      1      Critical    1    M     1.42    1.690    3.9294    10.4
    6      1      Medium      1    M     1.11    1.398    3.5185     9.7
   11      1      Medium      1    M     1.11    1.279    3.8808    14.0
    6      1      Critical    0    M     2.11    1.362    3.5441    10.2
   13      1      Medium      0    M     0.78    1.398    3.5798     5.5
   18      1      Critical    1    F     1.45    0.903    3.5682     7.5
    7      0      Low         1    F     1.53    1.881    3.5911    10.2
   53      0      Low         1    M     1.11    2.000    3.6128    12.0
   11      1      High        1    M     1.08    1.903    3.5051     9.6
   11      0      High        1    M     1.61    1.845    3.7324    14.0
   89      1      High        1    M     1.32    1.623    3.6532    14.0
;
```

The following statements fit a Cox proportional hazards model to these data by using three classification effects for the variables C1–C3 and four regressor effects for the variables X1–X4. The ITHIST option displays a table that summarizes the steps of the optimization.

```
proc phselect data=mylib.getStarted ithist;
   class C1-C3;
   model Time*Status(0) = C1-C3 X1-X4;
run;
```

The output from this analysis is presented in Figure 26.1 through Figure 26.8.

Figure 26.1 displays the "Model Information" table. The variable Time is the failure time variable. The variable Status is the censoring variable; the value 0 indicates censored observations. The PHSELECT procedure uses a quasi-Newton algorithm to maximize the partial likelihood to estimate the regression coefficients.

Figure 26.1 Model Information

The PHSELECT Procedure

Model Information	
Data Source	GETSTARTED
Response Variable	Time
Censoring Variable	Status
Censoring Values	0
Optimization technique	Dual Quasi-Newton

Figure 26.2 displays the "Number of Observations" table. All 100 observations in the data table are used in

the analysis; of them, 26 are censored and 74 are uncensored.

Figure 26.2 Number of Observations

Number of Observations			
Description	**Total**	**Event**	**Censored**
Number of Observations Read	100	74	26
Number of Observations Used	100	74	26

The classification variables C1–C3 are parameterized using the GLM parameterization, which is the default. The variable C1 has four unique formatted levels; each of the two variables C2 and C3 has two levels. The classification levels are displayed in the "Class Level Information" table in Figure 26.3.

Figure 26.3 Class Level Information

Class Level Information		
Class	**Levels**	**Values**
C1	4	Critical High Low Medium
C2	2	0 1
C3	2	F M

The "Iteration History" table is shown in Figure 26.4. The quasi-Newton algorithm converged after 12 iterations, not counting the initial setup iteration.

Figure 26.4 Iteration History

Iteration History				
Iteration	**Evaluations**	**Objective Function**	**Change**	**Max Gradient**
0	4	272.56284984	.	65.17039
1	3	267.70594206	4.85690778	7.86724
2	2	261.80868056	5.89726150	4.723085
3	2	256.62059571	5.18808485	4.650619
4	2	255.59346125	1.02713446	1.0462
5	2	255.44142594	0.15203531	1.018105
6	3	255.40692196	0.03450398	0.353611
7	3	255.39461736	0.01230460	0.415741
8	3	255.39362671	0.00099065	0.074411
9	3	255.39339211	0.00023460	0.062435
10	3	255.39336653	0.00002558	0.004922
11	3	255.39336262	0.00000391	0.002273
12	3	255.39336237	0.00000026	0.000452

Figure 26.5 displays the final convergence status of the quasi-Newton algorithm. The GCONV=1E-8 convergence criterion is satisfied.

Figure 26.5 Convergence Status

Convergence criterion (GCONV=1E-8) satisfied.

Figure 26.6 displays the "Dimensions" table for this model. This table summarizes some important sizes of various model components. For example, it shows that the design matrix **X** has 12 columns: 4 columns for the effects that are associated with the classification variable C1, 2 columns for each of the classification variables C2 and C3, and 1 column for each of the continuous variables X1–X4. However, the rank of the crossproducts matrix is only 9. Because the classification variables C1–C3 use GLM parameterization, there is one singularity in the crossproducts matrix of the model for each classification variable. Consequently, only nine parameters enter the optimization.

Figure 26.6 Dimensions in Cox Regression

Dimensions	
Number of Effects	7
Max Effect Columns	4
Columns in Design	12
Rank of Design	9

The "Fit Statistics" table is shown in Figure 26.7. The –2 log likelihood at the converged estimates is 510.78672. You can use this value to compare the model to nested model alternatives by means of a likelihood ratio test. To compare models that are not nested, you can use information criteria such as Akaike's information criterion (AIC), Akaike's bias-corrected information criterion (AICC), and the Schwarz Bayesian information criterion (BIC). These criteria penalize the –2 log partial likelihood for the number of parameters.

Figure 26.7 Fit Statistics

Fit Statistics	
-2 Log Likelihood	510.78672
AIC (smaller is better)	528.78672
AICC (smaller is better)	531.59922
SBC (smaller is better)	549.52331

The "Parameter Estimates" table in Figure 26.8 shows that many parameters have fairly large *p*-values, indicating that one or more of the model effects might not be necessary.

Figure 26.8 Parameter Estimates

Parameter Estimates					
Parameter	**DF**	**Estimate**	**Standard Error**	**Chi-Square**	**Pr > ChiSq**
C1 Critical	1	0.400493	0.489562	0.6692	0.4133
C1 High	1	-0.990602	0.325145	9.2821	0.0023
C1 Low	1	-0.665495	0.339148	3.8504	0.0497
C1 Medium	0	0	.	.	.
C2 0	1	0.410704	0.398833	1.0604	0.3031
C2 1	0	0	.	.	.
C3 F	1	-0.172550	0.337940	0.2607	0.6096
C3 M	0	0	.	.	.
X1	1	1.970488	0.524142	14.1335	0.0002
X2	1	0.726605	0.415141	3.0634	0.0801
X3	1	0.725390	0.585658	1.5341	0.2155
X4	1	-0.131226	0.057103	5.2810	0.0216

Finally, the procedure displays the table in Figure 26.9, which shows the amount of time (in seconds) that PROC PHSELECT required to perform different tasks in the analysis.

Figure 26.9 Procedure Timing

Task Timing		
Task	**Seconds**	**Percent**
Setup and Parsing	0.02	9.03%
Levelization	0.02	8.67%
Model Initialization	0.01	2.18%
SSCP Computation	0.00	0.79%
Model Fitting	0.19	77.72%
Cleanup	0.00	0.46%
Total	0.24	100.00%

Syntax: PHSELECT Procedure

The following statements are available in the PHSELECT procedure:

PROC PHSELECT < *options* > **;**
BY *variables* **;**
CLASS *variable* < **(***options***)** > . . . < *variable* < **(***options***)** > > < / *global-options* > **;**
CODE < *options* > **;**
DISPLAY < *table-list* > < / *options* > **;**
DISPLAYOUT *table-spec-list* < / *options* > **;**
EFFECT *name***=***effect-type***(***variables* < / *options* >**)** **;**
FREQ *variable* **;**
MODEL *response* < ∗ *censor***(***list***)** > **=** < *effects* > < / *model-options* > **;**
OUTPUT OUT=*libref.data-table* < *options* > < *keyword* < **=***name* > > . . . < *keyword* < **=***name* > > **;**
PARTITION *partition-options* **;**
SELECTION < **METHOD=***method* < **(***method-options***)** > >< *options* > **;**
STRATA *variable* < **MISSING** > **;**
WEIGHT *variable* **;**

The PROC PHSELECT statement and the MODEL statement are required. The CLASS statement can appear multiple times. If you use a CLASS statement, it must precede the MODEL statements.

PROC PHSELECT Statement

PROC PHSELECT < *options* > **;**

The PROC PHSELECT statement invokes the procedure. Table 26.1 summarizes the available *options* in the PROC PHSELECT statement by function. They are then described fully in alphabetical order.

Table 26.1 PROC PHSELECT Statement Options

Option	Description
ALPHA=	Specifies a global significance level
DATA=	Specifies the CAS input data table
MULTIPASS=	Specifies whether to levelize the data table every time it is read
Output Options	
CORRB	Displays the "Parameter Estimates Correlation Matrix" table
COVB	Displays the "Parameter Estimates Covariance Matrix" table
ITHIST	Displays the "Iteration History" table
LOGLIKENULL	Displays the "–2 Log Likelihood for Null Model" table
NOCLPRINT	Limits or suppresses the display of class levels
NOSTDERR	Suppresses computation of the covariance matrix and standard errors
Optimization Options	
ABSCONV=	Tunes the absolute function convergence criterion
ABSFCONV=	Tunes the absolute function difference convergence criterion
ABSGCONV=	Tunes the absolute gradient convergence criterion
ABSXCONV=	Tunes the absolute parameter convergence criterion
FCONV=	Tunes the relative function difference convergence criterion
FCONV2=	Tunes the second relative function difference convergence criterion
GCONV=	Tunes the relative gradient convergence criterion
GCONV2=	Tunes the second relative gradient convergence criterion
XCONV=	Tunes the relative gradient convergence criterion
HESSIAN	Uses analytic Hessian instead of finite-difference Hessian
MAXFUNC=	Specifies the maximum number of function evaluations in any optimization
MAXITER=	Specifies the maximum number of iterations in any optimization
MAXTIME=	Specifies the upper limit of CPU time (in seconds) for any optimization
MINITER=	Specifies the minimum number of iterations in any optimization
TECHNIQUE=	Selects the optimization technique
LASSO Options	
LASSORHO=	Specifies the base regularization parameter for the LASSO method
LASSOSTEPS=	Specifies the maximum number of steps for the LASSO method
LASSOTOL=	Specifies the convergence criterion for the LASSO method

The optimization options, with the exception of the HESSIAN option, are fully described in the section "Optimization Options" on page 57 in Chapter 2, "Shared Concepts." The following list describes the other options available in the PROC PHSELECT statement:

ALPHA=*number*

specifies a global significance level for the construction of confidence intervals. The confidence level is 1–*number*. The value of *number* must be between 0 and 1. By default, ALPHA=0.05.

CORRB

creates the "Parameter Estimates Correlation Matrix" table. The correlation matrix is computed by normalizing the covariance matrix $\mathbf{\Sigma}$. That is, if σ_{ij} is an element of $\mathbf{\Sigma}$, then the corresponding element of the correlation matrix is $\sigma_{ij}/\sigma_i\sigma_j$, where $\sigma_i = \sqrt{\sigma_{ii}}$.

COVB

creates the "Parameter Estimates Covariance Matrix" table. The covariance matrix is computed as the inverse of the negative Hessian matrix, which is the matrix of second derivatives of the log-likelihood function with respect to the model parameters.

DATA=*libref.data-table*

names the input data table for PROC PHSELECT to use. The default is the most recently created data table. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to the data, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 1333.

data-table specifies the name of the input data table.

HESSIAN

computes the Hessian matrix by using the analytic expression for the second-order derivatives of the log partial likelihood function instead of using the finite-difference method. When you specify this option, the optimization technique defaults to the Newton-Raphson method with ridging (TECH=NRRIDG), but you can use the TECH= option to specify the technique of your choice. The HESSIAN option requires a large amount of memory for each machine in the cluster.

ITHIST

generates the "Iteration History" table.

LASSORHO=*r*

specifies the base regularization parameter for the LASSO model selection method. The regularization parameter for step i is r^i. By default, LASSORHO=0.8.

LASSOSTEPS=*n*

specifies the maximum number of steps for LASSO model selection. By default, LASSOSTEPS=20.

LASSOTOL=*r*

specifies the convergence tolerance for the optimization algorithm that solves for the LASSO parameter estimates at each step of LASSO model selection. By default, LASSOTOL=1E–6.

LOGLIKENULL
creates the "–2 Log Likelihood for the Null Model" table. If you also specify the PARTITION statement, the table displays the –2 log likelihood for the null model for each data partition.

MULTIPASS=TRUE | FALSE
specifies when to levelize the data table—that is, when to generate the design rows for your model from the data. This option does not affect how the OUTPUT statement is processed. You can specify the following values:

TRUE levelizes the data every time they are read. This method is slower but uses less memory.

FALSE levelizes the data once and stores them in a temporary data table, then rereads the data from that table as needed. This method is faster but uses more memory.

By default, MULTIPASS=FALSE.

NOCLPRINT< =*number* **>**
suppresses the display of the "Class Level Information" table if you do not specify *number*. If you specify *number*, the values of the classification variables are displayed for only those variables whose number of levels is less than *number*. Specifying *number* helps reduce the size of the "Class Level Information" table if some classification variables have a large number of levels.

NOSTDERR
suppresses computation of the covariance matrix and the standard errors of the regression coefficients. When the model contains many variables (such as thousands), inverting the Hessian matrix to derive the covariance matrix and the standard errors of the regression coefficients can be time-consuming. The CORRB, COVB, and TYPE3 options are not available when you specify this option.

BY Statement

BY *variables* **;**

You can specify a BY statement in PROC PHSELECT to obtain separate analyses of observations in groups that are defined by the values of the BY variables. If you specify more than one BY statement, only the last one specified is used. For more information, see the discussion of BY-group processing in the "Grouping Data" section of *SAS Programmers Guide: Essentials*.

CLASS Statement

CLASS *variable* < **(***options***)** >. . . < *variable* < **(***options***)** > > < / *global-options* > **;**

The CLASS statement names the classification variables to be used as explanatory variables in the analysis. You can list the response variable for binary models in the CLASS statement, but this is not required. Table 26.2 summarizes the values that you can use for either an *option* or a *global-option*. The options are fully documented in the section "CLASS Statement" on page 9 in Chapter 2, "Shared Concepts."

Table 26.2 CLASS Statement Options

Option	Description
DESCENDING	Reverses the sort order
MISSING	Treats missing values as valid levels
ORDER=	Specifies the sort order for the levels
PARAM=	Specifies the parameterization of the variable
REF=	Specifies the reference level of the variable
SPLIT	Splits levels of CLASS variables into independent effects

CODE Statement

CODE < *options* > **;**

The CODE statement writes SAS DATA step code for computing predicted values of the fitted model to a file or to a table. The predicted values include survival probabilities and cumulative hazards at specific time points. To score new data, you can then include the file in a DATA step, or you can specify the table in the **`runCodeTable`** action in the **`dataStep`** action set (for more information, see *SAS Viya: System Programming Guide*).

Table 26.3 summarizes the *options* available in the CODE statement.

Table 26.3 CODE Statement Options

Option	Description
COMMENT	Adds comments to the generated code
CUMHAZ	Generates cumulative hazard values at specified time points
FILE=	Names the file in which to save the generated code
FORMATWIDTH=	Specifies the numeric format width for the regression coefficients
INDENTSIZE=	Specifies the number of spaces to indent the generated code
LABELID=	Specifies a number used to construct names and labels
LINESIZE=	Specifies the line size for the generated code
NOSURVIVAL	Suppresses the generation of survival probabilities
NOTRIM	Compares formatted values, including blank padding
OUT=	Names an output table in which to save the generated code
SHOWTIME	Creates variables that contain the time points at which predictions are made
TIMEPOINT=	Specifies the time points at which survival probabilities or cumulative hazards are predicted

The following CODE statement *options* are particularly important:

CUMHAZ
generates SAS code to predict the cumulative hazard function at the time points that you specify in the TIMEPOINT= option.

FILE=*filename*
names the external file that saves the generated code. When enclosed in a quoted string (for example, FILE="c:\mydir\scorecode.sas"), this option specifies the path and filename for writing the code to an external file. If you do not specify a path but your SAS client has a default path, then the code is written to an external file named *filename* at that location. You can also specify an unquoted *filename* of no more than eight characters. If the *filename* is assigned as a fileref in a Base SAS FILENAME statement, the file that is specified in the FILENAME statement is opened; otherwise, if your SAS client has a default path, an external file named *filename* is created. This option cannot be specified together with the OUT= option.

NOSURVIVAL
suppresses SAS code for survival probabilities prediction. If you do not specify this option, PROC PHSELECT generates SAS code to predict survival probabilities at the time points that you specify in the TIMEPOINT= option.

SHOWTIME
creates variables that contain the time points at which predictions are made. If T is the name of the failure time variable, these time point variables are named T_1, T_2, and so on.

TIMEPOINT=*list* | **QUANTILE(***probability-list***)**

TIME=*list* | **QUANTILE(***probability-list***)**
specifies the time points at which survival probabilities or cumulative hazards are predicted. You can specify a list of numbers that represent exact time points. For example:

```
code cumhaz timepoint = 40 to 60 by 10;
```

If T is the name of the failure time variable, the preceding specification requests the predicted cumulative hazard and survival probability at the time points T=40, T=50, and T=60. If you also specify the SHOWTIME option, these time points are saved in the variables T_1, T_2, and T_3, respectively. The predicted variables are named and labeled as shown in Table 26.4.

Table 26.4 Predicted Variables

T	Name	Label
40	C_T_1	Cumulative Hazard at T_1
50	C_T_2	Cumulative Hazard at T_2
60	C_T_3	Cumulative Hazard at T_3
40	S_T_1	Survival Probability at T_1
50	S_T_2	Survival Probability at T_2
60	S_T_3	Survival Probability at T_3

Alternatively, you can use the keyword QUANTILE to specify a list of quantile probabilities. For example:

```
code timepoint = quantile(.2, .5, .8);
```

The PHSELECT procedure computes the 20th, 50th, and 80th percentiles from the Kaplan-Meier curve (or the Breslow curve if you also specify the ENTRY= option in the MODEL statement) and use them as the time points for the prediction.

By default, TIMEPOINT=QUANTILE(0.25, 0.50, 0.75).

For more information about the syntax of the CODE statement, see the section "CODE Statement" on page 13 in Chapter 2, "Shared Concepts."

DISPLAY Statement

DISPLAY < *table-list* > < / *options* > **;**

The DISPLAY statement enables you to specify a list of display tables to display or exclude. This statement is similar to the ODS SELECT, ODS EXCLUDE, and ODS TRACE statements. However, the DISPLAY statement can improve performance when a large number of tables could be generated (such as in BY-group processing). The procedure processes the DISPLAY statement on a CAS server and thus sends only a subset of ODS tables to the SAS client. Because ODS statements are processed on a SAS client, first all the generated display tables are sent to the client, and then the client creates a subset.

If you use both DISPLAY and ODS statements together, the DISPLAY statement takes precedence over the ODS statements. Note that the ODS EXCLUDE statement processes tables that are sent to the client after they have been filtered by the DISPLAY statement. In some cases, it might appear that the ODS EXCLUDE

statement is taking precedence because it can further filter the tables. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

You can specify the *table-list* as a list of table names, paths, partial pathnames, and regular expressions.

The table names that you can specify are listed in the section "ODS Table Names" on page 1366. A path is a table name that is prefixed with dot-separated grouping information. For example, a SelectionSummary table that a procedure produces during a selection routine might have the path *Bygroup1.Summary.SelectionSummary*. A partial pathname does not include all groups; for example, *SelectionSummary* and *Summary.SelectionSummary* are partial pathnames for *Bygroup1.Summary.SelectionSummary*.

When you specify a table name or partial pathname, all display tables whose paths end in the specified name are selected for display or exclusion. For example, both *SelectionSummary* and *Summary.SelectionSummary* select *Bygroup1.Summary.SelectionSummary*.

A regular expression is enclosed in forward slashes (/). For example, specifying "/tions/" selects all pathnames that contain the substring "tions"; in particular, the *Bygroup1.Summary.SelectionSummary* table is selected. Specifying "!/tions/" selects all pathnames that do not contain the substring "tions"; in particular, the *Bygroup1.Summary.SelectionSummary* table is not selected.

You can specify the following *options* after a slash (/):

CASESENSITIVE
performs a case-sensitive comparison of table names in the *table-list* to display table names when tables are subsetted for display. To preserve case, you must enclose table names in the *table-list* in quotation marks.

EXCLUDE
displays all display tables except those that you specify in the *table-list*.

EXCLUDEALL
suppresses display of all tables. This option takes precedence over the other options.

TRACE
displays the display table names, labels, and paths.

DISPLAYOUT Statement

DISPLAYOUT *table-spec-list* < / *options* > **;**

The DISPLAYOUT statement enables you to create output tables from your displayed output. This statement is similar to the ODS OUTPUT statement. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

The *table-spec-list* specifies a list of output tables to create. Each entry in the list has either a *key*=*value* format or a *key* format:

key=*value* specifies *key* as the ODS table name, path, or partial pathname, and specifies *value* as the output table name.

key specifies *key* as the ODS table name and also as the output table name.

The ODS table names that you can specify are listed in the section "ODS Table Names" on page 1366. You cannot specify the ODS table named OutputCasTables in the *table-spec-list*.

Table names and partial pathnames are discussed under the DISPLAY statement. The DISPLAYOUT statement does not support regular expressions.

You can specify the following *options* after a slash (/):

INCLUDEALL
creates output tables for all display tables. The name of the created output table is the same as the name of the corresponding display table. If you specify this option, the *table-spec-list* specification is ignored.

NOREPLACE
does not replace any existing output table of the same name.

REPEATED
replicates all output tables on all nodes.

EFFECT Statement

EFFECT *name*=*effect-type* **(***variables* < / *options* >**) ;**

The EFFECT statement enables you to construct special collections of columns for design matrices. These collections are referred to as *constructed effects* to distinguish them from the usual model effects that are formed from continuous or classification variables, as discussed in the section "GLM Parameterization of Classification Variables and Effects" on page 67 in Chapter 2, "Shared Concepts."

You can specify the following *effect-types*:

COLLECTION specifies a collection effect that defines one or more variables as a single effect that has multiple degrees of freedom. The variables in a collection are considered as a unit for purposes of estimation and inference.

MULTIMEMBER | MM specifies a multimember classification effect whose levels are determined by one or more variables that appear in a CLASS statement.

POLYNOMIAL | POLY specifies a multivariate polynomial effect in the specified numeric variables.

SPLINE specifies a regression spline effect whose columns are univariate spline expansions of one or more variables. A spline expansion replaces the original variable with an expanded or larger set of new variables.

Table 26.5 summarizes the *options* available in the EFFECT statement.

Table 26.5 EFFECT Statement Options

Option	Description
Collection Effects Options	
DETAILS	Displays the constituents of the collection effect

Table 26.5 *continued*

Option	Description
Multimember Effects Options	
DETAILS	Displays the levels of the multimember effect
NOEFFECT	Specifies that observations whose levels are all missing for the multimember variables should have 0 values in the corresponding design matrix columns
STDIZE	Standardizes the design matrix entries so that each observation has a sum of 1
WEIGHT=	Specifies the weight variable for the contributions of each classification effect
Polynomial Effects Options	
DEGREE=	Specifies the degree of the polynomial
DETAILS	Displays details of the specified polynomial
MDEGREE=	Specifies the maximum degree of any variable in a term of the polynomial
NOSEPARATE	Treats the polynomial as a single effect with multiple degrees of freedom
STANDARDIZE=	Specifies centering and scaling suboptions for the variables that define the polynomial
Spline Effects Options	
BASIS=	Specifies the type of basis (B-spline basis or truncated power function basis) for the spline effect
DATABOUNDARY	Uses the extremes of the data as boundary knots for a B-spline basis
DEGREE=	Specifies the degree of the spline effect
DETAILS	Displays the knots and locations for each spline basis function
KNOTMAX=	Requests equally spaced right-side boundary knots starting at the variables' maximum and ending at the KNOTMAX= value
KNOTMETHOD=	Specifies how to construct the knots for the spline effect
KNOTMIN=	Requests equally spaced left-side boundary knots starting at the KNOTMIN= value and ending at the variables' minimum value
NATURALCUBIC	Specifies a natural cubic spline basis for the spline effect
SEPARATE	Treats the spline basis for each variable as a separate effect when multiple variables are specified
SPLIT	Treats each design matrix column as a separate effect for selection methods

For more information about the syntax of these *effect-types* and how columns of constructed effects are computed, see the section "EFFECT Statement" on page 18 in Chapter 2, "Shared Concepts."

FREQ Statement

FREQ *variable* **;**

The *variable* in the FREQ statement identifies a numeric variable in the input data table that contains the frequency of occurrence of each observation. PROC PHSELECT treats each observation as if it appears f times, where f is the value of the FREQ *variable* for the observation. If f is not an integer, it is truncated to an integer. If f is less than 1 or missing, the observation is not used in the analysis. When you omit the FREQ statement, each observation is assigned a frequency of 1.

MODEL Statement

MODEL *response* < ∗ *censor* **(***list***)** > **=** *effects* < / *options* > **;**

The MODEL statement identifies the variables to be used as the failure time variable, the optional censoring variable, and the explanatory effects, including covariates, main effects, interactions, and nested effects.

The name of the failure time variable precedes the equal sign. This name can optionally be followed by an asterisk, the name of the censoring variable, and a list of censoring values (separated by blanks or commas) enclosed in parentheses.

The censoring variable must be numeric, and the failure time variables must contain nonnegative values. Any observation that has a negative failure time is excluded from the analysis, as is any observation that has a missing value for any of the variables listed in the MODEL statement. Failure time variables in SAS date format are not recommended, because the dates might be translated into negative numbers and consequently the corresponding observations would be discarded.

If the censoring variable takes one of these values, the corresponding failure time is considered to be censored. Following the equal sign are the explanatory effects (sometimes called independent variables or covariates) for the model.

For information about constructing the model effects, see the section "Specification and Parameterization of Model Effects" on page 64 in Chapter 2, "Shared Concepts."

Table 26.6 summarizes the *options* available in the MODEL statement. You can specify these *options* after a slash (/).

Table 26.6 MODEL Statement Options

Option	Description
Model Options	
CLB	Requests confidence limits
ENTRY=	Specifies the left-truncation time variable
INCLUDE=	Includes effects in all models for model selection
INFORMATIVE	Models missing values by using extra indicator variables
OFFSET=	Specifies the offset variable
START=	Includes effects in the initial model for model selection
TYPE3	Displays the Type 3 or joint tests of effects

The following list describes these options:

CLB
constructs confidence limits for each of the parameter estimates. The confidence level is 0.95 by default; you can change it by specifying the ALPHA= option in the PROC PHSELECT statement.

ENTRY=*variable*
specifies the left-truncation time variable. The left-truncation time is the time at which an individual starts to be observed.

INCLUDE=*n* | *single-effect* | **(***effects***)**
forces effects to be included in all models. If you specify INCLUDE=*n*, then the first *n* effects that you list in the MODEL statement are included in all models. If you specify INCLUDE=*single-effect* or INCLUDE=(*effects*), then the specified *effects* are forced into all models. The *effects* that you specify in this option must be explanatory effects that you specify in the MODEL statement before the slash (/).

INFORMATIVE
models missing values by using extra model effects. These effects consist of dummy variables that take the value 1 when the value of a continuous model variable involved in the effect is missing, and take the value 0 otherwise. The missing value in the original model effect is replaced by the average value of the effect for the nonmissing values. For continuous-by-class effects, such as A*x, where A is a classification variable and x is a continuous variable, informative missingness creates multiple dummy columns and substitutes the effect mean of x that corresponds to the respective level of A. Missing values for classification variables are treated as valid levels. For more information about informative missingness, see the section "Informative Missingness" on page 99 in Chapter 2, "Shared Concepts."

OFFSET=*variable*
specifies a *variable* to be used as an offset to the linear predictor. An offset plays the role of an effect whose coefficient is known to be 1. The offset variable cannot appear in the CLASS statement or elsewhere in the MODEL statement. Observations that have missing values for the offset variable are excluded from the analysis.

START=*n* | *single-effect* | **(***effects***)**
begins the selection process from the designated initial model for the forward and stepwise selection methods. If you specify START=*n*, then the starting model includes the first *n* effects that you list in the MODEL statement. If you specify START=*single-effect* or START=(*effects*), then the starting model includes those specified *effects*. The *effects* that you specify in the START= option must be explanatory effects that you specify in the MODEL statement before the slash (/). This option is not available when you specify METHOD=BACKWARD in the SELECTION statement.

TYPE3
requests that Wald statistics for Type 3 contrasts be computed for each effect that you specify in the MODEL statement. For more information, see the section "Joint Tests and Type 3 Tests" on page 1361.

OUTPUT Statement

OUTPUT OUT=*libref.data-table*
< **COPYVARS=(***variables***)** >
< *keyword* < *=name* > >. . . < *keyword* < *=name* > > **;**

The OUTPUT statement creates a data table that contains observationwise statistics that PROC PHSELECT computes after fitting the model. In order to avoid data duplication for large data tables, the variables in the input data table are *not* included in the output data table unless you specify them in the COPYVAR= option.

The output statistics are computed based on the final parameter estimates. If the optimization does not converge, then the output data table is not created.

For observations in which only the censoring variable is missing, values of the linear predictor and the predicted survival probabilities are computed even though these observations do not affect the model fit. This enables, for example, predicted survival probabilities to be computed for new observations.

You must specify the following option:

OUT=*libref.data-table*
names the output data table for PROC PHSELECT to use. You must specify this option before any other options. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to where the data table is to be stored, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 1333.

data-table specifies the name of the output data table.

You can also specify the following syntax elements:

COPYVAR=*variable*
COPYVARS=(*variables***)**
transfers one or more *variables* from the input data table to the output data table.

keyword **< =***name* **>**
specifies a statistic to include in the output data table. Four statistics (DFBETA, RESSCH, RESSCO, and WTRESSCH) are multidimensional; each has a dimension equal to the number of parameters in the model. For the one-dimensional statistic, you can name the variable *name*. For the multidimensional statistic, *name* is used as the prefix; for example, if X1–X10 are the explanatory variables in the model, specifying RESSCH=sch creates the variables schX1–schX10 as the 10-dimensional Schoenfeld residual variables. If you do not provide a *name*, the PHSELECT procedure assigns a default name based on the type of statistic that is requested.

Table 26.7 summarizes the *keywords* available in the OUTPUT statement.

Table 26.7 OUTPUT Statement Keywords

Keyword	Description	Default Names
Statistic Options		
CUMHAZ	Specifies the predicted cumulative hazard	_CUMHAZ_
STDXBETA	Specifies the standard error estimate of the linear predictor	_STDXBETA_
SURVIVAL	Specifies the predicted survival probability	_SURVIVAL_
XBETA	Specifies the linear predictor	_XBETA_
Diagnostic Options		
DFBETA	Specifies the standardize deletion parameter differences	_DFBETA_
LD	Specifies the likelihood displacement	_LD_
RESDEV	Specifies the deviance residual	_RESDEV_
RESMART	Specifies the martingale residual	_RESMART_
RESSCH	Specifies the Schoenfeld residuals	_RESSCH_
RESSCO	Specifies the score residuals	_RESSCO_
WTRESSCH	Specifies the weighted Schoenfeld residuals	_WTRESSCH_
Other Option		
ROLE	Specifies the observation role	_ROLE_

The following list describes these *keywords*. For more information, see the section "Predicted Values and Regression Diagnostics" on page 1359.

CUMHAZ
specifies the predicted cumulative hazard function at the observed time. The default *name* is _CUMHAZ_.

DFBETA
requests the approximate changes in the parameter estimates $(\hat{\boldsymbol{\beta}} - \hat{\boldsymbol{\beta}}_{(j)})$ when the jth observation is omitted. These variables are a weighted transform of the score residual variables and are useful in assessing local influence. The default *name* that is used as a prefix is _DFBETA_.

LD
specifies the approximate likelihood displacement when the observation is omitted. This diagnostic can be used to assess the impact of each observation on the overall fit of the model. The default *name* is _LD_.

RESDEV
specifies the deviance residual. This variable is a transform of RESMART to achieve a more symmetric distribution. The default *name* is _RESDEV_.

RESMART
specifies the martingale residual, which, at the observed time t, can be interpreted as the difference over $[0, t]$ in the observed number of events minus the expected number of events. The default *name* is _RESMART_.

RESSCH
requests the Schoenfeld residuals, which are useful in assessing the proportional hazards assumption. Schoenfeld residuals are computed only at uncensored times and are missing for censored times. If an effect in the MODEL statement is not included in the final model, the corresponding Schoenfeld residuals are set to missing. The default *name* that is used as a prefix is _RESSCH_.

RESSCO
requests the score residuals, which are a decomposition of the first partial derivative of the log likelihood. They can be used to assess the leverage that each subject exerts in the parameter estimation. The default *name* that is used as a prefix is _RESSCO_.

ROLE
specifies the numeric variable that indicates the role that each observation plays in fitting the model. The default *name* is _ROLE_. Table 26.8 shows how this variable is interpreted for each observation.

Table 26.8 Role Interpretation

Value	Observation Role
0	Not used
1	Training
2	Validation
3	Testing

If you do not partition the input data by specifying a PARTITION statement, then the role variable value is 1 for observations that are used in fitting the model and 0 for observations that have at least one missing or invalid value for the response, regressor, frequency, or weight variables.

STDXBETA
specifies the standard error estimates of XBETA. The default *name* is _STDXBETA_.

SURVIVAL
requests the predicted survival probabilities at the observed times. The default *name* is _SURVIVAL_.

WTRESSCH
requests the weighted Schoenfeld residuals, which are useful in investigating the nature of nonproportionality if the proportional hazard assumption does not hold. If an effect in the MODEL statement is not included in the final model, the corresponding weighted Schoenfeld residuals are set to missing. The default *name* that is used as a prefix is _WTRESSCH_.

XBETA
specifies the linear predictor. The default *name* is _XBETA_.

PARTITION Statement

PARTITION *partition-option* **;**

The PARTITION statement specifies how observations in the input data set are logically partitioned into disjoint subsets for model training, validation, and testing. For more information, see the section "Using Validation and Test Data" on page 101 in Chapter 2, "Shared Concepts." Either you can designate a variable in the input data table and a set of formatted values of that variable to determine the role of each observation, or you can specify proportions to use for randomly assigning observations to each role.

You must specify exactly one of the following *partition-options*:

FRACTION(<TEST=*fraction***> <VALIDATE=***fraction***> <SEED=***number***>)**
randomly assigns specified proportions of the observations in the input data table to the roles. You specify the proportions for testing and validation by using the TEST= and VALIDATE= suboptions. If you specify both the TEST= and VALIDATE= suboptions, then the sum of the specified fractions must be less than 1 and the remaining fraction is the proportion of the observations assigned to the training role. The SEED= option specifies an integer that is used to start the pseudorandom number generator for random partitioning of data for training, testing, and validation. If you do not specify SEED=*number* or if *number* is less than or equal to 0, the seed is generated by reading the time of day from the computer's clock.

ROLE=*variable* **(<TEST='***value***'> <TRAIN='***value***'> <VALIDATE='***value***'>)**
ROLEVAR=*variable* **(<TEST='***value***'> <TRAIN='***value***'> <VALIDATE='***value***'>)**
names the *variable* in the input data table whose values are used to assign roles to each observation. This *variable* cannot also appear as an analysis variable in other statements or options. The TEST=, TRAIN=, and VALIDATE= suboptions specify the formatted values of this variable that are used to assign observation roles. If you do not specify the TRAIN= suboption, then all observations whose role is not determined by the TEST= or VALIDATE= suboption are assigned to the training role.

SELECTION Statement

SELECTION <**METHOD=***method*< *(method-options)*>>< *options*> **;**

The SELECTION statement performs model selection by examining whether effects should be added to or removed from the model according to rules that are defined by model selection methods. You can use the PLOT= option to produce graphical summaries of the selection process or the HIERARCHY= option to specify whether and how a requirement of model hierarchy is applied. The statement is fully documented in the section "SELECTION Statement" on page 42 in Chapter 2, "Shared Concepts."

The PHSELECT procedure supports the following effect-selection *methods* in the SELECTION statement:

BACKWARD performs backward elimination. This method starts with all effects in the model and deletes effects.

BACKWARD(FAST) performs fast backward elimination. This method starts with all effects in the model and deletes effects without refitting the model.

FORWARD performs forward selection. This method starts with no effects in the model and adds effects.

LASSO performs model selection by the group LASSO method. This method adds and removes effects by using a sequence of LASSO steps. For more information, see the section "Group LASSO Selection" on page 87 in Chapter 2, "Shared Concepts."

NONE results in no model selection. This method fits the full model.

STEPWISE performs stepwise selection. This method is similar to the FORWARD method, except that effects already in the model do not necessarily stay there.

By default, METHOD=STEPWISE.

The SELECT=, CHOOSE=, and STOP= *method-options* default to the Schwartz Bayesian criterion (SBC).

You can specify the following criteria in the SELECT=, CHOOSE=, and STOP= *method-options*:

AIC uses Akaike's information criterion (Akaike 1974) on the training data.

AICC uses a small-sample bias-corrected version of Akaike's information criterion, as promoted by Hurvich and Tsai (1989) and Burnham and Anderson (1998), on the training data.

SBC | BIC uses the Schwarz Bayesian criterion (Schwarz 1978) on the training data.

SL uses the significance level of the score test as the criterion on the training data (not available for the CHOOSE= option).

VALIDATE uses the –2 log-likelihood (M2LL) value computed on the validation data as the criterion (not available for the SELECT= option).

If you specify METHOD=LASSO and do not specify either the CHOOSE= or STOP= option, then the model in the last LASSO step is chosen as the selected model.

NOTE: If you use the fast backward elimination method, then the –2 log-likelihood, AIC, AICC, and SBC statistics are approximated at each step where the model is not refit, and hence they do not match the values that are computed when that model is fit outside the selection routine. Similarly, if you specify SELECT=AIC, AICC, or SBC, the selection criteria are estimated (Lawless and Singhal 1978), and hence they do not match the values that are computed when that model is fit outside the selection routine.

NOTE: The default model hierarchy method is HIERARCHY=NONE for the stepwise, forward, and fast backward selection methods. The backward elimination method always uses the HIERARCHY=SINGLE *method-option*.

If you specify DETAILS=ALL, the LASSO method produces a table that displays the effects that are added or removed at each step and the corresponding fit statistics. For the other methods, you can specify the following values for the DETAILS= option:

SUMMARY produces a summary table that shows the effect that is added or removed at each step along with the criteria specified in the SELECT=, CHOOSE=, and STOP= *method-options*. The summary table is produced by default if you do not specify the DETAILS= option.

STEPS produces the preceding summary table and displays the results from fitting each model at each step.

ALL produces the preceding tables and a detailed listing of all candidates at each step along with their ranking in terms of the selection criterion for addition to or removal from the model.

If you specify the PLOTS=CRITERIA or PLOTS=ALL option, then a plot of the fit criterion by the selection step is created for the AIC, AICC, and SBC statistics. If you also specify a PARTITION statement, then the same type of plot is created for the M2LL statistics.

If you specify the PLOTS=FITBYROLE or PLOTS=ALL option and a PARTITION statement, then a plot of the M2LL by the selection step for each role is created.

The PLOTS= option is not available for the LASSO method.

STRATA Statement

STRATA *variable* < / **MISSING** > **;**

The STRATA statement specifies a *variable* whose values determine the stratification of the data. Strata are formed according to the nonmissing values of the STRATA *variable* unless you specify the MISSING option.

WEIGHT Statement

WEIGHT *variable* **;**

The *variable* in the WEIGHT statement is used as a weight to perform a weighted analysis of the data. Observations that have nonpositive or missing weights are not included in the analysis. The values of the WEIGHT variable can be nonintegral and are not truncated. When you omit the WEIGHT statement, all observations that are used in the analysis are assigned a weight of 1.

Details: PHSELECT Procedure

Missing Values

Any observation that has missing values for the response, frequency, weight, offset, or explanatory variables is excluded from the analysis; however, missing values are valid for response and explanatory variables that are specified along with the MISSING option in the CLASS statement. Observations that have a nonpositive weight or a frequency less than 1 are also excluded.

The estimated linear predictor and the fitted probabilities are not computed for any observation that has missing offset or explanatory variable values. However, if only the response value is missing, the linear predictor and the fitted probabilities can be computed and output to a data table by using the OUTPUT statement.

You can also model the missing values by specifying the INFORMATIVE option in the MODEL statement. For more information about informative missingness, see the section "Informative Missingness" on page 99 in Chapter 2, "Shared Concepts."

Cox Proportional Hazards Regression Model

The Cox proportional hazards regression model is a semiparametric model that assumes a parametric form for the effects of the explanatory variables, but it allows an unspecified form for the underlying survivor function. The survival time of each member of a population is assumed to follow its own hazard function, $\lambda_j(t)$, expressed as

$$\lambda_j(t) = \lambda(t; \mathbf{Z}_j) = \lambda_0(t) \exp(\mathbf{Z}_j' \boldsymbol{\beta})$$

where $\lambda_0(t)$ is an arbitrary and unspecified baseline hazard function, $\mathbf{Z}_j$ is the vector of explanatory variables for the jth individual, and $\boldsymbol{\beta}$ is the vector of unknown regression parameters that is associated with the explanatory variables.

The population under study can consist of a number of subpopulations, each of which has its own baseline hazard function. A stratified analysis is needed to adjust for such subpopulation differences. Under the stratified model, the hazard function for the jth individual in the ith stratum is expressed as

$$\lambda_{ij}(t) = \lambda_{i0}(t) \exp(\mathbf{Z}_{ij}' \boldsymbol{\beta})$$

where $\lambda_{i0}(t)$ is the baseline hazard function for the ith stratum and $\mathbf{Z}_{ij}$ is the vector of explanatory variables for the individual. The regression coefficients are assumed to be the same for all individuals across all strata.

To estimate $\boldsymbol{\beta}$, Cox (1972, 1975) introduced the partial likelihood function, which eliminates the unknown baseline hazard functions and accounts for censored survival times.

Partial Likelihood for the Cox Model

Let $\mathbf{Z}_l$ denote the vector explanatory variables for the lth individual, and let w_l be the associated weight. Let $t_1 < t_2 < \cdots < t_k$ denote the k distinct, ordered event times. Let d_i denote the multiplicity of failures at t_i; that is, d_i is the size of the set $\mathcal{D}_i$ of individuals that fail at t_i. Let $\mathcal{R}_i$ be the risk set at t_i. Using this notation, the following equations describe the partial likelihood functions that PROC PHSELECT uses to estimate $\boldsymbol{\beta}$.

Breslow Likelihood

$$L(\boldsymbol{\beta}) = \prod_{i=1}^{k} \frac{e^{\boldsymbol{\beta}' \sum_{j \in \mathcal{D}_i} \mathbf{Z}_j}}{\left[\sum_{l \in \mathcal{R}_i} e^{\boldsymbol{\beta}' \mathbf{Z}_l} \right]^{d_i}}$$

If you incorporate weights, the Breslow likelihood becomes

$$L(\boldsymbol{\beta}) = \prod_{i=1}^{k} \frac{e^{\boldsymbol{\beta}' \sum_{j \in \mathcal{D}_i} w_j \mathbf{Z}_j}}{\left[\sum_{l \in \mathcal{R}_i} w_l e^{\boldsymbol{\beta}' \mathbf{Z}_l} \right]^{\sum_{j \in \mathcal{D}_i} w_j}}$$

In a stratified analysis, the partial likelihood is the product of the partial likelihood functions for the individual strata.

The regression coefficients are estimated by maximizing the log of the partial likelihood function with respect to $\boldsymbol{\beta}$.

The LASSO Method of Model Selection

LASSO Selection

The PHSELECT procedure implements the group LASSO method, which is described in the section "Group LASSO Selection" on page 87 in Chapter 2, "Shared Concepts." The current section provides some background about the LASSO method that you need in order to understand the group LASSO method.

LASSO (least absolute shrinkage and selection operator) selection arises from a constrained form of ordinary least squares regression in which the sum of the absolute values of the regression coefficients is constrained to be smaller than a specified parameter. More precisely, let $\mathbf{Z} = (\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_m)$ denote the matrix of covariates, and let $\mathbf{y}$ denote the response. Then for a given parameter t, the LASSO regression coefficients $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_m)$ are the solution to the following constrained least squares problem:

$$\min ||\mathbf{y} - \mathbf{Z}\boldsymbol{\beta}||^2 \qquad \text{subject to} \quad \sum_{j=1}^{m} |\beta_j| \leq t$$

For the Cox proportional hazards regression model, the LASSO regression coefficients $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_m)$ are the solution to the following constrained optimization problem,

$$\min\{-l(\boldsymbol{\beta})\} \qquad \text{subject to} \quad \sum_{j=1}^{m} |\beta_j| \leq t$$

where $l(\boldsymbol{\beta})$ is the log of the partial likelihood function $L(\boldsymbol{\beta})$ defined in the section "Partial Likelihood for the Cox Model" on page 1357.

Provided that the LASSO parameter t is small enough, some of the regression coefficients will be exactly 0. Hence, you can think of the LASSO method as selecting a subset of the regression coefficients for each LASSO parameter. By increasing the LASSO parameter in discrete steps, you obtain a sequence of regression coefficients for which the nonzero coefficients at each step correspond to selected parameters. For more information about the LASSO method, see, for example, Hastie, Tibshirani, and Friedman (2009).

Model Fit and Assessment Statistics

The statistics that are defined in this section are useful for assessing the fit of the model to your data; they are displayed in the "Fit Statistics" table. The statistics are computed for each data role when you specify a PARTITION statement.

Information Criteria

The calculation of the information criteria uses the following formulas, where p denotes the number of effective parameters in the candidate model, F denotes the sum of the frequencies of the uncensored observations that are used, and l is the log partial likelihood that is evaluated at the converged estimates:

$$\begin{aligned}
\text{AIC} &= -2l + 2p \\
\text{AICC} &= \begin{cases} -2l + 2pF/(F-p-1) & \text{when } F > p+2 \\ -2l + 2p(p+2) & \text{otherwise} \end{cases} \\
\text{SBC} &= -2l + p\log(F)
\end{aligned}$$

If you do not specify a FREQ statement, F equals n, the number of uncensored observations that are used.

Predicted Values and Regression Diagnostics

For the ith observation, let $\{(X_i, \Delta_i, \mathbf{Z}_i)\}$ represent the failure time, the event indicator, and the vector of covariate values, respectively. Let $\boldsymbol{\beta}$ be the vector of regression coefficients. For $t \geq 0$, let

$$\begin{aligned}
Y_i(t) &= I(X_i \geq t) \\
N_i(t) &= I(X_i \leq t) \\
S^{(0)}(\boldsymbol{\beta}, t) &= \sum_i Y_i(t)\mathrm{e}^{\boldsymbol{\beta}'\mathbf{Z}_i} \\
S^{(1)}(\boldsymbol{\beta}, t) &= \sum_i Y_i(t)\mathrm{e}^{\boldsymbol{\beta}'\mathbf{Z}_i}\mathbf{Z}_i \\
\bar{\mathbf{Z}}(\boldsymbol{\beta}, t) &= \frac{S^{(1)}(\boldsymbol{\beta}, t)}{S^{(0)}(\boldsymbol{\beta}, t)} \\
d\Lambda_0(\boldsymbol{\beta}, t) &= \sum_i \frac{dN_i(t)}{S^{(0)}(\boldsymbol{\beta}, t)} \\
dM_i(\boldsymbol{\beta}, t) &= dN_i(t) - Y_i(t)\mathrm{e}^{\boldsymbol{\beta}'\mathbf{Z}_i(t)}d\Lambda_0(\boldsymbol{\beta}, t)
\end{aligned}$$

Let $\hat{\boldsymbol{\beta}}$ be the estimator of $\boldsymbol{\beta}$, and let $\mathcal{I}(\hat{\boldsymbol{\beta}})$ be the observed information matrix.

Predicted Values

Let $\mathbf{z}$ be a vector of covariates. The linear predictor and its standard error estimate are, respectively,

$$\mathbf{z}'\hat{\boldsymbol{\beta}} \quad \text{and} \quad \sqrt{\mathbf{z}'\mathcal{I}^{-1}(\hat{\boldsymbol{\beta}})\mathbf{z}}$$

The predicted cumulative hazard at a time point t is

$$\hat{\Lambda}(t, \mathbf{z}) = \mathrm{e}^{\hat{\boldsymbol{\beta}}'\mathbf{z}} \int_0^t d\Lambda_0(\hat{\boldsymbol{\beta}}, s)$$

and the predicted survival probability is

$$\hat{\mathbf{S}}(t, \mathbf{z}) = \mathrm{e}^{-\hat{\Lambda}(t,\mathbf{z})}$$

Residuals

The martingale residual for the *i*th observation is

$$\hat{M}_i = \Delta_i - \hat{\Lambda}(X_i, \mathbf{Z}_i)$$

The deviance residual is a transform of the martingale residual,

$$\hat{D}_i = \text{sign}(\hat{M}_i)\sqrt{2[-\hat{M}_i - \Delta_i \log(\Delta_i - \hat{M}_i)]}$$

The square root shrinks large negative martingale residuals, and the logarithmic transformation expands martingale residuals that are close to unity.

The Schoenfeld (1982) residual vector is calculated per event time. If the *i*th observation is uncensored, the Schoenfeld residual vector is

$$\hat{\mathbf{U}}_i = \mathbf{Z}_i - \bar{\mathbf{Z}}(\hat{\boldsymbol{\beta}}, X_i)$$

Otherwise, the elements of the Schoenfeld residual vector are set to missing values. Under the proportional hazards assumption, the Schoenfeld residuals have the sample path of a random walk; therefore, they are useful in assessing time trend or lack of proportionality. Harrell (1986) proposed a z-transform of the Pearson correlation between these residuals and the rank order of the failure time as a test statistic for nonproportional hazards.

The score residual vector for the *i*th observation is

$$\hat{\mathbf{L}}_i = \int_0^\infty [\mathbf{Z}_i(s) - \bar{\mathbf{Z}}(\hat{\boldsymbol{\beta}}, s)] dM_i(\hat{\boldsymbol{\beta}}, s)$$

The score residuals are a decomposition of the first partial derivative of the log likelihood. They are useful in assessing the influence of each observation on individual parameter estimates. Therneau, Grambsch, and Fleming (1990) have considered a Kolmogorov-type test based on the cumulative sum of the residuals for detecting nonproportional hazards.

Other Regression Diagnostics

The vector of weighted Schoenfeld residuals, $\mathbf{W}_i$, is computed as

$$\mathbf{W}_i = n_e \mathcal{I}^{-1}(\hat{\boldsymbol{\beta}})\hat{\mathbf{U}}_i$$

where n_e is the total number of events and $\hat{\mathbf{U}}_i$ is the vector of Schoenfeld residuals for the i observation. The weighted Schoenfeld residuals are useful in assessing the proportional hazards assumption. The idea is that most of the common alternatives to the proportional hazards can be cast in terms of a time-varying coefficient model,

$$\lambda(t, \mathbf{Z}) = \lambda_0(t)\exp(\beta_1(t)Z_1 + \beta_2(t)Z_2 + \cdots)$$

where $\lambda(t, \mathbf{Z})$ and $\lambda_0(t)$ are hazard rates. Let $\hat{\beta}_j$ and r_{ij} be the jth component of $\hat{\boldsymbol{\beta}}$ and $\mathbf{r}_i$, respectively. Grambsch and Therneau (1994) suggest using a smoothed plot of $(\hat{\beta}_j + r_{ij})$ versus t_i to discover the functional form of the time-varying coefficient $\beta_j(t)$. A zero slope indicates that the coefficient does not vary with time.

The DFBETA diagnostics approximate the changes in the parameter estimates, $\delta_i \hat{\boldsymbol{\beta}}_i = \hat{\boldsymbol{\beta}} - \hat{\boldsymbol{\beta}}_{(-i)}$, when the *i*th observation is left out. The *j*th component of $\delta_i \hat{\boldsymbol{\beta}}$ can be used to assess any unusual effect of the *i*th observation on $\hat{\beta}_j$. The exact computation of $\delta_i \hat{\boldsymbol{\beta}}$ involves refitting the model each time a subject is omitted. Cain and Lange (1984) derived the following approximation of $\delta_i \hat{\boldsymbol{\beta}}$ as weighted score residuals,

$$\text{DFBETA}_i = \mathcal{I}^{-1}(\hat{\boldsymbol{\beta}})\hat{\mathbf{L}}_i$$

where $\hat{\mathbf{L}}_i$ is the score residual vector of the *i*th observation.

The LD statistic approximates the likelihood displacement, which is the amount by which minus twice the log likelihood ($-2 \log L(\hat{\boldsymbol{\beta}})$), under a fitted model, changes when each observation in turn is left out. When the *i*th subject is omitted, the likelihood displacement is

$$2 \log L(\hat{\boldsymbol{\beta}}) - 2 \log L(\hat{\boldsymbol{\beta}}_{(-i)})$$

where $\hat{\boldsymbol{\beta}}_{(-i)}$ is the vector of parameter estimates that you obtain by fitting the model without the *i*th observation. Instead of refitting the model without the *i*th observation, Pettitt and Bin Daud (1989) propose that the likelihood displacement for the *i*th observation be approximated by

$$\text{LD}_i = \hat{\mathbf{L}}_i' \mathcal{I}^{-1}(\hat{\boldsymbol{\beta}})\hat{\mathbf{L}}_i$$

where $\hat{\mathbf{L}}_i$ is the score residual vector of the *i*th observation.

Joint Tests and Type 3 Tests

Linear hypotheses for $\boldsymbol{\beta}$ are expressed in matrix form as

$$H_0: \mathbf{L}\boldsymbol{\beta} = \mathbf{c}$$

where $\mathbf{L}$ is a matrix of coefficients for the linear hypotheses and $\mathbf{c}$ is a vector of constants. The vector of regression coefficients $\boldsymbol{\beta}$ includes slope parameters and intercept parameters. The Wald chi-square statistic for testing H_0 is computed as

$$\chi^2_W = (\mathbf{L}\widehat{\boldsymbol{\beta}} - \mathbf{c})'[\mathbf{L}\widehat{\mathbf{V}}(\widehat{\boldsymbol{\beta}})\mathbf{L}']^{-1}(\mathbf{L}\widehat{\boldsymbol{\beta}} - \mathbf{c})$$

where $\widehat{\mathbf{V}}(\widehat{\boldsymbol{\beta}})$ is the estimated covariance matrix. Under H_0, χ^2_W has an asymptotic chi-square distribution with r degrees of freedom, where r is the rank of $\mathbf{L}$.

For models that use less-than-full-rank parameterization (as specified by the PARAM=GLM option in the CLASS statement), a Type 3 test of an effect of interest (main effect or interaction) is a test of the Type III estimable functions that are defined for that effect. When the model contains no missing cells, the Type 3 test of a main effect is equivalent to testing the hypothesis of equal marginal means. For more information about Type III estimable functions, see the chapter "The GLM Procedure" and the section "The Four Types of Estimable Functions" in *SAS/STAT User's Guide*. Also see Littell, Freund, and Spector (1991).

For models that use full-rank parameterization, all parameters are estimable when there are no missing cells, so it is unnecessary to define estimable functions. The standard test of an effect of interest in this case is the joint test that the values of the parameters associated with that effect are 0. For a model that uses effects parameterization (as specified by the PARAM=EFFECT option in the CLASS statement), the joint test for a main effect is equivalent to testing the equality of marginal means. For a model that uses

reference parameterization (as specified by the PARAM=REF option in the CLASS statement), the joint test is equivalent to testing the equality of cell means at the reference level of the other model effects. For more information about the coding scheme and the associated interpretation of results, see Muller and Fetterman (2002, Chapter 14).

If there is no interaction term, the Type 3 test of an effect for a model that uses GLM parameterization is the same as the joint test of the effect for the model that uses full-rank parameterization. In this situation, the joint test is also called the Type 3 test. For a model that contains an interaction term and no missing cells, the Type 3 test of a component main effect under GLM parameterization is the same as the joint test of the component main effect under effect parameterization. Both test the equality of cell means. But this Type 3 test differs from the joint test under reference parameterization, which tests the equality of cell means at the reference level of the other component main effect. If some cells are missing, you can obtain meaningful tests only by testing a Type III estimation function, so in this case you should use GLM parameterization.

The results of a Type 3 test or a joint test do not depend on the order in which you specify the terms in the MODEL statement.

Multithreading

The PHSELECT procedure allocates data to different threads and calculates the likelihood function, gradient, and Hessian by accumulating the values from all threads. For more information about how PROC PHSELECT uses threads, see the section "Multithreading" on page 102 in Chapter 2, "Shared Concepts."

Optimization Algorithms

Several optimization techniques are available in PROC PHSELECT. You can choose a particular optimizer by using the TECHNIQUE= option in the PROC PHSELECT statement. Table 26.9 summarizes the optimization techniques available in PROC PHSELECT.

Table 26.9 Optimization Techniques

TECHNIQUE=	Algorithm
TRUREG	Trust region method
NEWRAP	Newton-Raphson method with line search
NRRIDG	Newton-Raphson method with ridging
QUANEW	Quasi-Newton methods
DBLDOG	Double-dogleg method
CONGRA	Conjugate gradient methods
NMSIMP	Nelder-Mead simplex method

There is no algorithm for optimizing general nonlinear functions that always finds the global optimum for a general nonlinear optimization problem in a reasonable amount of time. Because no single optimization technique is always superior to others, PROC PHSELECT provides a variety of optimization techniques that work well in various circumstances. However, you can devise problems for which none of the techniques in PROC PHSELECT can find the correct solution. Moreover, nonlinear optimization can be computationally

expensive in terms of time and memory, so you must be careful when matching an algorithm to a problem. The section "Choosing an Optimization Algorithm" on page 103 in Chapter 2, "Shared Concepts," is helpful in choosing a suitable optimization algorithm.

In addition, PROC PHSELECT uses the finite-difference method to compute second-order derivatives, unless you specify the HESSIAN option. The HESSIAN option requires extra memory, the size of which also depends on the number of distinct uncensored failure times in the data.

Displayed Output

The following sections describe the output that PROC PHSELECT produces. The output is organized into various tables, which are discussed in their order of appearance.

Model Information

The "Model Information" table displays basic information about the model, such as the response variable, the frequency variable, the link function, and the distribution of the data that is assumed by the PHSELECT procedure.

Number of Observations

The "Number of Observations" table displays the number of observations that are read from the input data table and the number of observations that are used in the analysis. If you specify a FREQ statement, the table displays the sum of the frequencies that are read and used. If you specify a WEIGHT statement, the table displays the sum of $f_i w_i$ that are read and used. If you specify a PARTITION statement, the table displays the values for each role.

–2 Log Likelihood for Null Model

When you specify the LOGLIKENULL option in the PROC PHSELECT statement, the PHSELECT procedure displays the –2 log-likelihood value of the null model for the training data. If you also specify a PARTITION statement, PROC PHSELECT displays the corresponding statistic for the validation data and the test data.

Class Level Information

The "Class Level Information" table lists the levels of every variable that you specify in the CLASS statement. You should check this information to make sure that the data are correct. You can adjust the order of the CLASS variable levels by using the ORDER= option in the CLASS statement. You can suppress the "Class Level Information" table completely or partially by using the NOCLPRINT= option in the PROC PHSELECT statement.

If the classification variables use a nonsingular parameterization, the "Class Level Information" table also displays the reference value for each variable.

Selection Information

When you specify the SELECTION statement, the PHSELECT procedure by default produces a series of tables that display information about the model selection. The "Selection Information" table informs you about the model selection method, selection and stop criteria, and other parameters that govern the selection. You can suppress this table by specifying DETAILS=NONE in the SELECTION statement.

Iteration History

When you specify the ITHIST option in the PROC PHSELECT statement, the "Iteration History" table displays, for each iteration of the optimization, the number of function evaluations (including gradient and Hessian evaluations), the value of the objective function, the change in the objective function from the previous iteration, and the absolute value of the largest (projected) gradient element.

Convergence Status

The convergence status table is a small ODS table that appears as a message that indicates whether the optimization succeeded and which convergence criterion was met. If the optimization fails, the message indicates the reason for the failure. If you save the convergence status table to an output data set, a numeric Status variable is added that enables you to assess convergence programmatically. The values of the Status variable indicate the following:

0 Convergence was achieved, or an optimization was not performed (because TECHNIQUE=NONE is specified).

1 The objective function could not be improved.

2 Convergence was not achieved because of a user interruption or because a limit was exceeded, such as the maximum number of iterations or the maximum number of function evaluations. To modify these limits, see the MAXITER=, MAXFUNC=, and MAXTIME= options in the PROC PHSELECT statement.

3 Optimization failed to converge because function or derivative evaluations failed at the starting values or during the iterations or because a feasible point that satisfies the parameter constraints could not be found in the parameter space.

Entry and Removal Candidates

When you specify DETAILS=ALL or DETAILS=STEPS in the SELECTION statement, the PHSELECT procedure produces the "Entry Candidates" or "Removal Candidates" table, which displays the effect names and values of the criterion used to select entering or departing effects at each step of the selection process. For each step, the effects are displayed in sorted order from best to worst of the selection criterion.

Selection Summary

When you specify the SELECTION statement, the PHSELECT procedure produces the "Selection Summary" table, which displays information about which effects were added to or removed from the model in the various steps of the model selection process. The statistic that led to the entry or removal decision is also displayed. You can request further details about the model selection steps by specifying DETAILS=STEPS or DETAILS=ALL in the SELECTION statement. You can suppress the display of this table by specifying DETAILS=NONE in the SELECTION statement.

If you specify the LASSO selection method, then this table displays information about which effect was added to the model, the number of effects in the model, the lambda value, and the information criteria.

Stop Reason

When you specify the SELECTION statement, the PHSELECT procedure produces a simple table that tells you why model selection stopped.

Selection Reason

When you specify the SELECTION statement, the PHSELECT procedure produces a simple table that tells you why the final model was selected.

Selected Effects

When you specify the SELECTION statement, the PHSELECT procedure produces a simple table that tells you which effects are in the final model.

Dimensions

The "Dimensions" table displays size measures that are derived from the model. It displays the number of columns in the current design matrix, the number of effects in the current design, the largest number of design columns associated with an effect, the rank of the matrix, and the number of parameters in the current model, including any scale parameters.

Fit Statistics

The "Fit Statistics" table displays a variety of likelihood-based measures of fit. The values that the "Fit Statistics" table displays are not based on a normalized log-likelihood function. If you specify a PARTITION statement, the table displays the values for each role along with statistics for comparing the training, validation, and testing results. For more information about the statistics that this table displays, see the section "Model Fit and Assessment Statistics" on page 1358.

Parameter Estimates

The parameter estimates, their estimated (asymptotic) standard errors, and p-values for the hypothesis that the parameter is 0 are displayed in the "Parameter Estimates" table. If you request confidence intervals by specifying the CLB option in the MODEL statement, confidence limits are displayed for the parameters.

Parameter Estimates Covariance Matrix

When you specify the COVB option in the PROC PHSELECT statement, the PHSELECT procedure displays the covariance matrix of the parameter estimates. The covariance matrix is computed as the inverse of the negative of the matrix of second derivatives of the log-likelihood function with respect to the model parameters (the Hessian matrix), evaluated at the parameter estimates.

Parameter Estimates Correlation Matrix

When you specify the CORRB option in the PROC PHSELECT statement, the PHSELECT procedure displays the correlation matrix of the parameter estimates.

Model Analysis of Variance (Type III)

When you specify the TYPE3 option in the MODEL statement, the PHSELECT procedure produces the "Model Analysis of Variance (Type III)" table. This table displays tests that all parameters for a particular effect are equal to zero. For more information, see the section "Joint Tests and Type 3 Tests" on page 1361.

Timing

The "Timing" table displays the amount of time (in seconds) that PROC PHSELECT required to perform different tasks in the analysis.

OutputCasTables Table

The OutputCasTables table is a special table that has information about each CAS table that is created during a CAS action execution. The information for each CAS table consists of the CAS table name, the caslib in which the table resides, and the number of columns and rows in the CAS table. Because this table is not a typical ODS table that contains analytical results, you cannot include it in the *table-spec-list* in the DISPLAYOUT statement.

ODS Table Names

Each table that the PHSELECT procedure creates has a name associated with it. You must use this name to refer to the table when you use the DISPLAY statement, the DISPLAYOUT statement, or ODS statements. These names are listed in Table 26.10.

NOTE: The EFFECT statement also create tables which are not listed in this section. For information about these tables, see "ODS Table Names" on page 33 in Chapter 2, "Shared Concepts."

Table 26.10 ODS Tables Produced by PROC PHSELECT

Table Name	Description	Statement	Option
ClassInfo	Level information from the CLASS statement	CLASS	Default
ConvergenceStatus	Status of optimization at conclusion of optimization	Default	
CorrB	Correlation matrix of parameter estimates	PROC PHSELECT	CORRB
CovB	Covariance matrix of parameter estimates	PROC PHSELECT	COVB
Dimensions	Model dimensions	Default	

Table 26.10 *continued*

Table Name	Description	Statement	Option
EntryCandidates	Details about candidates for entry into the model	SELECTION	METHOD=FORWARD or STEPWISE and DETAILS=STEP
FitStatistics	Fit statistics	Default	
IterHistory	Iteration history	PROC PHSELECT	ITHIST
LogLikeNull	–2 log likelihood for null model	PROC PHSELECT	LOGLIKENULL
ModelAnova	Model analysis of variance (Type III)	MODEL	TYPE3
ModelInfo	Information about the modeling environment	Default	
NObs	Number of observations read and used, and breakdown of censored and uncensored values	Default	
OutputCasTables	See the section "OutputCasTables Table" on page 1366	OUTPUT DISPLAYOUT	OUT=
ParameterEstimates	Solutions for the parameter estimates associated with effects in MODEL statements	Default	
RemovalCandidates	Details about candidates for removal from the model	SELECTION	METHOD=BACKWARD or STEPWISE and DETAILS=STEP
SelectedEffects	List of effects selected for the model	SELECTION	Default
SelectionInfo	Information about the settings for model selection	SELECTION	Default
SelectionReason	Reason why the particular model was selected	SELECTION	Default
SelectionSummary	Summary information about model selection steps	SELECTION	Default
StopReason	Reason for termination of model selection	SELECTION	Default
Timing	Absolute and relative times for tasks performed by the procedure	Default	

ODS Graphics

Statistical procedures use ODS Graphics to create graphs as part of their output. ODS Graphics is described in detail in the "Statistical Graphics Using ODS" chapter in *SAS/STAT User's Guide*.

Before you create graphs, ODS Graphics must be enabled (for example, by using the ODS GRAPHICS ON statement). For more information about enabling and disabling ODS Graphics, see the section "Enabling and Disabling ODS Graphics" in that chapter.

The overall appearance of graphs is controlled by ODS styles. Styles and other aspects of using ODS Graphics are discussed in the section "A Primer on ODS Statistical Graphics" in that chapter.

When ODS Graphics is enabled, the SELECTION statement can produce plots to help evaluate the selection process. For information about these plots, see the section "Model Selection Plots" on page 93 in Chapter 2, "Shared Concepts."

PROC PHSELECT assigns a name to each graph that it creates using ODS. You can use these names to refer to the graphs when using ODS. The names are listed in Table 26.11.

Table 26.11 Graphs Produced by PROC PHSELECT

ODS Graph Name	Plot Description	PLOTS= Option
CoefficientPanel	Coefficients and CHOOSE= criterion by step	COEFFICIENTS
ChooseCriterionPlot	CHOOSE= criterion by step	COEFFICIENTS(UNPACK)
CoefficientPlot	Coefficients by step	COEFFICIENTS(UNPACK)
CriterionPanel	Fit criteria by step	CRITERIA
TEST_M2LLPlot	–2 log likelihood on testing data by step	CRITERIA(UNPACK)
TRAIN_M2LLPlot	–2 log likelihood on training data by step	CRITERIA(UNPACK)
VAL_M2LLPlot	–2 log likelihood on validation data by step	CRITERIA(UNPACK)
FitByRolePlot	Progression of average square error by role	FITBYROLE

Examples: PHSELECT Procedure

The examples in this chapter are available in the GitHub repository located at https://github.com/sascommunities/doc-supplement-casstat.

NOTE: Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections "Using CAS Sessions and CAS Engine Librefs" on page 7 and "Loading a SAS Data Set onto a CAS Server" on page 7 in Chapter 2, "Shared Concepts."

The following examples assume that your libref is named mylib, but you can substitute any appropriately defined libref.

Example 26.1: Model Selection

(View the complete code for this example (phex1.sas) in the example repository.)

The following statements examine the same data as in the section "Getting Started: PHSELECT Procedure" on page 1334, but they request model selection via the forward selection technique. Effects that provide the best improvement to the selection criterion, the Schwartz Bayesian criterion (SBC), are added until no more effects can improve the selection criterion. The DETAILS=ALL option in the SELECTION statement produces all tables that are related to model selection. The PLOTS=ALL option produces graphics to help you interpret the selection process. ODS Graphics must be enabled before you can request plots. (For more information about ODS Graphics, see the section "ODS Graphics" on page 1368.)

```
ods graphics on;

proc phselect data=mylib.getStarted;
   class C1-C3;
   model Time*Status(0) = C1-C3 X1-X4;
   selection method=forward(stop=sbc select=sbc) details=all plots=all;
run;
```

The model selection tables are shown in Output 26.1.1 through Output 26.1.3. Results from the selected model are shown in Output 26.1.4 and Output 26.1.5. Selection graphics that the PLOTS= option produces are displayed in Output 26.1.6 and Output 26.1.7.

The "Selection Information" table in Output 26.1.1 summarizes the settings for the model selection. Effects are added to the model only if they produce a significant improvement, which is determined by comparing their SBC values. The forward selection stops three steps after the smallest SBC is obtained or when all effects have been added to the model.

Output 26.1.1 Selection Information

The PHSELECT Procedure

Selection Information	
Description	**Value**
Selection Method	Forward
Select Criterion	SBC
Stop Criterion	SBC
Effect Hierarchy Enforced	None
Stop Horizon	3

For each step of the selection process, the DETAILS=ALL option displays the candidate effects for entering the model along with their SELECT= criterion. Output 26.1.2 displays this table for the first step; the other steps are not shown here.

Output 26.1.2 Step1 Entry Candidates

The PHSELECT Procedure

Forward Selection: Step 1
Effect Entered: X1

Entry Candidates		
Rank	**Effect**	**SBC**
1	**X1**	536.7462
2	**X4**	539.8497
3	**C2**	544.9948
4	**X3**	548.2815
5	**X2**	548.9490
6	**C3**	549.1132
7	**C1**	551.8649

The DETAILS=ALL option also displays the dimensions, fit statistics, and parameter estimates at each step of the selection process; these details are not shown here.

When the selection procedure is complete, the "Selection Summary" table in Output 26.1.3 shows the effects that were added to the model and the value of their selection criterion (and the choose and stop criteria, if they are specified). In step 1, effect X1 made the most significant contribution to the model among the candidate effects, according to the SBC statistic. In step 2, X4 made the most significant contribution when an effect was added to a model that contains X1. In the three subsequent steps, no effect could be added to the model that would reduce the SBC value, so variable selection stopped because the stop horizon (see Output 26.1.1) indicates that at most three steps beyond the minimum SBC value are taken.

In Output 26.1.3, the "Selection Summary" table is followed by three small tables that summarize why the process stopped and which model is selected.

Output 26.1.3 Selection Summary Information

The PHSELECT Procedure

Selection Details

Selection Summary

Step	Effect Entered	Number Effects In	SBC
1	X1	1	536.7462
2	X4	2	535.1652*
3	C1	3	538.2917
4	X2	4	539.2661
5	X3	5	542.4842

* Optimal Value Of Criterion

Selection stopped at a local minimum of the SBC criterion.

The model at step 2 is selected.

Selected Effects: X1 X4

Output 26.1.4 displays information about the selected model. Notice that the –2 log-likelihood value in the "Fit Statistics" table is larger than the value for the full model in Figure 26.7. This is expected because the selected model contains only a subset of the parameters. Because the selected model is more parsimonious than the full model, the discrepancy between the –2 log likelihood and the information criteria is less severe than that shown in Figure 26.7 in the "Getting Started" example.

Output 26.1.4 Dimensions and Fit Statistics

The PHSELECT Procedure

Selected Model

Dimensions	
Number of Effects	2
Max Effect Columns	1
Columns in Design	2
Rank of Design	2

Fit Statistics	
-2 Log Likelihood	526.74445
AIC (smaller is better)	530.74445
AICC (smaller is better)	530.91346
SBC (smaller is better)	535.35258

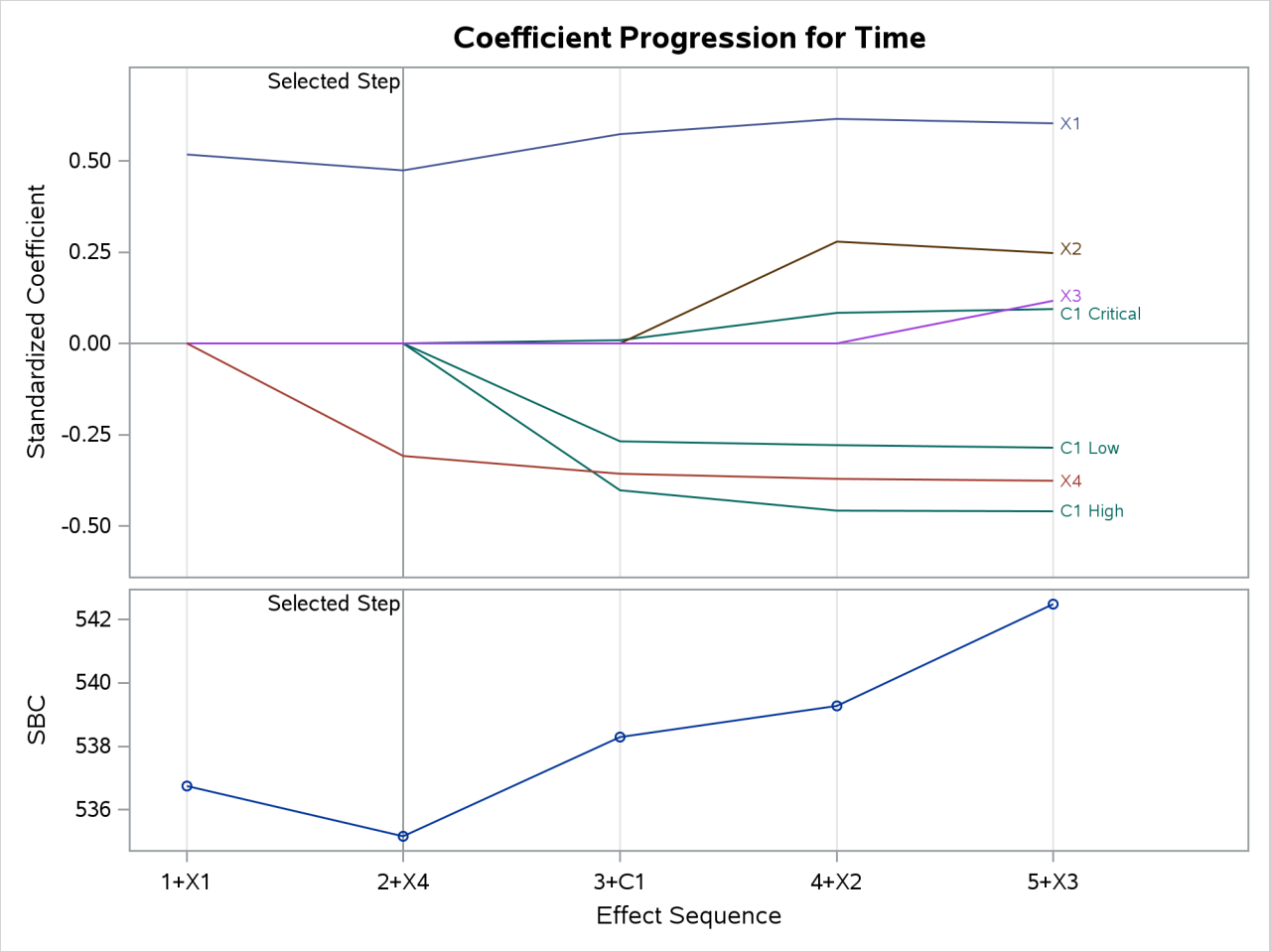
The parameter estimates of the selected model are shown in Output 26.1.5. Notice that the effects are listed in the "Parameter Estimates" table in the order in which they are specified in the MODEL statement and not necessarily in the order in which they are added to the model.

Output 26.1.5 Parameter Estimates

Parameter Estimates					
Parameter	DF	Estimate	Standard Error	Chi-Square	Pr > ChiSq
X1	1	1.605306	0.513811	9.7613	0.0018
X4	1	-0.123304	0.046697	6.9723	0.0083

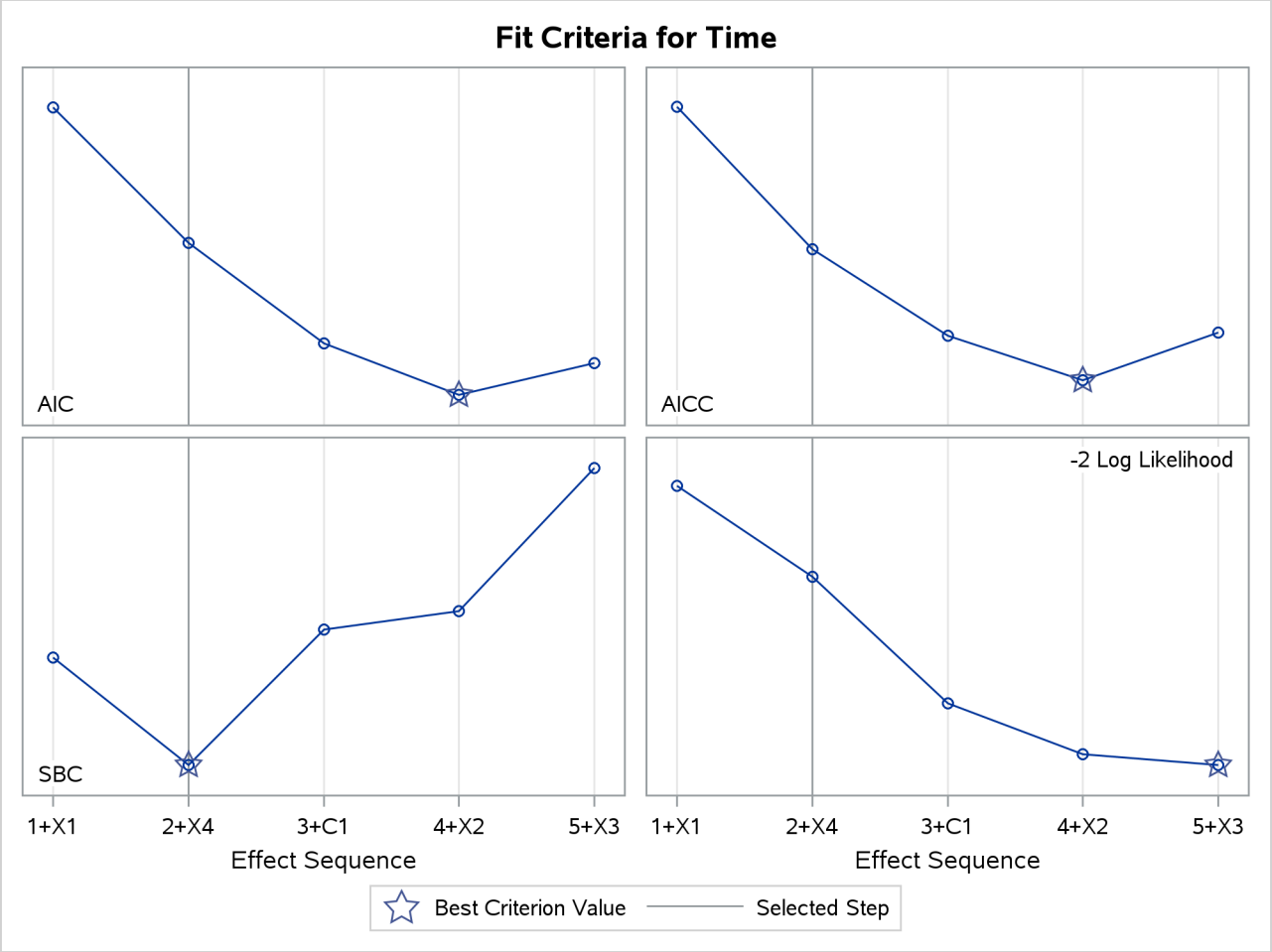
The coefficient panel in Figure 26.1.6 enables you to visualize the selection process. In this plot, standardized coefficients of all the effects that are selected at some step of the stepwise method are plotted as a function of the step number. This enables you to assess the relative importance of the effects that are selected at any step of the selection process and to know when effects entered the model. The lower plot in the panel shows how the criterion that is used to choose the selected model changes as effects enter or leave the model.

Output 26.1.6 Coefficient Progression



The criterion panel in Figure 26.1.7 provides a graphical view of the progression of the fit criteria as the selection process evolves. Notice at Step 2 that the SBC value seems to be at its minimum. Because the stop horizon value is 3 (see Output 26.1.1), three more steps are taken to determine if Step 2 is a global optimum. None of these three subsequent steps have a smaller SBC value than in Step 2, so the global optimum is at step 2.

Output 26.1.7 Criterion Panel



Example 26.2: Stratified Analysis

(View the complete code for this example (phex2.sas) in the example repository.)

In the getStarted data set that is presented in the section "Getting Started: PHSELECT Procedure" on page 1334, suppose the variable C3 represents the gender of the subjects. It is conceivable that males and females belong to different subpopulations and might have different baseline hazard functions. A stratified analysis is conducted by specifying C3 as the stratifying variable in the STRATA statement. The CODE statement requests that a text file named *ScoreCode.txt* be created. This file contains SAS DATA step code to predict survival probabilities at time points that are specified by the TIMEPOINT= option. By default, PROC PHSELECT uses the estimated quartiles of the KM curve (no covariates) of the training data. The SHOWTIME option produces variables that contain these estimated time points.

```
proc phselect data=mylib.getStarted;
   class C1 C2;
   model Time*Status(0) = C1 C2 X1-X4;
   strata C3;
   code file='ScoreCode.txt' showtime;
```

```
run;
```

The "Parameter Estimates" table in Output 26.2.1 shows the resulting regression parameter estimates of the stratified analysis.

Output 26.2.1 Parameter Estimates

The PHSELECT Procedure

Parameter Estimates

Parameter	DF	Estimate	Standard Error	Chi-Square	Pr > ChiSq
C1 Critical	1	0.301314	0.512375	0.3458	0.5565
C1 High	1	-1.030679	0.334457	9.4966	0.0021
C1 Low	1	-0.649179	0.342083	3.6014	0.0577
C1 Medium	0	0	.	.	.
C2 0	1	0.519843	0.405972	1.6396	0.2004
C2 1	0	0	.	.	.
X1	1	1.970290	0.534947	13.5656	0.0002
X2	1	0.752247	0.419589	3.2142	0.0730
X3	1	0.959931	0.600959	2.5515	0.1102
X4	1	-0.117182	0.059406	3.8910	0.0485

You can use the DATA step code from the CODE statement to compute direct adjusted survival probabilities by taking the average of individual predicted survival probabilities. The method of direct adjustment controls for possible confounders that result from an imbalance of subject characteristics between groups. This adjustment is especially useful in nonrandomized studies. In the following statements, the DATA step code is applied to the getStarted data set to obtain the predicted survival probabilities for individuals in that data set and the predicted probabilities at each time point are averaged by gender:

```
data Scores;
   set mylib.getStarted;
   %inc 'ScoreCode.txt';
run;

proc print data=Scores (obs=1);
   var Time_:;
run;

proc means data=Scores mean;
   class C3;
   var S_Time_:;
run;
```

Output 26.2.2 displays the time points at which the survival probabilities were predicted. These time points, Time_1=7, Time_2=19, and Time_3=54, are the quartiles of the failure times based on the Kaplan-Meier curve. Output 26.2.3 displays the direct adjusted survival probabilities at these quartiles. The predicted survival probability at Time_1 for females is slightly lower than that for males, but at Time_3 the direction is reversed. Also revealed is that the getStarted data set has 76 males but only 24 females.

Output 26.2.2 Quartiles of the Observed Failure Times

Obs	Time_1	Time_2	Time_3
1	7	19	54

Output 26.2.3 Direct Adjusted Survival Probabilities

The MEANS Procedure

C3	N Obs	Variable	Label	Mean
F	24	S_Time_1	Survival Probability at Time_1	0.7016195
		S_Time_2	Survival Probability at Time_2	0.3693234
		S_Time_3	Survival Probability at Time_3	0.2765601
M	76	S_Time_1	Survival Probability at Time_1	0.7485799
		S_Time_2	Survival Probability at Time_2	0.4568778
		S_Time_3	Survival Probability at Time_3	0.2354066

References

Akaike, H. (1974). "A New Look at the Statistical Model Identification." *IEEE Transactions on Automatic Control* AC-19:716–723.

Burnham, K. P., and Anderson, D. R. (1998). *Model Selection and Inference: A Practical Information-Theoretic Approach*. New York: Springer-Verlag.

Cain, K. C., and Lange, N. T. (1984). "Approximate Case Influence for the Proportional Hazards Regression Model with Censored Data." *Biometrics* 40:493–499.

Cox, D. R. (1972). "Regression Models and Life-Tables." *Journal of the Royal Statistical Society, Series B* 34:187–220. With discussion.

Cox, D. R. (1975). "Partial Likelihood." *Biometrika* 62:269–276.

Grambsch, P. M., and Therneau, T. M. (1994). "Proportional Hazards Tests and Diagnostics Based on Weighted Residuals." *Biometrika* 81:515–526.

Harrell, F. E. (1986). "The PHGLM Procedure." In *SUGI Supplemental Library Guide, Version 5 Edition*. Cary, NC: SAS Institute Inc.

Hastie, T. J., Tibshirani, R. J., and Friedman, J. H. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd ed. New York: Springer-Verlag.

Hurvich, C. M., and Tsai, C.-L. (1989). "Regression and Time Series Model Selection in Small Samples." *Biometrika* 76:297–307.

Lawless, J. F., and Singhal, K. (1978). "Efficient Screening of Nonnormal Regression Models." *Biometrics* 34:318–327.

Littell, R. C., Freund, R. J., and Spector, P. C. (1991). *SAS System for Linear Models*. 3rd ed. Cary, NC: SAS Institute Inc.

Muller, K. E., and Fetterman, B. A. (2002). *Regression and ANOVA: An Integrated Approach Using SAS Software*. Cary, NC: SAS Institute Inc.

Pettitt, A. N., and Bin Daud, I. (1989). "Case-Weighted Measures of Influence for Proportional Hazards Regression." *Journal of the Royal Statistical Society, Series C* 38:313–329.

Schoenfeld, D. A. (1982). "Partial Residuals for the Proportional Hazards Regression Model." *Biometrika* 69:239–241.

Schwarz, G. (1978). "Estimating the Dimension of a Model." *Annals of Statistics* 6:461–464.

Therneau, T. M., Grambsch, P. M., and Fleming, T. R. (1990). "Martingale-Based Residuals and Survival Models." *Biometrika* 77:147–160.

Chapter 27
PLSMOD Procedure

Contents

Overview: PLSMOD Procedure **1378**
PROC PLSMOD Features 1379
PROC PLSMOD Compared with Other SAS Procedures 1380
PROC PLSMOD Compared with the HPPLS Procedure 1380
PROC PLSMOD Compared with the PLS Procedure 1381
Using CAS Sessions and CAS Engine Librefs 1382
Getting Started: PLSMOD Procedure **1383**
Fitting a PLS Model 1384
Selecting the Number of Factors by Test Set Validation 1386
Predicting New Observations 1389
Syntax: PLSMOD Procedure **1390**
PROC PLSMOD Statement 1391
BY Statement 1394
CLASS Statement 1394
DISPLAY Statement 1395
DISPLAYOUT Statement 1396
EFFECT Statement 1397
MODEL Statement 1398
OUTPUT Statement 1399
PARTITION Statement 1401
Details: PLSMOD Procedure **1402**
Regression Methods 1402
Partial Least Squares 1402
SIMPLS 1403
Principal Component Regression 1404
Reduced Rank Regression 1404
Relationships between Methods 1404
Test Set Validation 1407
Centering and Scaling 1408
Missing Values 1409
Displayed Output 1409
Model Information 1409
Dimensions 1409
Number of Observations 1410
Class Level Information 1410
Centering and Scaling Information 1410

Test Set Validation 1410
Percentage Variation Accounted for by Extracted Factors 1410
Model Details 1410
Parameter Estimates 1411
Timing Information 1411
OutputCasTables Table 1411
ODS Table Names 1411
Examples: PLSMOD Procedure **1412**
Example 27.1: Choosing a PLS Model by Test Set Validation 1413
References **1417**

Overview: PLSMOD Procedure

The PLSMOD procedure fits reduced-rank linear models in SAS Viya

by using any one of a number of linear predictive methods, including *partial least squares* (PLS).

Ordinary least squares regression, as implemented in SAS/STAT procedures such as the GLM and REG procedures, has the single goal of minimizing sample response prediction error, and it seeks linear functions of the predictors that explain as much variation in each response as possible. The PLSMOD procedure implements techniques that have the additional goal of accounting for variation in the predictors, under the assumption that directions in the predictor space that are well sampled should provide better prediction for *new* observations when the predictors are highly correlated. All the techniques that the PLSMOD procedure implements work by extracting successive linear combinations of the predictors, called *factors* (also called *components*, *latent vectors*, or *latent variables*), that optimally address one or both of these two goals: explaining response variation and explaining predictor variation. In particular, the method of partial least squares balances the two objectives by seeking factors that explain both response and predictor variation.

The name "partial least squares" also applies to a more general statistical method that is *not* implemented in this procedure. The partial least squares method was originally developed in the 1960s by the econometrician Herman Wold (1966) for modeling "paths" of causal relation between any number of "blocks" of variables. However, the PLSMOD procedure fits only *predictive* partial least squares models that have one "block" of predictors and one "block" of responses. If you are interested in fitting more general path models, consider using the CALIS procedure in SAS/STAT.

PROC PLSMOD Features

The main features of the PLSMOD procedure are as follows:

- provides model-building syntax with classification variables, continuous variables, interactions, and nestings
- provides effect-construction syntax for polynomial and spline effects
- supports partitioning of data into training and testing roles
- provides test set validation to choose the number of extracted factors, where the model is fit to only part of the available data (the training set) and the fit is evaluated over the other part of the data (the test set)
- produces an output data table that contains predicted values and other observationwise statistics

The PLSMOD procedure implements the following methods:

- principal component regression, which extracts factors to explain as much predictor sample variation as possible
- reduced rank regression, which extracts factors to explain as much response variation as possible. This technique, also known as (maximum) redundancy analysis, differs from multivariate linear regression only when there are multiple responses.
- partial least squares regression, which balances the two objectives of explaining response variation and explaining predictor variation. Two different formulations for partial least squares are available: the original predictive method of Wold (1966) and the straightforward implementation of a statistically inspired modification of the partial least squares (SIMPLS) method of De Jong (1993).

Because the PLSMOD procedure runs on SAS Cloud Analytic Services (CAS), it also does the following:

- enables you to run on a cluster of machines that distribute the data and the computations
- enables you to run in single-machine mode
- exploits all the available cores and concurrent threads. For information about how PROC PLSMOD uses threads, see the section "Multithreading" on page 102 in Chapter 2, "Shared Concepts."

PROC PLSMOD Compared with Other SAS Procedures

The PLSMOD procedure provides functionality to fit reduced-rank linear models that is comparable to that of the HPPLS and PLS procedures in SAS/STAT software.

PROC PLSMOD Compared with the HPPLS Procedure

The functionality of the PLSMOD procedure closely resembles that of the HPPLS procedure, which is a high-performance procedure. The PLSMOD procedure is the next generation of the HPPLS procedure, and it was developed specifically for SAS Viya. Both procedures are designed to run on a cluster of machines that distribute the data and the computations.

The PLSMOD procedure and the HPPLS procedure have the similarities and differences shown in Table 27.1.

Table 27.1 Comparison of PROC PLSMOD and PROC HPPLS

Feature	PROC PLSMOD	PROC HPPLS
Supports general factor extraction methods	Yes	Yes
Supports EFFECT statement	Yes	No
Supports DISPLAY and DISPLAYOUT statements	Yes	No
Supports BY statement	Yes	Yes
Supports ID statement	No	Yes
Supports MISSING=, VARSCALE, and PLOTS options	No	No
Supports cross validation methods	Only test set validation by using the PARTITION statement	Only test set validation by using the PARTITION statement
Includes input variables in output data table	Not by default, in order to avoid data duplication for large data tables. To include input variables, specify them in the COPYVARS= option in the OUTPUT statement.	Not by default, in order to avoid data duplication for large data sets. To include input variables, specify them in the ID statement.
Supports parameterization of classification variables	Full set of parameterizations (supports all values of the PARAM= option in the CLASS statement)	Supports only PARAM=GLM or PARAM=REFERENCE
Threading	Specifically designed for CAS; executes on multiple threads	Primarily designed for a distributed environment; executes on multiple threads

PROC PLSMOD Compared with the PLS Procedure

The PLSMOD procedure and the PLS procedure have the similarities and differences shown in Table 27.2.

Table 27.2 Comparison of PROC PLSMOD and PROC PLS

Feature	PROC PLSMOD	PROC PLS
Supports general factor extraction methods	Yes	Yes
Supports EFFECT statement	Yes	Yes
Supports DISPLAY and DISPLAYOUT statements	Yes	No
Supports BY statement	Yes	Yes
Supports ID statement	No	Yes
Supports MISSING=, VARSCALE, and PLOTS options	No	Yes
Supports cross validation methods	Only test set validation by using the PARTITION statement	Various methods
Syntax of OUTPUT statement	No prefix needed; default prefix used if none is provided. If you do not specify any keywords, the output data table includes predicted values for response variables.	You must provide keyword and prefix
Includes input variables in output data table	Not by default, in order to avoid data duplication for large data tables. To include input variables, specify them in the COPYVARS= option in the OUTPUT statement.	Yes
Supports parameterization of classification variables	Full set of parameterizations (supports all values of the PARAM= option in the CLASS statement)	Only GLM parameterization
Threading	Specifically designed for CAS; executes on multiple threads	Executes on a single thread

Using CAS Sessions and CAS Engine Librefs

SAS Cloud Analytic Services (CAS) is the analytic server and associated cloud services in SAS Viya. This section describes how to create a CAS session and set up a CAS engine libref that you can use to connect to the CAS session. It assumes that you have a CAS server already available; contact your system administrator if you need help starting and terminating a server. This CAS server is identified by specifying the host on which it runs and the port on which it listens for communications. To simplify your interactions with this CAS server, the host information and port information for the server are stored as SAS option values that are retrieved automatically whenever this CAS server needs to be accessed. You can examine the host and port values for the server at your site by using the following statements:

```
proc options option=(CASHOST CASPORT);
run;
```

In addition to starting a CAS server, your system administrator might also have created a CAS session and a CAS engine libref for your use. You can define your own sessions and CAS engine librefs that connect to the CAS server as shown in the following statements:

```
cas mysess;
libname mylib cas sessref=mysess;
```

The CAS statement creates the CAS session named mysess, and the LIBNAME statement creates the mylib CAS engine libref that you use to connect to this session. It is not necessary to explicitly name the CASHOST and CASPORT of the CAS server in the CAS statement, because these values are retrieved from the corresponding SAS option values.

If you have created the mysess session, you can terminate it by using the TERMINATE option in the CAS statement as follows:

```
cas mysess terminate;
```

For more information about the CAS and LIBNAME statements, see the section “Introduction to Shared Concepts” on page 6 in Chapter 2, “Shared Concepts.”

Getting Started: PLSMOD Procedure

(View the complete code for this example at https://github.com/sascommunities/doc-supplement-casstat/tree/main/Examples/plsmodgs1.sas.)

The example in this section illustrates basic features of the PLSMOD procedure. The data are reported in Umetrics (1995); the original source is Lindberg, Persson, and Wold (1983). Suppose you are researching pollution in the Baltic Sea and you want to use the fluorescence spectra of seawater samples to determine the amounts of three compounds present: lignin sulfonate (ls: pulp industry pollution), humic acids (ha: natural forest products), and optical whitener from detergent (dt). Spectrometric calibration is a type of problem in which partial least squares can be very effective. The predictors are the spectra emission intensities at different frequencies in a sample spectrum, and the responses are the amounts of various chemicals in the sample.

For the purpose of calibrating the model, samples that have known compositions are used. The calibration data consist of 16 samples of known concentrations of ls, ha, and dt, with spectra based on 27 frequencies (or, equivalently, wavelengths). In order to demonstrate the use of test set validation, the data contain a variable Role, which is used to assign observations to the training and testing roles. In this case, the training role has nine samples and the testing role has seven samples.

NOTE: Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections "Using CAS Sessions and CAS Engine Librefs" on page 7 and "Loading a SAS Data Set onto a CAS Server" on page 7 in Chapter 2, "Shared Concepts."

The following DATA step creates the mylib.Sample data table, which provides the calibration data, in your CAS session. These statements assume that your CAS engine libref is named mylib, but you can substitute any appropriately defined CAS engine libref.

```
data mylib.Sample;
   input obsnam $ v1-v27 ls ha dt Role $5. @@;
   datalines;
EM1   2766 2610 3306 3630 3600 3438 3213 3051 2907 2844 2796
      2787 2760 2754 2670 2520 2310 2100 1917 1755 1602 1467
      1353 1260 1167 1101 1017         3.0110  0.0000   0.00 TRAIN
EM2   1492 1419 1369 1158  958  887  905  929  920  887  800
       710  617  535  451  368  296  241  190  157  128  106
        89   70   65   56   50         0.0000  0.4005   0.00 TEST
EM3   2450 2379 2400 2055 1689 1355 1109  908  750  673  644
       640  630  618  571  512  440  368  305  247  196  156
       120   98   80   61   50         0.0000  0.0000  90.63 TRAIN
EM4   2751 2883 3492 3570 3282 2937 2634 2370 2187 2070 2007
      1974 1950 1890 1824 1680 1527 1350 1206 1080  984  888
       810  732  669  630  582         1.4820  0.1580  40.00 TEST
EM5   2652 2691 3225 3285 3033 2784 2520 2340 2235 2148 2094
      2049 2007 1917 1800 1650 1464 1299 1140 1020  909  810
       726  657  594  549  507         1.1160  0.4104  30.45 TEST
EM6   3993 4722 6147 6720 6531 5970 5382 4842 4470 4200 4077
      4008 3948 3864 3663 3390 3090 2787 2481 2241 2028 1830
      1680 1533 1440 1314 1227         3.3970  0.3032  50.82 TRAIN
EM7   4032 4350 5430 5763 5490 4974 4452 3990 3690 3474 3357
```

```
      3300 3213 3147 3000 2772 2490 2220 1980 1779 1599 1440
      1320 1200 1119 1032  957         2.4280  0.2981  70.59 TRAIN
EM8   4530 5190 6910 7580 7510 6930 6150 5490 4990 4670 4490
      4370 4300 4210 4000 3770 3420 3060 2760 2490 2230 2060
      1860 1700 1590 1490 1380         4.0240  0.1153  89.39 TRAIN
EM9   4077 4410 5460 5857 5607 5097 4605 4170 3864 3708 3588
      3537 3480 3330 3192 2910 2610 2325 2064 1830 1638 1476
      1350 1236 1122 1044  963         2.2750  0.5040  81.75 TEST
EM10  3450 3432 3969 4020 3678 3237 2814 2487 2205 2061 2001
      1965 1947 1890 1776 1635 1452 1278 1128  981  867  753
       663  600  552  507  468         0.9588  0.1450 101.10 TRAIN
EM11  4989 5301 6807 7425 7155 6525 5784 5166 4695 4380 4197
      4131 4077 3972 3777 3531 3168 2835 2517 2244 2004 1809
      1620 1470 1359 1266 1167         3.1900  0.2530 120.00 TRAIN
EM12  5340 5790 7590 8390 8310 7670 6890 6190 5700 5380 5200
      5110 5040 4900 4700 4390 3970 3540 3170 2810 2490 2240
      2060 1870 1700 1590 1470         4.1320  0.5691 117.70 TEST
EM13  3162 3477 4365 4650 4470 4107 3717 3432 3228 3093 3009
      2964 2916 2838 2694 2490 2253 2013 1788 1599 1431 1305
      1194 1077  990  927  855         2.1600  0.4360  27.59 TRAIN
EM14  4380 4695 6018 6510 6342 5760 5151 4596 4200 3948 3807
      3720 3672 3567 3438 3171 2880 2571 2280 2046 1857 1680
      1548 1413 1314 1200 1119         3.0940  0.2471  61.71 TRAIN
EM15  4587 4200 5040 5289 4965 4449 3939 3507 3174 2970 2850
      2814 2748 2670 2529 2328 2088 1851 1641 1431 1284 1134
      1020  918  840  756  714         1.6040  0.2856 108.80 TEST
EM16  4017 4725 6090 6570 6354 5895 5346 4911 4611 4422 4314
      4287 4224 4110 3915 3600 3240 2913 2598 2325 2088 1917
      1734 1587 1452 1356 1257         3.1620  0.7012  60.00 TEST
;
```

Fitting a PLS Model

To isolate a few underlying spectral factors that provide a good predictive model, you can fit a PLS model to the 16 samples by using the following SAS statements:

```
proc plsmod data=mylib.sample;
   model ls ha dt = v1-v27;
run;
```

By default, the PLSMOD procedure extracts at most 15 factors. The default output from this analysis is presented in Figure 27.1 and Figure 27.2.

Figure 27.1 displays the "Model Information," "Dimensions," and "Number of Observations" tables.

The "Model Information" table identifies the data source and shows that the factor extraction method is partial least squares regression (which is the default) and that the nonlinear iterative partial least squares (NIPALS) algorithm (which is also the default) is used to compute extracted PLS factors.

The "Dimensions" table shows the number of response variables, the number of effects, the number of predictor parameters, and the number of factors to extract.

The "Number of Observations" table shows that all 16 of the sample observations in the input data are used in the analysis; all the samples are used because they all contain complete data.

Figure 27.1 Model Information, Dimensions, and Number of Observations

The PLSMOD Procedure

Model Information	
Data Source	SAMPLE
Factor Extraction Method	Partial Least Squares
PLS Algorithm	NIPALS
Validation Method	None

Dimensions	
Number of Response Variables	3
Number of Effects	27
Number of Predictor Parameters	27
Number of Factors	15

Number of Observations Read	16
Number of Observations Used	16

Figure 27.2 lists the amount of variation, both individual and cumulative, that is accounted for by each of the 15 factors. All the variation in both the predictors and the responses is accounted for by only 15 factors because there are only 16 sample observations. More important, almost all the variation is accounted for by even fewer factors—one or two for the predictors and three to eight for the responses.

Figure 27.2 PLS Variation Summary

Percentage Variation Accounted for by Partial Least Squares Factors

	Model Effects		Response Variables	
Number of Extracted Factors	**Current**	**Total**	**Current**	**Total**
1	97.46068	97.46068	41.91546	41.91546
2	2.18296	99.64365	24.24355	66.15900
3	0.17806	99.82170	24.53393	90.69293
4	0.11973	99.94143	3.78978	94.48271
5	0.04146	99.98289	1.00454	95.48725
6	0.01058	99.99347	2.28084	97.76809
7	0.00168	99.99515	1.16935	98.93744
8	0.00097586	99.99613	0.50410	99.44153
9	0.00142	99.99755	0.12292	99.56446
10	0.00097037	99.99852	0.11027	99.67472
11	0.00032725	99.99884	0.15227	99.82699
12	0.00029338	99.99914	0.12907	99.95606
13	0.00024792	99.99939	0.03121	99.98727
14	0.00042742	99.99981	0.00651	99.99378
15	0.00018639	100.00000	0.00622	100.00000

Selecting the Number of Factors by Test Set Validation

A PLS model is not complete until you choose the number of factors. You can choose the number of factors by using test set validation, in which the data table is divided into two groups called the training data and test data. You fit the model to the training data, and then you check the capability of the model to predict responses for the test data. The predicted residual sum of squares (PRESS) statistic is based on the residuals that are generated by this process.

To select the number of extracted factors by test set validation, you use the PARTITION statement to specify how to logically divide observations in the input data table into two subsets for model training and testing. For example, you can designate a variable in the input data table and a set of formatted values of that variable to determine the role of each observation, as in the following SAS statements:

```
proc plsmod data=mylib.sample;
   model ls ha dt = v1-v27;
   partition roleVar = Role(train='TRAIN' test='TEST');
run;
```

The resulting output is shown in Figure 27.3 through Figure 27.5.

Figure 27.3 Model Information, Dimensions, and Number of Observations with Test Set Validation

The PLSMOD Procedure

Model Information	
Data Source	SAMPLE
Factor Extraction Method	Partial Least Squares
PLS Algorithm	NIPALS
Validation Method	Test Set Validation

Dimensions	
Number of Response Variables	3
Number of Effects	27
Number of Predictor Parameters	27
Maximum Number of Factors	9

Number of Observations Read	16
Number of Observations Used	16
Number of Observations Used for Training	9
Number of Observations Used for Testing	7

Figure 27.4 Test-Set-Validated PRESS Statistics for Number of Factors

The PLSMOD Procedure

Test Set Validation for the Number of Extracted Factors	
Number of Extracted Factors	Root Mean PRESS
0	1.426362
1	1.276694
2	1.181752
3	0.656999
4	0.43457
5	0.420916
6	0.585031
7	0.576586
8	0.563935
9	0.563935

Minimum Root Mean PRESS	0.420916
Minimizing Number of Factors	5

Figure 27.5 PLS Variation Summary for Test-Set-Validated Model

Percentage Variation Accounted for by Partial Least Squares Factors				
	Model Effects		Response Variables	
Number of Extracted Factors	Current	Total	Current	Total
1	95.92495	95.92495	37.27071	37.27071
2	3.86407	99.78903	32.38167	69.65238
3	0.10170	99.89073	20.76882	90.42120
4	0.08979	99.98052	4.66666	95.08787
5	0.01142	99.99194	3.88184	98.96971

In Figure 27.3, the "Model Information" table indicates that test set validation is used. The "Dimensions" table shows that the maximum number of factors to extract is nine. The "Number of Observations" table shows that nine sample observations are assigned for training roles and seven are assigned for testing roles.

Figure 27.4 provides details about the results from test set validation. These results show that the absolute minimum PRESS is achieved with five extracted factors. Notice, however, that this is not much smaller than the PRESS for three factors. By using the CVTEST option, you can perform a statistical model comparison that is suggested by Van der Voet (1994) to test whether this difference is significant, as shown in the following SAS statements:

```
proc plsmod data=mylib.sample cvtest(pval=0.15 seed=12345);
   model ls ha dt = v1-v27;
   partition roleVar = Role(train='TRAIN' test='TEST');
run;
```

The model comparison test is based on a rerandomization of the data. By default, the seed for this randomization is based on the system clock, but it is specified here. The resulting output is presented in Figure 27.6 through Figure 27.8.

Figure 27.6 Model Information with Model Comparison Test

The PLSMOD Procedure

Model Information	
Data Source	SAMPLE
Factor Extraction Method	Partial Least Squares
PLS Algorithm	NIPALS
Validation Method	Test Set Validation
Validation Testing Criterion	Prob T**2 > 0.15
Number of Random Permutations	1000
Random Number Seed for Permutation	12345

Figure 27.7 Testing Test Set Validation for Number of Factors

The PLSMOD Procedure

Test Set Validation for the Number of Extracted Factors			
Number of Extracted Factors	**Root Mean PRESS**	**T**2**	**Prob > T**2**
0	1.426362	5.191629	0.0480
1	1.276694	6.174825	<.0001
2	1.181752	4.60203	0.0620
3	0.656999	3.09999	0.5400
4	0.43457	4.980226	0.0960
5	0.420916	0	1.0000
6	0.585031	2.05496	0.7470
7	0.576586	3.009172	0.4980
8	0.563935	2.416635	0.7500
9	0.563935	2.416635	0.7500

Minimum Root Mean PRESS	0.420916
Minimizing Number of Factors	5
Smallest Number of Factors with p > 0.15	3

Figure 27.8 PLS Variation Summary for Tested Test-Set-Validated Model

Percentage Variation Accounted for by Partial Least Squares Factors				
	Model Effects		Response Variables	
Number of Extracted Factors	Current	Total	Current	Total
1	95.92495	95.92495	37.27071	37.27071
2	3.86407	99.78903	32.38167	69.65238
3	0.10170	99.89073	20.76882	90.42120

The "Model Information" table in Figure 27.6 displays information about the options that are used in the model comparison test. In Figure 27.7, the *p*-value in comparing the test-set-validated residuals from models that have five and three factors indicates that the difference between the two models is insignificant; therefore, the model with fewer factors is preferred. The variation summary in Figure 27.8 shows that more than 99% of the predictor variation and more than 90% of the response variation are accounted for by the three factors.

Predicting New Observations

Now that you have chosen a three-factor PLS model for predicting pollutant concentrations that are based on sample spectra, suppose that you have two new samples. The following SAS statements create a data table that contains the spectra for the new samples:

```
data newobs;
   input obsnam $ v1-v27 @@;
   datalines;
EM17  3933 4518 5637 6006 5721 5187 4641 4149 3789
      3579 3447 3381 3327 3234 3078 2832 2571 2274
      2040 1818 1629 1470 1350 1245 1134 1050  987
EM25  2904 2997 3255 3150 2922 2778 2700 2646 2571
      2487 2370 2250 2127 2052 1713 1419 1200  984
       795  648  525  426  351  291  240  204  162
;
```

You can apply the PLS model to these samples to estimate pollutant concentration by appending the new samples to the original 16 and specifying that the predicted values for all 18 samples be output to a data table, as shown in the following statements:

```
data mylib.all;
   set mylib.sample newobs;
run;

proc plsmod data=mylib.all nfac=2;
   model ls ha dt = v1-v27;
   partition roleVar = Role(train='TRAIN' test='TEST');
   output out=mylib.result pred=p copyvars=(obsnam);
run;
```

```
proc print data=mylib.result;
   where (obsnam in ('EM17','EM25'));
   var obsnam p_ls p_ha p_dt;
run;
```

The COPYVARS= option in the OUTPUT statement requests that the variable obsnam from the input data table be copied to the output data table. The new observations are not used in calculating the PLS model because they have no response values. Their predicted concentrations are shown in Figure 27.9.

Figure 27.9 Predicted Concentrations for New Observations

Obs	obsnam	p_ls	p_ha	p_dt
1	EM17	2.63326	0.22343	80.2027
2	EM25	0.69865	0.14308	98.9937

Syntax: PLSMOD Procedure

The following statements are available in the PLSMOD procedure:

PROC PLSMOD < *options* > **;**
BY *variables* **;**
CLASS *variable* < **(***options***)** >*...* < *variable* < **(***options***)** > > < */ global-options* > **;**
DISPLAY < *table-list* > < */ options* > **;**
DISPLAYOUT *table-spec-list* < */ options* > **;**
EFFECT *name* **=** *effect-type* **(***variables* < */ options* >**)** **;**
MODEL *response-variables* **=** *predictor-effects* < */ options* > **;**
OUTPUT OUT=*libref.data-table*
< **COPYVARS=***(variables)* >
< *keyword* < **=***prefix* > >*...* < *keyword* < **=***prefix* > > **;**
PARTITION *partition-options* **;**

The PROC PLSMOD statement and a single MODEL statement are required. All other statements are optional. The CLASS statement can appear multiple times. If a CLASS statement is specified, it must precede the MODEL statement. The following sections describe the PROC PLSMOD statement and then describe the other statements in alphabetical order.

PROC PLSMOD Statement

PROC PLSMOD < *options* > **;**

The PROC PLSMOD statement invokes the PLSMOD procedure. Table 27.3 summarizes the options available in the PROC PLSMOD statement.

Table 27.3 PROC PLSMOD Statement Options

Option	Description
Basic Options	
DATA=	Specifies the input data table
Model Fitting Options	
CVTEST	Requests that van der Voet's (1994) randomization-based model comparison test be performed
METHOD=	Specifies the general factor extraction method to be used
NFAC=	Specifies the number of factors to extract
NOCENTER	Suppresses centering of the responses and predictors before fitting
NOCVSTDIZE	Suppresses re-centering and rescaling of the responses and predictors when cross validating
NOSCALE	Suppresses scaling of the responses and predictors before fitting
Output Options	
CENSCALE	Displays the centering and scaling information
DETAILS	Displays the details of the fitted model
NOCLPRINT	Limits or suppresses the display of class levels
VARSS	Displays the amount of variation accounted for in each response and predictor

The following list provides details about these *options*.

CENSCALE
lists the centering and scaling information for each response and predictor.

CVTEST < **(***cvtest-options***)** >
requests that van der Voet's (1994) randomization-based model comparison test be performed to test models that have different numbers of extracted factors against the model that minimizes the predicted residual sum of squares. For more information, see the section "Test Set Validation" on page 1407. You can also specify the following *cvtest-options* in parentheses:

NSAMP=*number*
specifies the number of randomizations to perform. By default, NSAMP=1000.

PVAL=*number*
specifies the cutoff probability for declaring an insignificant difference. By default, PVAL=0.10.

SEED=*number*
specifies the seed value for the random number stream. If you do not specify this option or if *number* is less than or equal to 0, the seed is generated by reading the time of day from the computer's clock.

Analyses that use the same (nonzero) seed are not completely reproducible if they are executed on a different number of compute nodes, because the random number streams in separate compute nodes are independent.

STAT=PRESS | T2
specifies the test statistic for the model comparison. You can specify the following values:

PRESS	uses the predicted residual sum of squares.
T2	uses Hotelling's T^2 statistic.

By default, STAT=T2.

DATA=*libref.data-table*
names the input data table for PROC PLSMOD to use. The default is the most recently created data table. *libref.data-table* is a two-level name, where

libref	refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to the data, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 1382.
data-table	specifies the name of the input data table.

DETAILS
lists the details of the fitted model for each successive factor. The listed details are different for different extraction methods. For more information, see the section "Displayed Output" on page 1409.

METHOD=PLS< (*PLS-options***) > | SIMPLS | PCR | RRR**
specifies the general factor extraction method to be used. You can specify the following values:

PCR
uses principal component regression.

PLS< (*PLS-options***) >**
uses partial least squares. You can also specify the following optional *PLS-options* in parentheses:

ALGORITHM=NIPALS | SVD | EIG
names the specific algorithm to use to compute extracted PLS factors. You can specify the following values:

NIPALS requests the usual iterative NIPALS algorithm.

SVD bases the extraction on the singular value decomposition of $\mathbf{X}'\mathbf{Y}$. This algorithm is the most accurate but least efficient approach.

EIG bases the extraction on the eigenvalue decomposition of $\mathbf{Y}'\mathbf{X}\mathbf{X}'\mathbf{Y}$.

By default, ALGORITHM=NIPALS.

EPSILON=*number*
specifies the convergence criterion for the NIPALS algorithm. By default, EPSILON=10^{-12}.

MAXITER=*number*
specifies the maximum number of iterations for the NIPALS algorithm. By default, MAXITER=200.

RRR
uses reduced rank regression.

SIMPLS
uses the straightforward implementation of a statistically inspired modification of the partial least squares (SIMPLS) method of De Jong (1993).

By default, METHOD=PLS(NIPALS).

NFAC=*number*
specifies the number of factors to extract. The default is $\min\{15, p, N\}$, where p is the number of predictors (or the number of response variables when METHOD=RRR) and N is the number of runs (observations). You probably do not need to extract this many factors for most applications. Extracting too many factors can lead to an overfitted model (one that matches the training data too well), sacrificing predictive ability. Thus, if you use the default, you should also either specify the PARTITION statement to select the appropriate number of factors for the final model or consider the analysis to be preliminary and examine the results to determine the appropriate number of factors for a subsequent analysis.

NOCENTER
suppresses centering of the responses and predictors before fitting. This option is useful if the analysis variables are already centered and scaled. For more information, see the section "Centering and Scaling" on page 1408.

NOCLPRINT<=*number***>**
suppresses the display of the "Class Level Information" table if you do not specify *number*. If you specify *number*, the values of the classification variables are displayed only for variables whose number of levels is less than *number*. Specifying a *number* helps to reduce the size of the "Class Level Information" table if some classification variables have a large number of levels.

NOCVSTDIZE
suppresses re-centering and rescaling of the responses and predictors before each model is fit in the cross validation. For more information, see the section "Centering and Scaling" on page 1408.

NOSCALE
suppresses scaling of the responses and predictors before fitting. This option is useful if the analysis variables are already centered and scaled. For more information, see the section "Centering and Scaling" on page 1408.

VARSS
lists, in addition to the average response and predictor sum of squares accounted for by each successive factor, the amount of variation accounted for in each response and predictor.

BY Statement

BY *variables* **;**

You can specify a BY statement in PROC PLSMOD to obtain separate analyses of observations in groups that are defined by the values of the BY variables. If you specify more than one BY statement, only the last one specified is used. For more information, see the discussion of BY-group processing in the "Grouping Data" section of *SAS Programmers Guide: Essentials*.

CLASS Statement

CLASS *variable* <**(***options***)**>. . . <*variable* <**(***options***)**>> </ *global-options*> **;**

The CLASS statement names the classification variables to be used as explanatory variables in the analysis. You can list the response variable for binary models in the CLASS statement, but this is not required. Table 27.4 summarizes the values that you can use for either an *option* or a *global-option*. The options are fully documented in the section "CLASS Statement" on page 9 in Chapter 2, "Shared Concepts."

Table 27.4 CLASS Statement Options

Option	Description
DESCENDING	Reverses the sort order
MISSING	Treats missing values as valid levels
ORDER=	Specifies the sort order for the levels
PARAM=	Specifies the parameterization of the variable
REF=	Specifies the reference level of the variable
SPLIT	Splits levels of CLASS variables into independent effects

DISPLAY Statement

DISPLAY < *table-list* > < / *options* > **;**

The DISPLAY statement enables you to specify a list of display tables to display or exclude. This statement is similar to the ODS SELECT, ODS EXCLUDE, and ODS TRACE statements. However, the DISPLAY statement can improve performance when a large number of tables could be generated (such as in BY-group processing). The procedure processes the DISPLAY statement on a CAS server and thus sends only a subset of ODS tables to the SAS client. Because ODS statements are processed on a SAS client, first all the generated display tables are sent to the client, and then the client creates a subset.

If you use both DISPLAY and ODS statements together, the DISPLAY statement takes precedence over the ODS statements. Note that the ODS EXCLUDE statement processes tables that are sent to the client after they have been filtered by the DISPLAY statement. In some cases, it might appear that the ODS EXCLUDE statement is taking precedence because it can further filter the tables. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

You can specify the *table-list* as a list of table names, paths, partial pathnames, and regular expressions.

The table names that you can specify are listed in the section "ODS Table Names" on page 1411. A path is a table name that is prefixed with dot-separated grouping information. For example, a SelectionSummary table that a procedure produces during a selection routine might have the path *Bygroup1.Summary.SelectionSummary*. A partial pathname does not include all groups; for example, *SelectionSummary* and *Summary.SelectionSummary* are partial pathnames for *Bygroup1.Summary.SelectionSummary*.

When you specify a table name or partial pathname, all display tables whose paths end in the specified name are selected for display or exclusion. For example, both *SelectionSummary* and *Summary.SelectionSummary* select *Bygroup1.Summary.SelectionSummary*.

A regular expression is enclosed in forward slashes (/). For example, specifying "/tions/" selects all pathnames that contain the substring "tions"; in particular, the *Bygroup1.Summary.SelectionSummary* table is selected. Specifying "!/tions/" selects all pathnames that do not contain the substring "tions"; in particular, the *Bygroup1.Summary.SelectionSummary* table is not selected.

You can specify the following *options* after a slash (/):

CASESENSITIVE
performs a case-sensitive comparison of table names in the *table-list* to display table names when tables are subsetted for display. To preserve case, you must enclose table names in the *table-list* in quotation marks.

EXCLUDE
displays all display tables except those that you specify in the *table-list*.

EXCLUDEALL
suppresses display of all tables. This option takes precedence over the other options.

TRACE
displays the display table names, labels, and paths.

DISPLAYOUT Statement

DISPLAYOUT *table-spec-list* < / *options* > **;**

The DISPLAYOUT statement enables you to create output tables from your displayed output. This statement is similar to the ODS OUTPUT statement. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

The *table-spec-list* specifies a list of output tables to create. Each entry in the list has either a *key*=*value* format or a *key* format:

key=*value* specifies *key* as the ODS table name, path, or partial pathname, and specifies *value* as the output table name.

key specifies *key* as the ODS table name and also as the output table name.

The ODS table names that you can specify are listed in the section "ODS Table Names" on page 1411. You cannot specify the ODS table named OutputCasTables in the *table-spec-list*.

Table names and partial pathnames are discussed under the DISPLAY statement. The DISPLAYOUT statement does not support regular expressions.

You can specify the following *options* after a slash (/):

INCLUDEALL
creates output tables for all display tables. The name of the created output table is the same as the name of the corresponding display table. If you specify this option, the *table-spec-list* specification is ignored.

NOREPLACE
does not replace any existing output table of the same name.

REPEATED
replicates all output tables on all nodes.

EFFECT Statement

EFFECT *name*=*effect-type* **(***variables* < / *options* >**) ;**

The EFFECT statement enables you to construct special collections of columns for design matrices. These collections are referred to as *constructed effects* to distinguish them from the usual model effects that are formed from continuous or classification variables, as discussed in the section "GLM Parameterization of Classification Variables and Effects" on page 67 in Chapter 2, "Shared Concepts."

You can specify the following *effect-types*:

COLLECTION specifies a collection effect that defines one or more variables as a single effect that has multiple degrees of freedom. The variables in a collection are considered as a unit for purposes of estimation and inference.

MULTIMEMBER | MM specifies a multimember classification effect whose levels are determined by one or more variables that appear in a CLASS statement.

POLYNOMIAL | POLY specifies a multivariate polynomial effect in the specified numeric variables.

SPLINE specifies a regression spline effect whose columns are univariate spline expansions of one or more variables. A spline expansion replaces the original variable with an expanded or larger set of new variables.

Table 27.5 summarizes the *options* available in the EFFECT statement.

Table 27.5 EFFECT Statement Options

Option	Description
Collection Effects Options	
DETAILS	Displays the constituents of the collection effect
Multimember Effects Options	
DETAILS	Displays the levels of the multimember effect
NOEFFECT	Specifies that observations whose levels are all missing for the multimember variables should have 0 values in the corresponding design matrix columns
STDIZE	Standardizes the design matrix entries so that each observation has a sum of 1
WEIGHT=	Specifies the weight variable for the contributions of each classification effect
Polynomial Effects Options	
DEGREE=	Specifies the degree of the polynomial
DETAILS	Displays details of the specified polynomial
MDEGREE=	Specifies the maximum degree of any variable in a term of the polynomial
NOSEPARATE	Treats the polynomial as a single effect with multiple degrees of freedom
STANDARDIZE=	Specifies centering and scaling suboptions for the variables that define the polynomial

Table 27.5 *continued*

Option	Description
Spline Effects Options	
BASIS=	Specifies the type of basis (B-spline basis or truncated power function basis) for the spline effect
DATABOUNDARY	Uses the extremes of the data as boundary knots for a B-spline basis
DEGREE=	Specifies the degree of the spline effect
DETAILS	Displays the knots and locations for each spline basis function
KNOTMAX=	Requests equally spaced right-side boundary knots starting at the variables' maximum and ending at the KNOTMAX= value
KNOTMETHOD=	Specifies how to construct the knots for the spline effect
KNOTMIN=	Requests equally spaced left-side boundary knots starting at the KNOTMIN= value and ending at the variables' minimum value
NATURALCUBIC	Specifies a natural cubic spline basis for the spline effect
SEPARATE	Treats the spline basis for each variable as a separate effect when multiple variables are specified
SPLIT	Treats each design matrix column as a separate effect for selection methods

For more information about the syntax of these *effect-types* and how columns of constructed effects are computed, see the section "EFFECT Statement" on page 18 in Chapter 2, "Shared Concepts."

MODEL Statement

MODEL *response-variables* **=** *predictor-effects* < / *options* > **;**

The MODEL statement names the responses and the predictors, which determine, respectively, the **Y** and **X** matrices of the model. You can simply list the names of the predictor variables as the model effects, but you can also specify other types of effects, including polynomial effects and interactions. For information about constructing the model effects, see the section "Specification and Parameterization of Model Effects" on page 64 in Chapter 2, "Shared Concepts."

The MODEL statement is required. You can specify only one MODEL statement.

You can specify the following *options* after a slash (/).

INTERCEPT
overrides the default, in which the responses and predictors are centered. When responses and predictors are centered, no intercept is required in the model.

SOLUTION
lists the coefficients of the final predictive model for the responses. The coefficients for predicting the centered and scaled responses that are based on the centered and scaled predictors are displayed, in addition to the coefficients for predicting the raw responses that are based on the raw predictors.

OUTPUT Statement

OUTPUT OUT=*libref.data-table*
< **COPYVARS=***(variables)* >
< *keyword* < **=***prefix* > > . . . < *keyword* < **=***prefix* > > **;**

The OUTPUT statement creates a data table that contains observationwise statistics, which are computed after fitting the model. If you do not specify any *keywords*, then only the predicted values for responses are included.

The variables in the input data table are *not* included in the output data table, in order to avoid data duplication for large data tables; however, variables that you specify in the COPYVARS= option are included.

You must specify the following option:

OUT=*libref.data-table*
names the output data table for PROC PLSMOD to use. You must specify this option before any other options. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to where the data table is to be stored, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 1382.

data-table specifies the name of the output data table.

You can also specify the following syntax elements:

COPYVAR=*variable*
COPYVARS=(*variables***)**
copies one or more *variables* from the input data table to the output data table.

keyword **< =***prefix* **>**
specifies a statistic to include in the output data table and optionally a *prefix* for naming the output variables. If you do not provide a *prefix*, the PLSMOD procedure assigns a default prefix based on the type of statistic requested. For example, for response variables y1 and y2, a specification of PREDICTED produces two predicted value variables, Pred_y1 and Pred_y2.

You can specify the following *keywords* to add statistics to the OUTPUT data table:

H
requests the approximate leverage. The default prefix is H.

PREDICTED
PRED
P
requests predicted values for each response. The default prefix is Pred.

PRESS
requests approximate predicted residuals for each response. The default prefix is PRESS.

ROLE
requests numeric values that indicate the role played by each observation in fitting the model. The default prefix is _ROLE_. Table 27.6 shows the interpretation of this variable for each observation.

Table 27.6 Role Interpretation

Value	Observation Role
0	Not used
1	Training
2	Testing

If you do not partition the input data by using a PARTITION statement, then the role variable value is 1 for observations that are used in fitting the model, and 0 for observations that have at least one missing or invalid value for the responses or predictors.

STDX
requests standardized (centered and scaled) predictor values for each predictor. The default prefix is StdX.

STDXSSE
requests the sum of squares of residuals for standardized predictors. The default prefix is StdXSSE.

STDY
requests standardized (centered and scaled) response values for each response. The default prefix is StdY.

STDYSSE
requests the sum of squares of residuals for standardized responses. The default prefix is StdYSSE.

TSQUARE
T2
requests a scaled sum of squares of score values. The default prefix is TSquare.

XRESIDUAL
XRESID
XR
requests residuals for each predictor. The default prefix is XResid.

XSCORE
requests extracted factors (X-scores, latent vectors, latent variables, and T) for each selected model factor. The default prefix is XScore.

YRESIDUAL
YRESID
YR
requests residuals for each response. The default prefix is YResid.

YSCORE
requests extracted responses (Y-scores and U) for each selected model factor. The default prefix is YScore.

The output variables that contain the requested statistic are named as follows, according to the *keyword* that you specify:

- The *keywords* XRESIDUAL and STDX define an output variable for each predictor, so the variables that correspond to each predictor are named by appending a number (which starts from 1) to the prefix. For each defined variable, a label is also generated automatically; the label contains the prefix of the variable and the name of the predictor. For example, if the model has three predictors, then a specification of XRESIDUAL=XR produces the variables XR1, XR2, and XR3.
- The *keywords* PREDICTED, YRESIDUAL, STDY, and PRESS define an output variable for each response, so the variables that correspond to each response are named by appending an underscore followed by the name of the response variable to the prefix. For example, if the model has response variables y1 and y2, then a specification of PREDICTED=P produces the variables P_y1 and P_y2.
- The *keywords* XSCORE and YSCORE define an output variable for each selected model factor, so the variables that correspond to each successive factor are named by appending the factor number to the prefix. For example, if the model has three selected factors, then a specification of XSCORE=T produces the variables T1, T2, and T3.
- The *keywords* H, TSQUARE, STDXSSE, STDYSSE, and ROLE each define a single output variable, so the variable name matches the prefix.

PARTITION Statement

PARTITION *partition-options* **;**

The PARTITION statement specifies how to logically partition observations in the input data table into disjoint subsets for model training and testing. Either you can designate a variable in the input data table and a set of formatted values of that variable to determine the role of each observation, or you can specify proportions to use for random assignment of observations to each role.

You must specify exactly one of the following *partition-options*:

FRACTION(<TEST=*fraction***> <SEED=***number***>)**
requests that specified proportions of the observations in the input data table be randomly assigned training and testing roles. You specify the proportions for testing by using the TEST= suboption; the specified fraction must be less than 1 and the remaining fraction of the observations are assigned to the training role. If you do not specify the TEST= suboption, all observations are assigned to the training role. The SEED= suboption specifies an integer that is used to start the pseudorandom number generator for random partitioning of data for training and testing. If you do not specify SEED=*number* or if *number* is less than or equal to 0, the seed is generated by reading the time of day from the computer's clock.

Because *fraction* is a per-observation probability (which means that any particular observation has a probability of *fraction* of being assigned the testing role), using the FRACTION option can cause different numbers of observations to be assigned training and testing roles. You can specify the SEED= suboption to create the same partition data tables for a particular number of compute nodes. However, changing the number of compute nodes changes the initial distribution of data, resulting in different partition data tables.

ROLEVAR | ROLE=*variable* **(<TEST='***value***'> <TRAIN='***value***'>)**
names the *variable* in the input data table whose values are used to assign roles to each observation. This *variable* cannot also appear as an analysis variable in other statements or options. The TEST= and TRAIN= suboptions specify the formatted values of this variable that are used to assign observation roles. If you specify only the TEST= suboption, then all observations whose role is not determined by the TEST= suboption are assigned to training.

For more information, see the section "Test Set Validation" on page 1407. For an illustration, see Example 27.1.

Details: PLSMOD Procedure

Regression Methods

All the predictive methods that PROC PLSMOD implements work essentially by finding linear combinations of the predictors (factors) to use to predict the responses linearly. The methods differ only in how the factors are derived, as explained in the following sections.

Partial Least Squares

Partial least squares (PLS) works by extracting one factor at a time. Let $\mathbf{X} = \mathbf{X}_0$ be the centered and scaled matrix of predictors, and let $\mathbf{Y} = \mathbf{Y}_0$ be the centered and scaled matrix of response values. The PLS method starts with a linear combination $\mathbf{t} = \mathbf{X}_0\mathbf{w}$ of the predictors, where $\mathbf{t}$ is called a *score vector* and $\mathbf{w}$ is its associated *weight vector*. The PLS method predicts both $\mathbf{X}_0$ and $\mathbf{Y}_0$ by regression on $\mathbf{t}$:

$$\begin{aligned} \hat{\mathbf{X}}_0 &= \mathbf{t}\mathbf{p}', \quad \text{where} \quad \mathbf{p}' = (\mathbf{t}'\mathbf{t})^{-1}\mathbf{t}'\mathbf{X}_0 \\ \hat{\mathbf{Y}}_0 &= \mathbf{t}\mathbf{c}', \quad \text{where} \quad \mathbf{c}' = (\mathbf{t}'\mathbf{t})^{-1}\mathbf{t}'\mathbf{Y}_0 \end{aligned}$$

The vectors $\mathbf{p}$ and $\mathbf{c}$ are called the X- and Y-*loadings*, respectively.

The specific linear combination $\mathbf{t} = \mathbf{X}_0\mathbf{w}$ is the combination that has maximum covariance $\mathbf{t}'\mathbf{u}$ with some response linear combination $\mathbf{u} = \mathbf{Y}_0\mathbf{q}$. Another characterization is that the X-weight, $\mathbf{w}$, and the Y-weight, $\mathbf{q}$, are proportional to the first left- and right-singular vectors, respectively, of the covariance matrix $\mathbf{X}_0'\mathbf{Y}_0$ or, equivalently, the first eigenvectors of $\mathbf{X}_0'\mathbf{Y}_0\mathbf{Y}_0'\mathbf{X}_0$ and $\mathbf{Y}_0'\mathbf{X}_0\mathbf{X}_0'\mathbf{Y}_0$, respectively.

This accounts for how the first PLS factor is extracted. The second factor is extracted in the same way by replacing $\mathbf{X}_0$ and $\mathbf{Y}_0$ with the X- and Y-residuals from the first factor:

$$\begin{aligned}\mathbf{X}_1 &= \mathbf{X}_0 - \hat{\mathbf{X}}_0 \\ \mathbf{Y}_1 &= \mathbf{Y}_0 - \hat{\mathbf{Y}}_0\end{aligned}$$

These residuals are also called the *deflated* $\mathbf{X}$ and $\mathbf{Y}$ blocks. The process of extracting a score vector and deflating the data matrices is repeated for as many extracted factors as you want.

SIMPLS

The fact that each extracted PLS factor is defined in terms of different X-variables $\mathbf{X}_i$ leads to difficulties in comparing different scores, weights, and so on. The SIMPLS method of De Jong (1993) overcomes these difficulties by computing each score $\mathbf{t}_i = \mathbf{X}\mathbf{r}_i$ in terms of the original (centered and scaled) predictors $\mathbf{X}$. The SIMPLS X-weight vectors r_i are similar to the eigenvectors of $\mathbf{S}\mathbf{S}' = \mathbf{X}'\mathbf{Y}\mathbf{Y}'\mathbf{X}$, but they satisfy a different orthogonality condition. The $\mathbf{r}_1$ vector is just the first eigenvector $\mathbf{e}_1$ (so that the first SIMPLS score is the same as the first PLS score). However, the second eigenvector maximizes

$$\mathbf{e}_1'\mathbf{S}\mathbf{S}'\mathbf{e}_2 \quad \text{subject to} \quad \mathbf{e}_1'\mathbf{e}_2 = 0$$

whereas the second SIMPLS weight $\mathbf{r}_2$ maximizes

$$\mathbf{r}_1'SS'\mathbf{r}_2 \quad \text{subject to} \quad \mathbf{r}_1'\mathbf{X}'\mathbf{X}\mathbf{r}_2 = \mathbf{t}_1'\mathbf{t}_2 = 0$$

The SIMPLS scores are identical to the PLS scores for one response but slightly different for more than one response; see De Jong (1993) for details. The X- and Y-loadings are defined as in PLS, but it is easy to compute the overall model coefficients $\mathbf{B}$ because the scores are all defined in terms of $\mathbf{X}$:

$$\begin{aligned}\hat{\mathbf{Y}} &= \sum_i \mathbf{t}_i\mathbf{c}_i' \\ &= \sum_i \mathbf{X}\mathbf{r}_i\mathbf{c}_i' \\ &= \mathbf{X}\mathbf{B}, \text{ where } \mathbf{B} = \mathbf{R}\mathbf{C}'\end{aligned}$$

Principal Component Regression

Like the SIMPLS method, principal component regression (PCR) defines all the scores in terms of the original (centered and scaled) predictors $\mathbf{X}$. However, unlike both the PLS and SIMPLS methods, the PCR method chooses the X-weights and X-scores without regard to the response data. The X-scores are chosen to explain as much variation in $\mathbf{X}$ as possible; equivalently, the X-weights for the PCR method are the eigenvectors of the predictor covariance matrix $\mathbf{X}'\mathbf{X}$. Again, the X- and Y-loadings are defined as in PLS; but, as in SIMPLS, it is easy to compute overall model coefficients for the original (centered and scaled) responses $\mathbf{Y}$ in terms of the original predictors $\mathbf{X}$.

Reduced Rank Regression

As discussed in the preceding sections, partial least squares depends on selecting the factors $\mathbf{t} = \mathbf{X}\mathbf{w}$ of the predictors and $\mathbf{u} = \mathbf{Y}\mathbf{q}$ of the responses that have maximum covariance, whereas principal component regression effectively ignores $\mathbf{u}$ and selects $\mathbf{t}$ to have maximum variance, subject to orthogonality constraints. In contrast, reduced rank regression selects $\mathbf{u}$ to account for as much variation in the *predicted* responses as possible, effectively ignoring the predictors for the purpose of factor extraction. In reduced rank regression, the Y-weights, $\mathbf{q}_i$, are the eigenvectors of the covariance matrix $\hat{\mathbf{Y}}'_{\mathrm{LS}}\hat{\mathbf{Y}}_{\mathrm{LS}}$ of the responses that are predicted by ordinary least squares regression, and the X-scores are the projections of the Y-scores, $\mathbf{Y}\mathbf{q}_i$, onto the X space.

Relationships between Methods

(View the complete code for this example at https://github.com/sascommunities/doc-supplement-casstat/tree/main/Examples/plsmodgs2.sas.)

When you develop a predictive model, it is important to consider not only the explanatory power of the model for current responses, but also how well the predictive functions are sampled, because the sampling affects how well the model can extrapolate to future observations. All the techniques that the PLSMOD procedure implements work by extracting successive factors (linear combinations of the predictors) that optimally address one or both of these two goals: explaining response variation and explaining predictor variation. In particular, principal component regression selects factors that explain as much predictor variation as possible, reduced rank regression selects factors that explain as much response variation as possible, and partial least squares balances the two objectives, seeking factors that explain both response and predictor variation.

To see the relationships between these methods, consider how each one extracts a single factor from the following artificial data table, artData, which consists of two predictors and one response:

```
data artData;
   input x1 x2 y;
   datalines;
    3.37651  2.30716        0.75615
    0.74193 -0.88845        1.15285
    4.18747  2.17373        1.42392
    0.96097  0.57301        0.27433
   -1.11161 -0.75225       -0.25410
   -1.38029 -1.31343       -0.04728
    1.28153 -0.13751        1.00341
   -1.39242 -2.03615        0.45518
    0.63741  0.06183        0.40699
   -2.52533 -1.23726       -0.91080
    2.44277  3.61077       -0.82590
```

```
;

data mylib.artData;
   set artData;
run;

proc plsmod data=mylib.artData nfac=1 method=rrr;
   model y = x1 x2;
run;

proc plsmod data=mylib.artData nfac=1 method=pcr;
   model y = x1 x2;
run;

proc plsmod data=mylib.artData nfac=1 method=pls;
   model y = x1 x2;
run;
```

The amount of model and response variation that are explained by the first factor for each method is shown in Figure 27.10 through Figure 27.12.

Figure 27.10 Variation Explained by the First Reduced Rank Regression Factor

The PLSMOD Procedure

Percentage Variation Accounted for by Reduced Rank Regression Factors

	Model Effects		Response Variables	
Number of Extracted Factors	Current	Total	Current	Total
1	15.06605	15.06605	100.00000	100.00000

Figure 27.11 Variation Explained by the First Principal Component Regression Factor

The PLSMOD Procedure

Percentage Variation Accounted for by Principal Components

	Model Effects		Response Variables	
Number of Extracted Factors	Current	Total	Current	Total
1	92.99959	92.99959	9.37874	9.37874

Figure 27.12 Variation Explained by the First Partial Least Squares Regression Factor

The PLSMOD Procedure

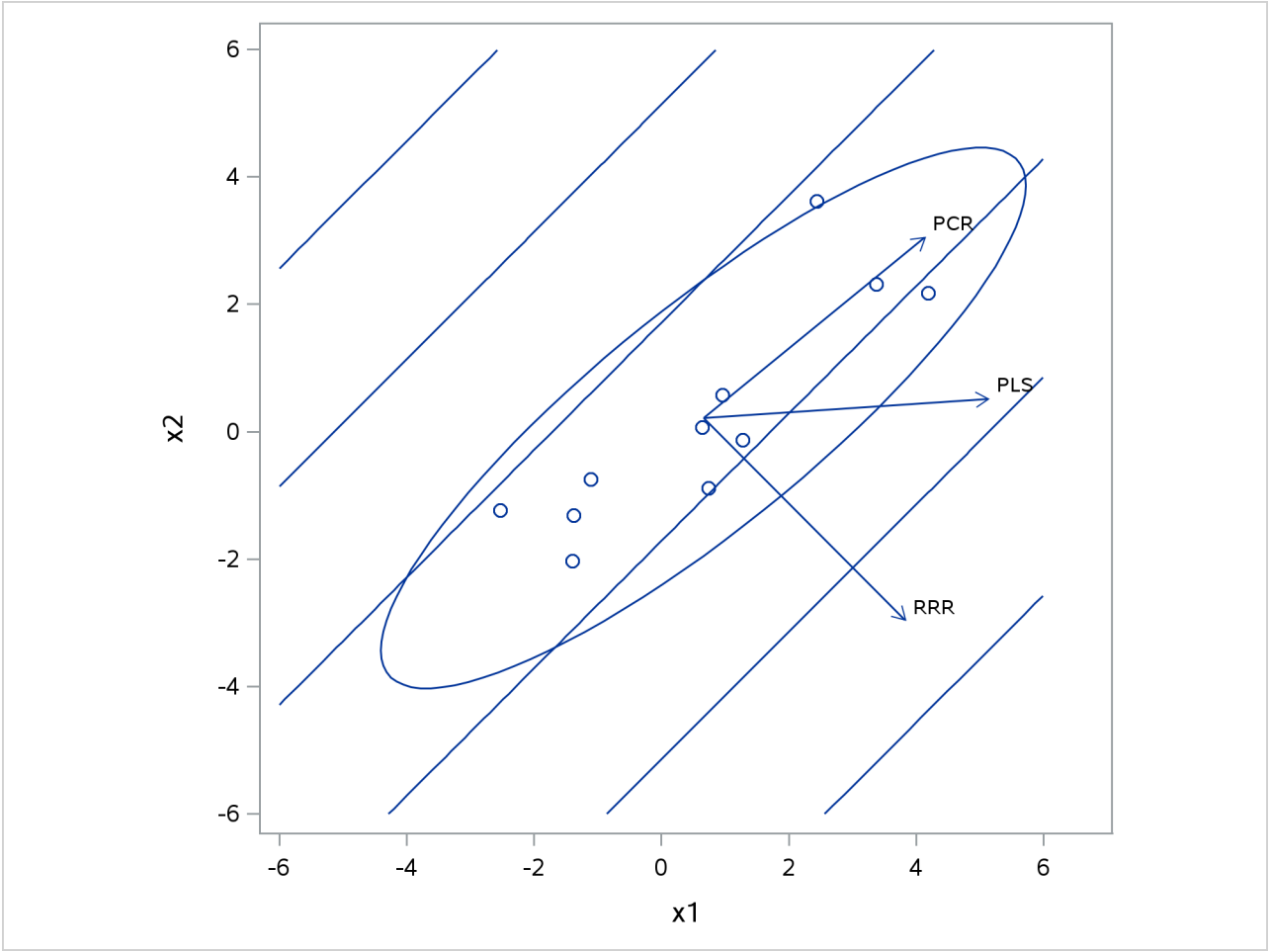
Percentage Variation Accounted for by Partial Least Squares Factors

	Model Effects		Response Variables	
Number of Extracted Factors	Current	Total	Current	Total
1	88.53567	88.53567	26.53038	26.53038

Notice that although the first reduced rank regression factor explains *all* of the response variation, it accounts for only about 15% of the predictor variation. In contrast, the first principal component regression factor accounts for most of the predictor variation (93%) but only 9% of the response variation. The first partial least squares factor accounts for only slightly less predictor variation than principal components but about three times as much response variation.

Figure 27.13 illustrates how partial least squares balances the goals of explaining response and predictor variation in this case.

Figure 27.13 Depiction of the First Factors for Three Different Regression Methods



The ellipse shows the general shape of the 11 observations in the predictor space, with the contours of increasing y overlaid. Also shown are the directions of the first factor for each of the three methods. Notice that although the predictors vary most in the x1 = x2 direction, the response changes most in the orthogonal x1 = –x2 direction. This explains why the first principal component accounts for little variation in the response and why the first reduced rank regression factor accounts for little variation in the predictors. The direction of the first partial least squares factor represents a compromise between the other two directions.

Test Set Validation

None of the regression methods that the PLSMOD procedure implements fit the observed data any better than ordinary least squares (OLS) regression; in fact, all the methods approach OLS as more factors are extracted. Basing the model on more extracted factors improves the model fit to the observed data, but extracting too many factors can cause *overfitting*—that is, tailoring the model too much to the current data to the detriment of future predictions. So the crucial point is that when there are many predictors, OLS can *overfit* the observed data; biased regression methods that use fewer extracted factors can provide better predictability of *future* observations. However, as the preceding observations imply, the quality of the observed data fit cannot be used to choose the number of factors to extract; the number of extracted factors must be chosen on the basis of how well the model fits observations that are not involved in the modeling procedure itself.

PROC PLSMOD implements the *test set validation* method of choosing the number of extracted factors. When you have sufficient data, you can subdivide your data into two parts: training data and test data. During the validation process, the model is fit on the training data, and the predicted residual sum of squares (PRESS) for models that have different numbers of extracted factors is found by using the test data. The number of factors chosen is usually the one that minimizes PRESS.

You use the PARTITION statement to logically subdivide the DATA= data table into separate roles. You can name the fractions of the data that you want to reserve as training data and test data. For example, the following statements randomly subdivide the inData data table, reserving 50% each for training and testing:

```
proc plsmod data=mylib.inData;
   partition fraction(test=0.5);
   ...
run;
```

In some cases you might need to exercise more control over the partitioning of the input data table. You can do this by naming both a variable in the input data table and a formatted value of that variable for each role. For example, the following statements assign roles to the observations in the inData data table based on the value of the variable Group in that data table. Observations whose value of Group is `'group 1'` are assigned for training, and those whose value is `'group 2'` are assigned to testing. All other observations are ignored.

```
proc plsmod data=mylib.inData;
   partition roleVar=Group(train='group 1' test='group 2')
   ...
run;
```

By default, the number of extracted factors is chosen to be the one that minimizes PRESS. However, models that have fewer factors often have PRESS statistics that are only marginally larger than the absolute minimum. To address this, Van der Voet (1994) proposed a statistical test for comparing the predicted residuals from

different models; when you apply van der Voet's test, the number of factors chosen is the fewest while still producing residuals that are insignificantly larger than the residuals of the model that has a minimum PRESS.

To see how van der Voet's test works, let $R_{i,jk}$ be the jth predicted residual for response k for the model that has i extracted factors. Then, the PRESS statistic is $\sum_{jk} R^2_{i,jk}$. Also, let $i_{\min}$ be the number of factors for which PRESS is minimized. The critical value for van der Voet's test is based on the differences between squared predicted residuals:

$$D_{i,jk} = R^2_{i,jk} - R^2_{i_{\min},jk}$$

One alternative for the critical value is $C_i = \sum_{jk} D_{i,jk}$, which is simply the difference between the PRESS statistics for i and $i_{\min}$ factors; alternatively, van der Voet suggests Hotelling's T^2 statistic $C_i = \mathbf{d}'_{i,.}\mathbf{S}_i^{-1}\mathbf{d}_{i,.}$, where $\mathbf{d}_{i,.}$ is the sum of the vectors $\mathbf{d}_{i,j} = \{D_{i,j1}, \ldots, D_{i,jN_y}\}'$ and $\mathbf{S}_i$ is the sum of squares and crossproducts matrix,

$$\mathbf{S}_i = \sum_j \mathbf{d}_{i,j}\mathbf{d}'_{i,j}$$

Theoretically, the significance level for van der Voet's test is obtained by comparing C_i with the distribution of values that result from randomly exchanging $R^2_{i,jk}$ and $R^2_{i_{\min},jk}$. In practice, a Monte Carlo sample of such values is simulated and the significance level is approximated as the proportion of simulated critical values that are greater than C_i. If you apply van der Voet's test by specifying the CVTEST option, then, by default, the number of extracted factors that are chosen is the smallest number of factors that have an approximate significance level that is greater than 0.1.

Centering and Scaling

By default, the predictors and the responses are centered and scaled to have mean 0 and standard deviation 1. Centering the predictors and the responses ensures that the criterion for choosing successive factors is based on how much *variation* they explain in either the predictors or the responses or in both. (For more information about how different methods explain variation, see the section "Regression Methods" on page 1402.) Without centering, both the value of the mean variable and the variation around that mean are involved in selecting factors. Scaling serves to place all predictors and responses on an equal footing relative to their variation in the data. For example, if Time and Temp are two of the predictors, then scaling says that a change of std(Time) in Time is approximately equivalent to a change of std(Temp) in Temp.

Usually, both the predictors and responses should be centered and scaled. However, if their values already represent variation around a nominal or target value, then you can use the NOCENTER option in the PROC PLSMOD statement to suppress centering. Likewise, if the predictors or responses are already all on comparable scales, then you can use the NOSCALE option to suppress scaling.

If the predictors involve crossproduct terms, PROC PLSMOD does not standardize the variables before it standardizes the crossproduct. That is, if the *i*th values of two predictors are denoted x_i^1 and x_i^2, then the default standardized *i*th value of the crossproduct is

$$\frac{x_i^1 x_i^2 - \text{mean}_j(x_j^1 x_j^2)}{\text{std}_j(x_j^1 x_j^2)}$$

When test set validation is performed for the number of effects, some practitioners disagree as to whether the training data should be retransformed. By default, PROC PLSMOD does retransform the training data, but you can suppress this behavior by specifying the NOCVSTDIZE option in the PROC PLSMOD statement.

Missing Values

Observations that have any missing independent variables (including all classification variables) are excluded from the analysis, and no predictions are computed for such observations. However, if you specify the MISSING option in the CLASS statement, missing values are treated as valid values for the classification variable. Observations that have no missing independent variables but do have missing response variables are also excluded from the analysis, but predictions are computed. If you use the PARTITION statement and specify the ROLEVAR= option, observations that contain missing values for the ROLEVAR= variable are excluded from the analysis, but predictions are computed for them.

Displayed Output

The following sections describe the output that PROC PLSMOD produces. The output is organized into various tables, which are discussed in the order of their appearance.

Model Information

The "Model Information" table displays basic information about the model, such as the input data table, the factor extraction method, the validation method, and the type of parameterization used for classification variables that are named in the CLASS statement. If you use the PARTITION statement, the table also displays the random number seed for partitioning, the validation testing criterion, the number of random permutations, and the random number seed for permutation, depending on whether you specify the FRACTION option in the PARTITION statement and the CVTEST option in the PROC PLSMOD statement.

Dimensions

The "Dimensions" table displays information about the number of response variables, the number of effects, and the number of predictor parameters. It also displays the number of factors to extract.

Number of Observations

The "Number of Observations" table displays the number of observations that are read from the input data table and the number of observations that are used in the analysis. If you use a PARTITION statement, the table also displays the number of observations that are used for each data role.

Class Level Information

The "Class Level Information" table lists the levels of every variable that is specified in the CLASS statement. You should check this information to make sure that the data are correct. You can adjust the order of the CLASS variable levels by specifying the ORDER= option in the CLASS statement. You can suppress the "Class Level Information" table completely or partially by specifying the NOCLPRINT= option in the PROC PLSMOD statement.

If the classification variables use a nonsingular parameterization, the "Class Level Information" table also displays the reference value for each variable.

Centering and Scaling Information

If you specify the CENSCALE option in the PROC PLSMOD statement, the PLSMOD procedure produces "Model Effect Centering and Scaling" and "Response Variable Centering and Scaling" tables, which display the centering and scaling information for each predictor and response.

Test Set Validation

If you use the PARTITION statement to perform a test set validation for choosing the number of extracted factors, the PLSMOD procedure produces a "Test Set Validation Residual Summary" table to display a residual summary of the validation for each number of factors. It also produces a "Test Set Validation Results" table to display information about the optimal number of factors.

Percentage Variation Accounted for by Extracted Factors

By default, the PLSMOD procedure produces the "Percentage Variation Accounted for by Extracted Factors" table to display the amount of predictor variation and response variation that are accounted for by each factor. If you specify the VARSS option in the PROC PLSMOD statement, the PLSMOD procedure also produces the "Model Effect Percentage Variation Accounted for by Extracted Factors" table and the "Response Variable Percentage Variation Accounted for by Extracted Factors" table to display the amount of variation that is accounted for in each predictor and response, in addition to the average predictor and response sum of squares that are accounted for by each successive factor.

Model Details

If you specify the DETAILS option in the PROC PLSMOD statement, the PLSMOD procedure produces tables to display details about the fitted model for each successive factor. These tables include the following:

- "Model Effect Loadings" table, which displays the predictor loadings
- "Model Effect Weights" table, which displays predictor weights
- "Response Variable Weights" table, which displays the response weights

- "Coded Regression Coefficients" tables, which display the coded regression coefficients, if you specify METHOD=SIMPLS, METHOD=PCR, or METHOD=RRR in the PROC PLSMOD statement.

Parameter Estimates

If you specify the SOLUTION option in the MODEL statement, the PLSMOD procedure produces a "Parameter Estimates" table to display the coefficients of the final predictive model for the responses. The coefficients for predicting the centered and scaled responses that are based on the centered and scaled predictors are displayed, in addition to the coefficients for predicting the raw responses based on the raw predictors.

Timing Information

The "Timing" table displays the elapsed time of each main task of the procedure.

OutputCasTables Table

The OutputCasTables table is a special table that has information about each CAS table that is created during a CAS action execution. The information for each CAS table consists of the CAS table name, the caslib in which the table resides, and the number of columns and rows in the CAS table. Because this table is not a typical ODS table that contains analytical results, you cannot include it in the *table-spec-list* in the DISPLAYOUT statement.

ODS Table Names

Each table that the PLSMOD procedure creates has a name associated with it. You must use this name to refer to the table when you use the DISPLAY statement, the DISPLAYOUT statement, or the Output Delivery System (ODS) statements. These names are listed in Table 27.7.

NOTE: The EFFECT statement also create tables which are not listed in this section. For information about these tables, see "ODS Table Names" on page 33 in Chapter 2, "Shared Concepts."

Table 27.7 ODS Tables Produced by PROC PLSMOD

Table Name	Description	Required Statement and Option
CVResults	Results of test set validation	PARTITION statement
CenScaleParms	Parameter estimates for centered and scaled data	SOLUTION option in MODEL statement
ClassInfo	Level information from the CLASS statement	CLASS statement
CodedCoef	Coded regression coefficients	DETAILS option in PROC PLSMOD statement
Dimensions	Model dimensions	Default output
ModelInfo	Model information	Default output
NObs	Number of observations read and used	Default output
OutputCasTables	See the section "OutputCasTables Table" on page 1411	DISPLAYOUT statement or OUTPUT statement

Table 27.7 *continued*

Table Name	Description	Required Statement and Option
ParameterEstimates	Parameter estimates for raw data	SOLUTION option in MODEL statement
PercentVariation	Predictor and response variation that are accounted for by each factor	Default output
ResidualSummary	Residual summary from test set validation	PARTITION statement
Timing	Absolute and relative times spent by tasks that are performed by the procedure	Default output
XEffectCenScale	Centering and scaling information for predictor effects	CENSCALE option in PROC PLSMOD statement
XLoadings	Loadings for predictor effects	DETAILS option in PROC PLSMOD statement
XPercentVariation	Variation that is accounted for by each factor for predictor effects	VARSS option in PROC PLSMOD statement
XWeights	Weights for predictor effects	DETAILS option in PROC PLSMOD statement
YPercentVariation	Variation that is accounted for by each factor for responses	VARSS option in PROC PLSMOD statement
YVariableCenScale	Centering and scaling information for responses	CENSCALE option in PROC PLSMOD statement
YWeights	Weights for responses	DETAILS option in PROC PLSMOD statement

Examples: PLSMOD Procedure

The examples in this chapter are available in the GitHub repository located at `https://github.com/sascommunities/doc-supplement-casstat`.

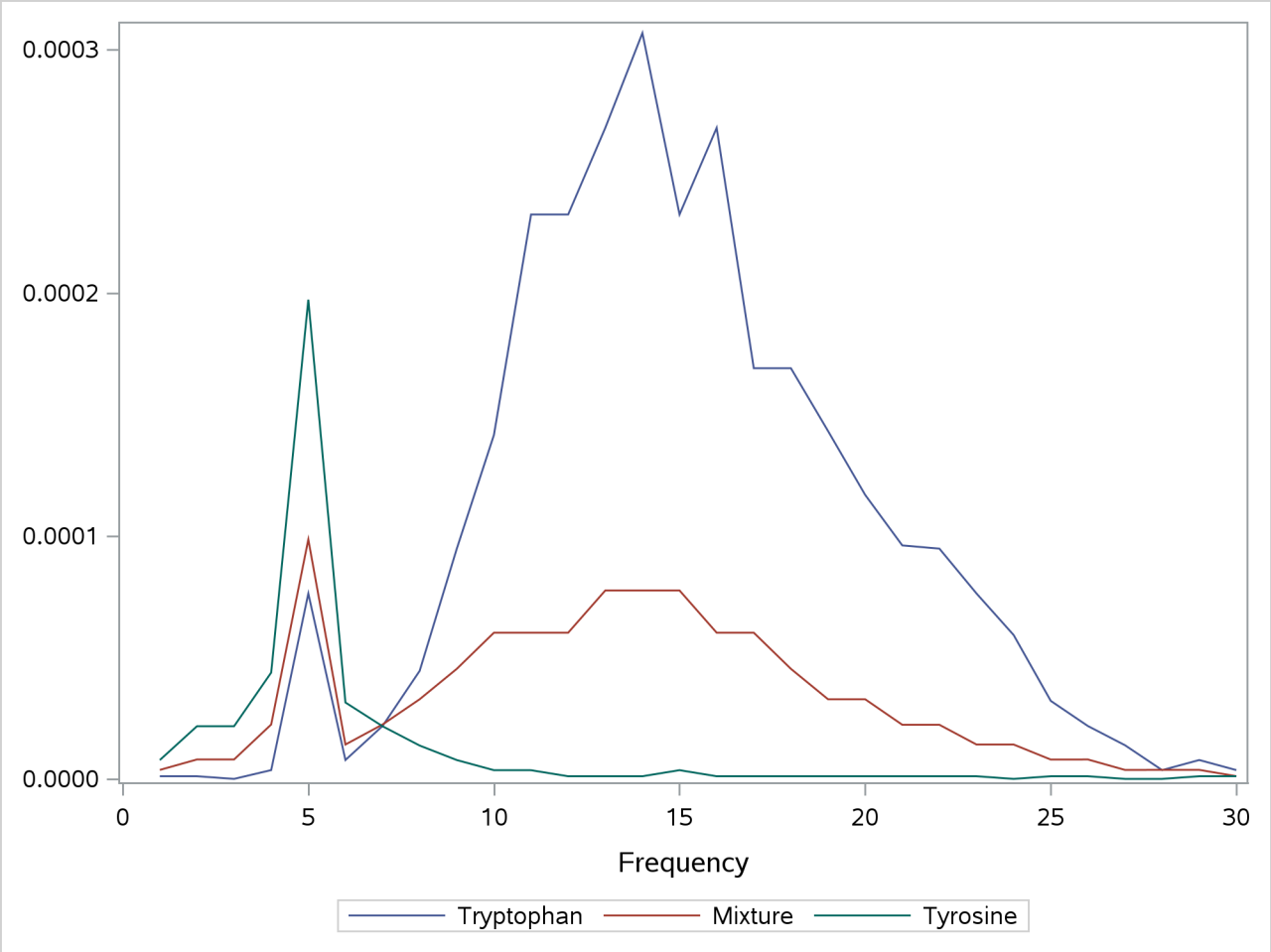
NOTE: Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections "Using CAS Sessions and CAS Engine Librefs" on page 7 and "Loading a SAS Data Set onto a CAS Server" on page 7 in Chapter 2, "Shared Concepts."

Example 27.1: Choosing a PLS Model by Test Set Validation

(View the complete code for this example (plsmodex1.sas) in the example repository.)

This example demonstrates issues in spectrometric calibration. The data (Umetrics 1995) consist of spectrographic readings on 33 samples that contain known concentrations of two amino acids, tyrosine and tryptophan. The spectra are measured at 30 frequencies across the overall range of frequencies. For example, Output 27.1.1 shows the observed spectra for three samples: one with only tryptophan, one with only tyrosine, and one with a mixture of the two, all at a total concentration of 10^{-6}.

Output 27.1.1 Spectra for Three Samples of Tyrosine and Tryptophan



Of the 33 samples, 18 are used as a training set and 15 as a test set. The data originally appear in McAvoy et al. (1989).

These data were created in a lab, where the concentrations are fixed in order to provide a wide range of applicability for the model. This example uses a linear function of the logarithms of the spectra to predict the logarithms of tyrosine and tryptophan concentration and the logarithm of the total concentration. Actually, because zeros are possible in both the responses and the predictors, slightly different transformations are used. The following statements create a data table named ex1Data for these data. The data table also contains a variable Role that is used to assign samples to the training and testing roles.

```
data ex1Data;
   input obsnam $ Role : $5. tot tyr f1-f30 @@;
   try = tot - tyr;
   if (tyr) then tyr_log = log10(tyr); else tyr_log = -8;
   if (try) then try_log = log10(try); else try_log = -8;
   tot_log = log10(tot);
   datalines;
17mix35 TRAIN 0.00003 0
 -6.215 -5.809 -5.114 -3.963 -2.897 -2.269 -1.675 -1.235
 -0.900 -0.659 -0.497 -0.395 -0.335 -0.315 -0.333 -0.377
 -0.453 -0.549 -0.658 -0.797 -0.878 -0.954 -1.060 -1.266
 -1.520 -1.804 -2.044 -2.269 -2.496 -2.714
19mix35 TRAIN 0.00003 3E-7
 -5.516 -5.294 -4.823 -3.858 -2.827 -2.249 -1.683 -1.218
 -0.907 -0.658 -0.501 -0.400 -0.345 -0.323 -0.342 -0.387
 -0.461 -0.554 -0.665 -0.803 -0.887 -0.960 -1.072 -1.272
 -1.541 -1.814 -2.058 -2.289 -2.496 -2.712
21mix35 TRAIN 0.00003 7.5E-7
 -5.519 -5.294 -4.501 -3.863 -2.827 -2.280 -1.716 -1.262
 -0.939 -0.694 -0.536 -0.444 -0.384 -0.369 -0.377 -0.421
 -0.495 -0.596 -0.706 -0.824 -0.917 -0.988 -1.103 -1.294
 -1.565 -1.841 -2.084 -2.320 -2.521 -2.729
23mix35 TRAIN 0.00003 1.5E-6

   ... more lines ...

 -5.138 -5.463 -5.461 -5.461 -5.461 -5.461
tyro2   TEST 0.0001 0.0001
 -1.081 -0.710 -0.470 -0.337 -0.327 -0.433 -0.602 -0.841
 -1.119 -1.423 -1.750 -2.121 -2.449 -2.818 -3.110 -3.467
 -3.781 -4.029 -4.241 -4.366 -4.501 -4.366 -4.501 -4.501
 -4.668 -4.668 -4.865 -4.865 -5.109 -5.111
;

data mylib.ex1Data;
   set ex1Data;
run;
```

The following statements fit a PLS model that has 10 factors:

```
proc plsmod data=mylib.ex1Data nfac=10;
   model tot_log tyr_log try_log = f1-f30;
run;
```

The "Model Information" table in Output 27.1.2 shows that no validation method is used. The "Number of Observations" table confirms that all 33 sample observations are used in the analysis.

Output 27.1.2 Model Information and Number of Observations

The PLSMOD Procedure

Model Information	
Data Source	EX1DATA
Factor Extraction Method	Partial Least Squares
PLS Algorithm	NIPALS
Validation Method	None

Number of Observations Read	33
Number of Observations Used	33

The table in Output 27.1.3 indicates that only four or five factors are required to explain almost all the variation in both the predictors and the responses.

Output 27.1.3 Amount of Variation Explained

Percentage Variation Accounted for by Partial Least Squares Factors

	Model Effects		Response Variables	
Number of Extracted Factors	**Current**	**Total**	**Current**	**Total**
1	77.67903	77.67903	47.80217	47.80217
2	20.62719	98.30622	38.96826	86.77043
3	1.00143	99.30766	6.89262	93.66305
4	0.24930	99.55696	1.77222	95.43528
5	0.13077	99.68773	1.71763	97.15290
6	0.08970	99.77742	0.58619	97.73909
7	0.05684	99.83426	0.29079	98.02988
8	0.06730	99.90156	0.13857	98.16845
9	0.01521	99.91676	0.68214	98.85059
10	0.02627	99.94304	0.14388	98.99447

In order to choose the optimal number of PLS factors, you can explore how well models that are based on data in training roles and have different numbers of factors fit the data in testing roles. To do so, you can use the PARTITION statement to assign observations to training and testing roles based on the values of the input variable named Role, as follows:

```
proc plsmod data=mylib.ex1Data nfac=10 cvtest(stat=press seed=12345);
   model tot_log tyr_log try_log = f1-f30;
   partition roleVar = Role(train='TRAIN' test='TEST');
run;
```

Output 27.1.4 shows the "Model Information" table and the "Number of Observations" table. The "Model Information" table indicates that test set validation is used, and it displays information about the options that are used in the model comparison test. The "Number of Observations" table confirms that there are 18 observations for the training role and 15 for the testing role.

Output 27.1.4 Model Information and Number of Observations with Test Set Validation

The PLSMOD Procedure

Model Information	
Data Source	EX1DATA
Factor Extraction Method	Partial Least Squares
PLS Algorithm	NIPALS
Validation Method	Test Set Validation
Validation Testing Criterion	Prob PRESS > 0.1
Number of Random Permutations	1000
Random Number Seed for Permutation	12345

Number of Observations Read	33
Number of Observations Used	33
Number of Observations Used for Training	18
Number of Observations Used for Testing	15

Output 27.1.5 displays the results of the test set validation. They indicate that although five PLS factors produce the minimum predicted residual sum of squares, the residuals for four factors are insignificantly different from the residuals for five factors. Thus, the smaller model is preferred.

Output 27.1.5 Test Set Validation for the Number of PLS Factors

The PLSMOD Procedure

Test Set Validation for the Number of Extracted Factors		
Number of Extracted Factors	**Root Mean PRESS**	**Prob > PRESS**
0	3.056797	<.0001
1	2.630561	<.0001
2	1.00706	0.0070
3	0.664603	0.0020
4	0.521578	0.3800
5	0.500034	1.0000
6	0.513561	0.5100
7	0.501431	0.6870
8	1.05579	0.1530
9	1.435085	0.1010
10	1.720389	0.0320

Minimum Root Mean PRESS	0.500034
Minimizing Number of Factors	5
Smallest Number of Factors with p > 0.1	4

Output 27.1.5 *continued*

Percentage Variation Accounted for by Partial Least Squares Factors				
	Model Effects		Response Variables	
Number of Extracted Factors	Current	Total	Current	Total
1	81.16545	81.16545	48.33854	48.33854
2	16.81131	97.97676	32.54654	80.88508
3	1.76391	99.74067	11.44380	92.32888
4	0.19507	99.93574	3.83631	96.16519

References

De Jong, S. (1993). "SIMPLS: An Alternative Approach to Partial Least Squares Regression." *Chemometrics and Intelligent Laboratory Systems* 18:251–263.

De Jong, S., and Kiers, H. (1992). "Principal Covariates Regression." *Chemometrics and Intelligent Laboratory Systems* 14:155–164.

Dijkstra, T. K. (1983). "Some Comments on Maximum Likelihood and Partial Least Squares Methods." *Journal of Econometrics* 22:67–90.

Dijkstra, T. K. (1985). *Latent Variables in Linear Stochastic Models: Reflections on Maximum Likelihood and Partial Least Squares Methods*. 2nd ed. Amsterdam: Sociometric Research Foundation.

Frank, I., and Friedman, J. (1993). "A Statistical View of Some Chemometrics Regression Tools." *Technometrics* 35:109–135.

Geladi, P., and Kowalski, B. (1986). "Partial Least-Squares Regression: A Tutorial." *Analytica Chimica Acta* 185:1–17.

Haykin, S. (1994). *Neural Networks: A Comprehensive Foundation*. New York: Macmillan.

Helland, I. S. (1988). "On the Structure of Partial Least Squares Regression." *Communications in Statistics—Simulation and Computation* 17:581–607.

Hoerl, A. E., and Kennard, R. W. (1970). "Ridge Regression: Biased Estimation for Non-orthogonal Problems." *Technometrics* 12:55–67.

Lindberg, W., Persson, J.-A., and Wold, S. (1983). "Partial Least-Squares Method for Spectrofluorimetric Analysis of Mixtures of Humic Acid and Ligninsulfonate." *Analytical Chemistry* 55:643–648.

McAvoy, T. J., Wang, N. S., Naidu, S., Bhat, N., Gunter, J., and Simmons, M. (1989). "Interpreting Biosensor Data via Backpropagation." *International Joint Conference on Neural Networks* 1:227–233.

Naes, T., and Martens, H. (1985). "Comparison of Prediction Methods for Multicollinear Data." *Communications in Statistics—Simulation and Computation* 14:545–576.

Ränner, S., Lindgren, F., Geladi, P., and Wold, S. (1994). "A PLS Kernel Algorithm for Data Sets with Many Variables and Fewer Objects." *Journal of Chemometrics* 8:111–125.

Sarle, W. S. (1994). "Neural Networks and Statistical Models." In *Proceedings of the Nineteenth Annual SAS Users Group International Conference*, 1538–1550. Cary, NC: SAS Institute Inc. https://support.sas.com/resources/papers/proceedings-archive/SUGI94/Sugi-94-255%20Sarle.pdf.

Shao, J. (1993). "Linear Model Selection by Cross-Validation." *Journal of the American Statistical Association* 88:486–494.

Tobias, R. D. (1995). "An Introduction to Partial Least Squares Regression." In *Proceedings of the Twentieth Annual SAS Users Group International Conference*, 1250–1257. Cary, NC: SAS Institute Inc. http://support.sas.com/rnd/app/stat/papers/pls.pdf.

Umetrics (1995). *Multivariate Analysis.* Three-day course. Winchester, MA: Umetrics.

Van den Wollenberg, A. L. (1977). "Redundancy Analysis: An Alternative to Canonical Correlation Analysis." *Psychometrika* 42:207–219.

Van der Voet, H. (1994). "Comparing the Predictive Accuracy of Models Using a Simple Randomization Test." *Chemometrics and Intelligent Laboratory Systems* 25:313–323.

Wold, H. (1966). "Estimation of Principal Components and Related Models by Iterative Least Squares." In *Multivariate Analysis*, edited by P. R. Krishnaiah, 391–420. New York: Academic Press.

Chapter 28
QTRSELECT Procedure

Contents

Overview: QTRSELECT Procedure **1420**
PROC QTRSELECT Features 1420
PROC QTRSELECT Compared with Other SAS Procedures 1421
Using CAS Sessions and CAS Engine Librefs 1422
Getting Started: QTRSELECT Procedure **1423**
Syntax: QTRSELECT Procedure **1431**
PROC QTRSELECT Statement 1432
BY Statement 1433
CLASS Statement 1433
CODE Statement 1434
DISPLAY Statement 1434
DISPLAYOUT Statement 1435
EFFECT Statement 1436
MODEL Statement 1438
OUTPUT Statement 1440
PARTITION Statement 1442
SELECTION Statement 1443
STORE Statement 1444
WEIGHT Statement 1444
Details: QTRSELECT Procedure **1445**
Quantile Regression 1445
Criteria Used in Model Selection 1447
Diagnostic Statistics 1450
Classification Variables and the SPLIT Option 1451
Using Validation and Test Data 1452
Computational Method 1455
Displayed Output 1455
ODS Table Names 1459
ODS Graphics 1460
Examples: QTRSELECT Procedure **1461**
Example 28.1: Simulation Study 1461
Example 28.2: Growth Charts for Body Mass Index 1463
Example 28.3: Pollution and Mortality 1466
References **1473**

Overview: QTRSELECT Procedure

The QTRSELECT procedure fits and performs model selection for quantile regression models in SAS Viya.

Quantile regression uses a linear function model to fit the quantiles of a response variable conditional on the explanatory variables. The model does not assume a particular parametric distribution for the response. The models that PROC QTRSELECT supports can contain main effects that consist of both continuous and classification variables and interaction effects of these variables. The models can also include constructed effects such as splines. The procedure offers a number of effect-selection methods, including stepwise methods. It also offers extensive capabilities for customizing the model selection by using a wide variety of selection and stopping criteria, including significance-level-based criteria, information criteria, and modern validation-based criteria. PROC QTRSELECT also provides a variety of quantile regression diagnostics that are conditional on the selected model.

PROC QTRSELECT Features

The main features of the QTRSELECT procedure are as follows:

- **Model specification**
  - supports quantile regression for single or multiple quantile levels
  - supports multiple parameterizations for classification effects
  - supports any degree of interactions (crossed effects) and nested effects
  - supports hierarchical model selection strategy among effects that have a containment relationship
  - supports partitioning of data into training, validation, and testing roles
  - provides a WEIGHT statement for weighted analysis
- **Selection control**
  - provides multiple effect-selection methods
  - offers selection of individual levels of classification effects
  - provides effect selection based on a variety of selection criteria
  - provides stopping rules based on a variety of model evaluation criteria
  - supports stopping and selection rules based on external validation
- **Display and output**
  - provides a CODE statement to produce SAS code that can score a new data set
  - produces output data tables that contain predicted values, residuals, and confidence limits
  - uses ODS Graphics to create model selection plots as part of its output. For more information about ODS Graphics, see the section "ODS Graphics" on page 1460

The QTRSELECT procedure supports the following effect-selection methods. For a more detailed description of these methods, see the section "SELECTION Statement" on page 42 in Chapter 2, "Shared Concepts."

- Forward selection starts with no effects in the model and adds effects.
- Backward elimination starts with all effects in the model and deletes effects.
- Stepwise regression is similar to forward selection, except that effects already in the model do not necessarily stay there.

Because the QTRSELECT procedure runs on CAS, it also does the following:

- enables you to run on a cluster of machines that distribute the data and the computations
- enables you to run in single-machine mode on CAS
- exploits all the available cores and concurrent threads. For information about how PROC QTRSELECT uses threads, see the section "Multithreading" on page 102 in Chapter 2, "Shared Concepts."

PROC QTRSELECT Compared with Other SAS Procedures

The QTRSELECT procedure provides quantile regression functionality that is comparable to that of the HPQUANTSELECT, QUANTSELECT, and QUANTREG procedures in SAS/STAT software.

PROC QTRSELECT Compared with the HPQUANTSELECT Procedure

The functionality of the QTRSELECT procedure closely resembles that of the HPQUANTSELECT procedure, which is a high-performance procedure. The QTRSELECT procedure is the next generation of the HPQUANTSELECT procedure, and it was developed specifically for SAS Viya. Both procedures are designed to run on a cluster of machines that distribute the data and the computations.

Both the QTRSELECT and HPQUANTSELECT procedures fit and perform effect selection for quantile regression models. The models can contain main effects that consist of both continuous and classification variables and interaction effects of these variables. The QTRSELECT procedure is additionally capable of constructing complex effects including univariate spline and polynomial expansions, and producing effect-selection plots by using ODS Graphics.

With the QTRSELECT and HPQUANTSELECT procedures, you request model selection by using the SELECTION statement. Both procedures offer the same methods of effect selection.

PROC QTRSELECT Compared with the QUANTSELECT Procedure

Both the QTRSELECT and QUANTSELECT procedures fit and perform model selection for quantile regression models. The models can contain main effects that consist of both continuous and classification variables and interaction effects of these variables. Both procedures support the forward, backward, and stepwise effect-selection methods and the ability to use separate validation and test data as specified in the PARTITION statement.

PROC QTRSELECT provides confidence limits and Wald tests for parameters, and prediction limits for quantiles.

PROC QUANTSELECT provides LASSO and adaptive LASSO effect-selection methods and effect selection for quantile process regression.

PROC QTRSELECT Compared with the QUANTREG Procedure

The QTRSELECT procedure chooses and fits quantile regression models. The QUANTREG procedure fits quantile regression models, but it does not offer model selection.

PROC QTRSELECT provides a variety of effect-selection methods, fit statistics, graphical effect-selection summary, and a cloud-enabled interior-point algorithm. It also supports partitioning the data into training, validation, and testing roles.

PROC QUANTREG provides simplex and smoothing algorithms, a rank-score test, a likelihood test, Markov-chain-marginal-bootstrap covariance estimation, many diagnostic and fit plots, and quantile process regression.

Using CAS Sessions and CAS Engine Librefs

SAS Cloud Analytic Services (CAS) is the analytic server and associated cloud services in SAS Viya. This section describes how to create a CAS session and set up a CAS engine libref that you can use to connect to the CAS session. It assumes that you have a CAS server already available; contact your system administrator if you need help starting and terminating a server. This CAS server is identified by specifying the host on which it runs and the port on which it listens for communications. To simplify your interactions with this CAS server, the host information and port information for the server are stored as SAS option values that are retrieved automatically whenever this CAS server needs to be accessed. You can examine the host and port values for the server at your site by using the following statements:

```
proc options option=(CASHOST CASPORT);
run;
```

In addition to starting a CAS server, your system administrator might also have created a CAS session and a CAS engine libref for your use. You can define your own sessions and CAS engine librefs that connect to the CAS server as shown in the following statements:

```
cas mysess;
libname mylib cas sessref=mysess;
```

The CAS statement creates the CAS session named mysess, and the LIBNAME statement creates the mylib CAS engine libref that you use to connect to this session. It is not necessary to explicitly name the CASHOST and CASPORT of the CAS server in the CAS statement, because these values are retrieved from the corresponding SAS option values.

If you have created the mysess session, you can terminate it by using the TERMINATE option in the CAS statement as follows:

```
cas mysess terminate;
```

For more information about the CAS and LIBNAME statements, see the section "Introduction to Shared Concepts" on page 6 in Chapter 2, "Shared Concepts."

Getting Started: QTRSELECT Procedure

NOTE: Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections "Using CAS Sessions and CAS Engine Librefs" on page 7 and "Loading a SAS Data Set onto a CAS Server" on page 7 in Chapter 2, "Shared Concepts."

(View the complete code for this example at https://github.com/sascommunities/doc-supplement-casstat/tree/main/Examples/qtrselgs.sas.)

The following example is modeled on the example in the section "Getting Started: QUANTSELECT Procedure" in the *SAS/STAT User's Guide*. The Sashelp.baseball data set contains salary and performance information for Major League Baseball (MLB) players, excluding pitchers, who played in at least one game in both the 1986 and 1987 seasons. The salaries (Time Inc. 1987) are for the 1987 season, and the performance measures are for the 1986 season (Reichler 1987).

The following statements display the variables in the data set. Figure 28.1 shows the results.

```
proc contents varnum data=sashelp.baseball;
   ods select position;
run;
```

Figure 28.1 Sashelp.Baseball Data Set

The CONTENTS Procedure

Variables in Creation Order

#	Variable	Type	Len	Label
1	Name	Char	18	Player's Name
2	Team	Char	14	Team at the End of 1986
3	nAtBat	Num	8	Times at Bat in 1986
4	nHits	Num	8	Hits in 1986
5	nHome	Num	8	Home Runs in 1986
6	nRuns	Num	8	Runs in 1986
7	nRBI	Num	8	RBIs in 1986
8	nBB	Num	8	Walks in 1986
9	YrMajor	Num	8	Years in the Major Leagues
10	CrAtBat	Num	8	Career Times at Bat
11	CrHits	Num	8	Career Hits
12	CrHome	Num	8	Career Home Runs
13	CrRuns	Num	8	Career Runs
14	CrRbi	Num	8	Career RBIs
15	CrBB	Num	8	Career Walks
16	League	Char	8	League at the End of 1986
17	Division	Char	8	Division at the End of 1986
18	Position	Char	8	Position(s) in 1986
19	nOuts	Num	8	Put Outs in 1986
20	nAssts	Num	8	Assists in 1986
21	nError	Num	8	Errors in 1986
22	Salary	Num	8	1987 Salary in $ Thousands
23	Div	Char	16	League and Division
24	logSalary	Num	8	Log Salary

You can load the Sashelp.Baseball data set into your CAS session by specifying your CAS engine libref in the first statement in the following DATA step:

```
data mylib.baseball;
   set sashelp.baseball;
run;
```

These statements assume that your CAS engine libref is named mylib, as in the section "Using CAS Sessions and CAS Engine Librefs" on page 1422, but you can substitute any appropriately defined CAS engine libref.

Suppose you want to investigate how the MLB players' salaries for the 1987 season depend on performance measures for the players' previous season and MLB career. You might worry that some players who are outliers could dominate your least squares analysis. To address this concern, you can use the following statements to obtain a median regression model, which is equivalent to the 50th conditional percentile or the quantile regression model at quantile level 0.5:

```
proc qtrselect data=mylib.baseball;
   class league division;
   model Salary = nAtBat nHits nHome nRuns nRBI nBB
                  yrMajor crAtBat crHits crHome crRuns crRbi
                  crBB league division nOuts nAssts nError;
```

```
run;
```

If you do not use the SELECTION statement, the QTRSELECT procedure fits the full model that is specified by the MODEL statement without any effect selection.

Figure 28.2 Number of Observations, Class Level Information, and Dimensions Tables

The QTRSELECT Procedure

Number of Observations Read	322
Number of Observations Used	263

Class Level Information

Class	Levels	Values
League	2	American National
Division	2	East West

Dimensions

Description	Value
Number of Effects	19
Number of Parameters	21

Figure 28.2 displays the "Number of Observations," "Class Level Information," and "Dimensions" tables.

The "Number of Observations" table shows that, of the 322 observations, PROC QTRSELECT uses only 263 observations for model fitting and ignores 59 incomplete observations.

The "Class Level Information" table shows level information for two CLASS effects that the CLASS statement identifies: League and Division. League has two levels: American League and National League. Division also has two levels: East Division and West Division.

The "Dimensions" table shows that the MODEL statement identifies 19 effects for model fitting besides the intercept effect. Because the 19 effects include two CLASS effects and each level of a CLASS effect corresponds to a parameter, the 19 effects contain a total of 21 parameters.

Figure 28.3 Fit Statistics

The QTRSELECT Procedure

Quantile Level = 0.5

Objective Function	25977
R1	0.40584
Adj R1	0.36200
AIC	2453.81587
AICC	2456.94344
SBC	2521.68680
ACL	98.77118

Figure 28.3 displays the "Fit Statistics" table, which shows the values of model fitting criteria for the fitted median model. For more information about model fitting criteria for quantile regression, see the section "Details: QTRSELECT Procedure" on page 1445.

Figure 28.4 Parameter Estimates

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	-67.75322	39.95908	-1.70	0.0912
nAtBat	1	-1.57112	0.44700	-3.51	0.0005
nHits	1	8.82192	1.94990	4.52	<.0001
nHome	1	-5.91757	4.91015	-1.21	0.2293
nRuns	1	-5.17076	2.14914	-2.41	0.0169
nRBI	1	0.77547	2.15469	0.36	0.7192
nBB	1	5.28866	1.67603	3.16	0.0018
YrMajor	1	6.61877	6.61798	1.00	0.3182
CrAtBat	1	-0.04463	0.15485	-0.29	0.7734
CrHits	1	0.07896	0.73594	0.11	0.9146
CrHome	1	3.78231	1.90065	1.99	0.0477
CrRuns	1	1.23105	0.77137	1.60	0.1118
CrRbi	1	-0.70695	0.76888	-0.92	0.3588
CrBB	1	-0.68911	0.41382	-1.67	0.0971
League American	1	-34.39136	24.37175	-1.41	0.1595
League National	0	0	.	.	.
Division East	1	60.30856	27.28730	2.21	0.0280
Division West	0	0	.	.	.
nOuts	1	0.23273	0.12110	1.92	0.0558
nAssts	1	0.09824	0.18888	0.52	0.6035
nError	1	-0.81574	3.51436	-0.23	0.8166

Figure 28.4 displays the "Parameter Estimates" table, which shows the parameter estimates of the fitted median model. You can see that, of the 19 effective parameters whose degrees of freedom are not zero, the fitted model contains 13 insignificant parameters whose 95% confidence intervals cover zeros. Because more than half of the 19 effective parameters are insignificant, you might worry that the model is overfitted.

It is well known that both overfitting and underfitting harm the prediction performance of a model. You can prevent overfitting and underfitting by using a good effect-selection technique. The following statements apply the forward selection method and the SL (significance level) criterion to choose a parsimonious model for the mylib.baseball data table:

```
proc qtrselect data=mylib.baseball;
   class league division;
   model Salary = nAtBat nHits nHome nRuns nRBI nBB
                  yrMajor crAtBat crHits crHome crRuns crRbi
                  crBB league division nOuts nAssts nError
         / clb;
   selection method=forward(select=sl sle=0.1);
run;
```

The CLB option in the MODEL statement requests 95% confidence limits for the parameter estimates. The SLE=0.1 option in the SELECTION statement specifies the significance level for entry. A candidate effect can enter the model at a certain selection step only if the following conditions are met:

- Its *p*-value is the smallest among all the valid candidate effects.

- Its p-value is smaller than 0.1 (the significance level for entry).

For more information about using significance levels in effect selection, see the section "Statistical Tests for Significance Level" on page 1449.

Figure 28.5 Selection Information

The QTRSELECT Procedure

Selection Information	
Selection Method	Forward
Select Criterion	Significance Level
Stop Criterion	Significance Level
Effect Hierarchy Enforced	None
Entry Significance Level (SLE)	0.1
Stop Horizon	1

Figure 28.5 displays the "Selection Information" table. The "Selection Information" provides details about the method and criteria used to perform the model selection. The requested selection method is the forward selection method where the decisions about what effects to add at any step and when to terminate the selection are both based on the significance level criterion.

Figure 28.6 Selection Summary

The QTRSELECT Procedure

Quantile Level = 0.5
Selection Details

Selection Summary			
Step	**Effect Entered**	**Number Effects In**	**p Value**
0	**Intercept**	1	.
1	**CrHome**	2	<.0001
2	**nHits**	3	<.0001
3	**CrHits**	4	<.0001
4	**nOuts**	5	0.0185
5	**nAtBat**	6	0.0182
6	**Division**	7	0.0118
7	**nBB**	8	0.0647
8	**nRuns**	9	0.0558

Figure 28.6 displays the "Selection Summary" table. Each row in the "Selection Summary" table shows the effect that enters the model at the corresponding step of the effect selection process together with its p-value for adding the effect into the model at that step.

Figure 28.7 Stopping and Selection Reasons

Selection stopped because no candidate for entry is significant at the 0.1 level.

The model at step 8 is selected.

Figure 28.7 *continued*

Selected Effects:	Intercept nAtBat nHits nRuns nBB CrHits CrHome Division nOuts

Figure 28.7 displays the "Stop Reason," "Selection Reason," and "Selected Effects" tables. The "Stop Reason" and "Selection Reason" tables indicate that effect selection stopped because no candidate for entry was significant at the 0.1 level after step 8. The "Selected Effects" table lists the effects that are included in the selected model.

Figure 28.8 Details of the Selected Model

The QTRSELECT Procedure

Quantile Level = 0.5
Selected Model

Objective Function	26568
R1	0.39232
Adj R1	0.37318
AIC	2445.64547
AICC	2446.35693
SBC	2477.79485
ACL	101.01768

Parameter Estimates

Parameter	DF	Estimate	Standard Error	95% Confidence Limits		t Value	Pr > |t|
Intercept	1	-130.65536	33.04546	-195.73336	-65.57735	-3.95	<.0001
nAtBat	1	-1.20522	0.41690	-2.02624	-0.38419	-2.89	0.0042
nHits	1	7.76667	1.81045	4.20127	11.33207	4.29	<.0001
nRuns	1	-3.92180	1.87567	-7.61566	-0.22795	-2.09	0.0375
nBB	1	3.92049	1.04341	1.86565	5.97532	3.76	0.0002
CrHits	1	0.17697	0.06856	0.04195	0.31198	2.58	0.0104
CrHome	1	1.66939	0.73083	0.23012	3.10866	2.28	0.0232
Division East	1	65.14327	24.48038	16.93289	113.35364	2.66	0.0083
Division West	0	0	.	.	.	.	.
nOuts	1	0.23719	0.11403	0.01261	0.46176	2.08	0.0385

The "Fit Statistics" and "Parameter Estimates" tables in Figure 28.8 give details of the final selected model. You can see that all nine effective parameters (excluding Division West) are significant at the 5% significance level, corresponding to the 95% confidence limits.

Like the sample median, a median regression model is robust to extreme observations, because it depends only on a small middle subset of all the observations in the data table. However, it is less representative of the entire conditional distribution of the response variable. You might want to further investigate the mylib.baseball data table at other quantile levels. The following statements select quantile regression models at the quantile levels 0.1 and 0.9, which correspond to the 10% and 90% conditional percentiles of the players' salaries:

```
proc qtrselect data=mylib.baseball alpha=0.1;
   class league division;
   model Salary = nAtBat nHits nHome nRuns nRBI nBB
                  yrMajor crAtBat crHits crHome crRuns crRbi
                  crBB league division nOuts nAssts nError
         / quantile=0.1 0.9 stb clb;
   selection method=backward(select=sl sls=0.1);
run;
```

The ALPHA=0.1 option in the PROC statement sets the significance level to 0.1. Combined with the CLB option in the MODEL statement, the ALPHA=0.1 option requests 90% confidence limits for parameter estimates. The QUANTILE= option in the MODEL statement specifies two quantile levels, 0.1 and 0.9, for fitting quantile regression models. The METHOD=BACKWARD option in the SELECTION statement specifies the backward elimination method for effect selection.

Figure 28.9 Parameter Estimates at Quantile Level 0.1

The QTRSELECT Procedure

Quantile Level = 0.1
Selection Details

Selected Effects: Intercept nAtBat nHits nBB CrRuns CrBB Division nAssts

Parameter Estimates

Parameter	DF	Estimate	Standardized Estimate	Standard Error	90% Confidence Limits		t Value	Pr > |t|
Intercept	1	4.75224	0	18.94983	-26.53111	36.03558	0.25	0.8022
nAtBat	1	-0.73670	-0.23293	0.17958	-1.03316	-0.44024	-4.10	<.0001
nHits	1	2.69396	0.26272	0.63510	1.64551	3.74241	4.24	<.0001
nBB	1	1.81807	0.08668	0.40595	1.14791	2.48823	4.48	<.0001
CrRuns	1	0.65476	0.48600	0.09560	0.49694	0.81259	6.85	<.0001
CrBB	1	-0.44622	-0.26842	0.16254	-0.71454	-0.17790	-2.75	0.0065
Division East	1	28.35406	0.03148	10.68922	10.70774	46.00038	2.65	0.0085
Division West	0	0	0	.	.	.	.	.
nAssts	1	0.14958	0.04811	0.05897	0.05223	0.24694	2.54	0.0118

Figure 28.9 displays the "Selected Effects" and "Parameter Estimates" tables at quantile level 0.1.

Figure 28.10 Parameter Estimates at Quantile Level 0.9

Selected Effects: Intercept nHits nBB CrAtBat CrHits CrHome CrRbi League Division nOuts

Figure 28.10 *continued*

Parameter Estimates

Parameter	DF	Estimate	Standardized Estimate	Standard Error	90% Confidence Limits		t Value	Pr > |t|
Intercept	1	20.39804	0	58.17164	-75.63745	116.43353	0.35	0.7261
nHits	1	2.30897	0.22517	0.55640	1.39042	3.22752	4.15	<.0001
nBB	1	3.09799	0.14770	1.44414	0.71386	5.48213	2.15	0.0329
CrAtBat	1	-0.44914	-2.28028	0.14651	-0.69101	-0.20727	-3.07	0.0024
CrHits	1	2.48064	3.56823	0.51725	1.62672	3.33457	4.80	<.0001
CrHome	1	6.29896	1.20138	1.37134	4.03502	8.56291	4.59	<.0001
CrRbi	1	-2.12293	-1.54553	0.76546	-3.38662	-0.85923	-2.77	0.0060
League American	1	-103.28955	-0.11451	33.68480	-158.89975	-47.67935	-3.07	0.0024
League National	0	0	0	.	.	.	.	.
Division East	1	107.46694	0.11932	50.82797	23.55512	191.37876	2.11	0.0355
Division West	0	0	0	.	.	.	.	.
nOuts	1	0.39766	0.24676	0.12820	0.18601	0.60931	3.10	0.0021

Figure 28.9 displays the "Selected Effects" and "Parameter Estimates" tables at quantile level 0.9.

You might want to compute the 90th percentile predictions for players' salaries and find out which players were overpaid based on the quantile regression model at quantile level 0.9. The following statements repeat the backward elimination method at quantile level 0.9, compute and sort the overpaid players' salaries, and output the observations for the top 10 overpaid players in the mylib.baseball data table:

```
proc qtrselect data=mylib.baseball alpha=0.1;
   class league division;
   model Salary = nAtBat nHits nHome nRuns nRBI nBB
                  yrMajor crAtBat crHits crHome crRuns crRbi
                  crBB league division nOuts nAssts nError
         / quantile=0.9 clb;
   selection method=backward(select=sl sls=0.1);
   output out=mylib.BaseballOverpaid copyvar=(Name Salary) r=Overpaid
          p=PredictedSalary lclm uclm;
run;

data BaseballOverpaid;
   set mylib.BaseballOverpaid;
run;

proc sort data=BaseballOverpaid;
   by descending Overpaid;
run;

proc print data=BaseballOverpaid(obs=10);
   var Name Salary Overpaid PredictedSalary lclm uclm;
run;
```

The LCLM and UCLM options, respectively, request lower and upper bounds of $100(1-\alpha)\%$ confidence intervals for the expected conditional quantile predictions of players' salaries at quantile level 0.9.

Figure 28.11 Top 10 Overpaid Baseball Players at Quantile Level 0.9

Obs	Name	Salary	Overpaid	PredictedSalary	LCLM	UCLM
1	Smith, Ozzie	1940.0	1084.54	855.46	611.09	1099.83
2	Wiggins, Alan	700.0	220.74	479.26	345.93	612.60
3	Murray, Eddie	2460.0	213.10	2246.90	1982.06	2511.74
4	Strawberry, Darryl	1220.0	187.64	1032.36	914.99	1149.72
5	Gibson, Kirk	1300.0	177.24	1122.76	1011.97	1233.56
6	Trevino, Alex	512.5	149.70	362.80	273.17	452.43
7	Ramirez, Rafael	875.0	141.09	733.91	599.52	868.31
8	Romero, Ed	375.0	128.71	246.29	144.88	347.70
9	Mattingly, Don	1975.0	124.82	1850.18	1557.81	2142.55
10	Puhl, Terry	900.0	104.34	795.66	625.51	965.81

Output 28.11 shows the information about the top 10 overpaid players according to the final selected quantile regression model at quantile level 0.9. Ozzie Smith is in first place. This might be because, although Smith was known for his defensive brilliance, the model weights offensive performance measures much more than defensive performance measures.

Syntax: QTRSELECT Procedure

The following statements are available in the QTRSELECT procedure:

PROC QTRSELECT < *options* > ;
  BY *variables* ;
  CLASS *variable* < **(***options***)** >. . . < *variable* < **(***options***)** > > < / *global-options* > ;
  CODE < *options* > ;
  DISPLAY < *table-list* > < / *options* > ;
  DISPLAYOUT *table-spec-list* < / *options* > ;
  EFFECT *name***=***effect-type***(***variables* < / *options* >**)** ;
  MODEL *dependent* **=** < *effects* > < / *model-options* > ;
  OUTPUT OUT=*libref.data-table* < *keyword* < **=***name* > >. . . < *keyword* < **=***name* > > < / *options* > ;
  PARTITION *partition-options* ;
  SELECTION < **METHOD=***method* < **(***method-options***)** > >< *options* > ;
  STORE < **OUT=** >*libref.data-table* ;
  WEIGHT *variable* ;

The PROC QTRSELECT statement and a single MODEL statement are required. All other statements are optional. The CLASS statement can appear multiple times. If a CLASS statement is specified, it must precede the MODEL statement.

The rest of this section provides detailed syntax information about each of the preceding statements, beginning with the PROC QTRSELECT statement. The remaining statements are described in alphabetical order.

PROC QTRSELECT Statement

PROC QTRSELECT < *options* > **;**

The PROC QTRSELECT statement invokes the procedure. Table 28.1 summarizes the options in the PROC QTRSELECT statement by function.

Table 28.1 PROC QTRSELECT Statement Options

Option	Description
Basic Options	
DATA=	Specifies the input data table
Other Options	
ALPHA=	Sets the significance level to be used for the construction of confidence intervals
COV=SPARSITY	Specifies the sparsity-function method for estimating the covariance matrix of the parameter estimates
NOCLPRINT	Limits or suppresses the display of class levels

You can specify the following *options*:

ALPHA=*number*
sets the significance level to be used for the construction of confidence intervals. The value must be between 0 and 1; the default value of 0.05 results in 95% intervals. This option affects the keywords LCLM, UCLM, and STDP in the OUTPUT statement and the CLB option in the MODEL statement.

COV=SPARSITY< (BF | HS) >
specifies the sparsity-function bandwidth method for estimating the covariance matrix of the parameter estimates. You can specify the following suboptions:

BF uses the Bofinger bandwidth method.

HS uses the Hall-Sheather bandwidth method.

By default, COV=SPARSITY(HS). For more information, see the section "Details: QTRSELECT Procedure" on page 1445.

DATA=*libref.data-table*
names the input data table for PROC QTRSELECT to use. The default is the most recently created data table. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to the data, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 1422.

data-table specifies the name of the input data table.

NOCLPRINT<**=***number*>
suppresses the display of the "Class Level Information" table if you do not specify *number*. If you specify *number*, the values of the classification variables are displayed for only those variables whose number of levels is less than *number*. Specifying *number* helps reduce the size of the "Class Level Information" table if some classification variables have a large number of levels.

BY Statement

BY *variables* **;**

You can specify a BY statement in PROC QTRSELECT to obtain separate analyses of observations in groups that are defined by the values of the BY variables. If you specify more than one BY statement, only the last one specified is used. For more information, see the discussion of BY-group processing in the "Grouping Data" section of *SAS Programmers Guide: Essentials*.

CLASS Statement

CLASS *variable* <**(***options***)**>. . . <*variable* <**(***options***)**>> </ *global-options*> **;**

The CLASS statement names the classification variables to be used as explanatory variables in the analysis. You can list the response variable for binary models in the CLASS statement, but this is not required. Table 28.2 summarizes the values that you can use for either an *option* or a *global-option*. The options are fully documented in the section "CLASS Statement" on page 9 in Chapter 2, "Shared Concepts."

Table 28.2 CLASS Statement Options

Option	Description
DESCENDING	Reverses the sort order
MISSING	Treats missing values as valid levels
ORDER=	Specifies the sort order for the levels
PARAM=	Specifies the parameterization of the variable
REF=	Specifies the reference level of the variable
SPLIT	Splits levels of CLASS variables into independent effects

CODE Statement

CODE < *options* > **;**

The CODE statement writes SAS DATA step code for computing predicted values of the fitted model to a file or to a table. To score new data, you can then include the file in a DATA step, or you can specify the table in the **runCodeTable** action in the **dataStep** action set (for more information, see *SAS Viya: System Programming Guide*).

Table 28.3 summarizes the *options* available in the CODE statement.

Table 28.3 CODE Statement Options

Option	Description
COMMENT	Adds comments to the generated code
FILE=	Names the file in which to save the generated code
FORMATWIDTH=	Specifies the numeric format width for the regression coefficients
INDENTSIZE=	Specifies the number of spaces to indent the generated code
LABELID=	Specifies a number used to construct names and labels
LINESIZE=	Specifies the line size for the generated code
NOTRIM	Compares formatted values, including blank padding
OUT=	Names an output table in which to save the generated code

For more information about the syntax of the CODE statement, see the section "CODE Statement" on page 13 in Chapter 2, "Shared Concepts."

The predicted values in the SAS DATA step code are computed based on the parameter estimates for the selected models. If you specify multiple quantile levels by using the QUANTILES option in the MODEL statement, then one variable of the predicted values is generated for each specified quantile level.

DISPLAY Statement

DISPLAY < *table-list* > < / *options* > **;**

The DISPLAY statement enables you to specify a list of display tables to display or exclude. This statement is similar to the ODS SELECT, ODS EXCLUDE, and ODS TRACE statements. However, the DISPLAY statement can improve performance when a large number of tables could be generated (such as in BY-group processing). The procedure processes the DISPLAY statement on a CAS server and thus sends only a subset of ODS tables to the SAS client. Because ODS statements are processed on a SAS client, first all the generated display tables are sent to the client, and then the client creates a subset.

If you use both DISPLAY and ODS statements together, the DISPLAY statement takes precedence over the ODS statements. Note that the ODS EXCLUDE statement processes tables that are sent to the client after they have been filtered by the DISPLAY statement. In some cases, it might appear that the ODS EXCLUDE statement is taking precedence because it can further filter the tables. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

You can specify the *table-list* as a list of table names, paths, partial pathnames, and regular expressions.

The table names that you can specify are listed in the section "ODS Table Names" on page 1459. A path is a table name that is prefixed with dot-separated grouping information. For example, a SelectionSummary table that a procedure produces during a selection routine might have the path *Bygroup1.Summary.SelectionSummary*. A partial pathname does not include all groups; for example, *SelectionSummary* and *Summary.SelectionSummary* are partial pathnames for *Bygroup1.Summary.SelectionSummary*.

When you specify a table name or partial pathname, all display tables whose paths end in the specified name are selected for display or exclusion. For example, both *SelectionSummary* and *Summary.SelectionSummary* select *Bygroup1.Summary.SelectionSummary*.

A regular expression is enclosed in forward slashes (/). For example, specifying "/tions/" selects all pathnames that contain the substring "tions"; in particular, the *Bygroup1.Summary.SelectionSummary* table is selected. Specifying "!/tions/" selects all pathnames that do not contain the substring "tions"; in particular, the *Bygroup1.Summary.SelectionSummary* table is not selected.

You can specify the following *options* after a slash (/):

CASESENSITIVE
performs a case-sensitive comparison of table names in the *table-list* to display table names when tables are subsetted for display. To preserve case, you must enclose table names in the *table-list* in quotation marks.

EXCLUDE
displays all display tables except those that you specify in the *table-list*.

EXCLUDEALL
suppresses display of all tables. This option takes precedence over the other options.

TRACE
displays the display table names, labels, and paths.

DISPLAYOUT Statement

DISPLAYOUT *table-spec-list* < / *options* > **;**

The DISPLAYOUT statement enables you to create output tables from your displayed output. This statement is similar to the ODS OUTPUT statement. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

The *table-spec-list* specifies a list of output tables to create. Each entry in the list has either a *key=value* format or a *key* format:

key=value specifies *key* as the ODS table name, path, or partial pathname, and specifies *value* as the output table name.

key specifies *key* as the ODS table name and also as the output table name.

The ODS table names that you can specify are listed in the section "ODS Table Names" on page 1459. You cannot specify the ODS table named OutputCasTables in the *table-spec-list*.

Table names and partial pathnames are discussed under the DISPLAY statement. The DISPLAYOUT statement does not support regular expressions.

You can specify the following *options* after a slash (/):

INCLUDEALL
creates output tables for all display tables. The name of the created output table is the same as the name of the corresponding display table. If you specify this option, the *table-spec-list* specification is ignored.

NOREPLACE
does not replace any existing output table of the same name.

REPEATED
replicates all output tables on all nodes.

EFFECT Statement

EFFECT *name*=*effect-type* **(***variables* < / *options* >**) ;**

The EFFECT statement enables you to construct special collections of columns for design matrices. These collections are referred to as *constructed effects* to distinguish them from the usual model effects that are formed from continuous or classification variables, as discussed in the section "GLM Parameterization of Classification Variables and Effects" on page 67 in Chapter 2, "Shared Concepts."

You can specify the following *effect-types*:

COLLECTION specifies a collection effect that defines one or more variables as a single effect that has multiple degrees of freedom. The variables in a collection are considered as a unit for purposes of estimation and inference.

MULTIMEMBER | MM specifies a multimember classification effect whose levels are determined by one or more variables that appear in a CLASS statement.

POLYNOMIAL | POLY specifies a multivariate polynomial effect in the specified numeric variables.

SPLINE specifies a regression spline effect whose columns are univariate spline expansions of one or more variables. A spline expansion replaces the original variable with an expanded or larger set of new variables.

Table 28.4 summarizes the *options* available in the EFFECT statement.

Table 28.4 EFFECT Statement Options

Option	Description
Collection Effects Options	
DETAILS	Displays the constituents of the collection effect

Table 28.4 *continued*

Option	Description
Multimember Effects Options	
DETAILS	Displays the levels of the multimember effect
NOEFFECT	Specifies that observations whose levels are all missing for the multimember variables should have 0 values in the corresponding design matrix columns
STDIZE	Standardizes the design matrix entries so that each observation has a sum of 1
WEIGHT=	Specifies the weight variable for the contributions of each classification effect
Polynomial Effects Options	
DEGREE=	Specifies the degree of the polynomial
DETAILS	Displays details of the specified polynomial
MDEGREE=	Specifies the maximum degree of any variable in a term of the polynomial
NOSEPARATE	Treats the polynomial as a single effect with multiple degrees of freedom
STANDARDIZE=	Specifies centering and scaling suboptions for the variables that define the polynomial
Spline Effects Options	
BASIS=	Specifies the type of basis (B-spline basis or truncated power function basis) for the spline effect
DATABOUNDARY	Uses the extremes of the data as boundary knots for a B-spline basis
DEGREE=	Specifies the degree of the spline effect
DETAILS	Displays the knots and locations for each spline basis function
KNOTMAX=	Requests equally spaced right-side boundary knots starting at the variables' maximum and ending at the KNOTMAX= value
KNOTMETHOD=	Specifies how to construct the knots for the spline effect
KNOTMIN=	Requests equally spaced left-side boundary knots starting at the KNOTMIN= value and ending at the variables' minimum value
NATURALCUBIC	Specifies a natural cubic spline basis for the spline effect
SEPARATE	Treats the spline basis for each variable as a separate effect when multiple variables are specified
SPLIT	Treats each design matrix column as a separate effect for selection methods

For more information about the syntax of these *effect-types* and how columns of constructed effects are computed, see the section "EFFECT Statement" on page 18 in Chapter 2, "Shared Concepts."

MODEL Statement

MODEL *dependent*=< *effects* > / < *model-options* > **;**

The MODEL statement names the dependent variable and the explanatory effects, including covariates, main effects, interactions, and nested effects. If you omit the explanatory effects, the procedure fits an intercept-only model.

After the keyword MODEL, the dependent (response) variable is specified, followed by an equal sign. The explanatory effects follow the equal sign.

For information about constructing the model effects, see the section "Specification and Parameterization of Model Effects" on page 64 in Chapter 2, "Shared Concepts."

The *model-options* control other aspects of model formation and inference. Table 28.5 summarizes these options.

Table 28.5 MODEL Statement Options

Option	Description
Model Options	
CLB	Requests confidence limits
INCLUDE=	Includes effects in all models for model selection
INFORMATIVE	Models missing values by using extra indicator variables
NOINT	Suppresses the intercept
QUANTILES	Specifies quantile levels for the quantile regression
START=	Includes effects in the initial model for model selection
STB	Displays standardized estimates

Model Options

You can specify the following *options* in the MODEL statement after a slash (/):

CLB

requests the $100(1-\alpha)\%$ upper and lower confidence limits for the parameter estimates. By default, the 95% limits are computed; you can use the ALPHA= option in the PROC QTRSELECT statement to change the α level.

INCLUDE=*n*

INCLUDE=*single-effect*

INCLUDE=(*effects***)**

forces effects to be included in all models. If you specify INCLUDE=*n*, then the first *n* effects listed in the MODEL statement are included in all models. If you specify INCLUDE=*single-effect* or INCLUDE=(*single-effect*), then the specified effects are forced into all models.

INFORMATIVE

models missing values by using extra model effects. These effects consist of dummy variables that take the value 1 when the value of a continuous model variable involved in the effect is missing, and take the value 0 otherwise. The missing value in the original model effect is replaced by the average value of the effect for the nonmissing values. For continuous-by-class effects, such as A*x, where A is a classification variable and x is a continuous variable, informative missingness creates multiple dummy columns and substitutes the effect mean of x that corresponds to the respective level of A. Missing values for classification variables are treated as valid levels. For more information about informative missingness, see the section "Informative Missingness" on page 99 in Chapter 2, "Shared Concepts."

NOINT

suppresses the intercept term that is otherwise included in the model.

QUANTILES< (*quantile-level-options*) **>< =***number-list* **>**

QUANTILE< (*quantile-level-options*) **>< =***number-list* **>**

Q< (*quantile-level-options*) **>< =***number-list* **>**

specifies the quantile levels for the quantile regression. You can request any values of quantile levels in $(0, 1)$ by specifying a *number-list*. By default, the QTRSELECT procedure uses QUANTILES=0.5, which corresponds to median regression.

You can also specify the following *quantile-level-options*:

NTAU=*n*

NQ=*n*

specifies the following *n* quantile levels for the quantile regression:

$$\left\{ \frac{1}{n+1}, \ldots, \frac{n}{n+1} \right\}$$

If you specify both a QUANTILES=*number-list* and NTAU=*n*, the QTRSELECT procedure uses all the specified quantile levels.

SORT

sorts all the specified quantile levels in ascending order. This option affects the order of the prediction variables in the SAS DATA step code that is created by the CODE statement, and the order of the appropriate keyword variables in the SAS data set that is created by the OUTPUT statement.

START=*n*

START=*single-effect*

START=(*effects***)**

begins the selection process in the FORWARD and STEPWISE selection methods from the initial model that you designate. If you specify START=*n*, then the starting model consists of the first *n* effects listed in the MODEL statement. If you specify START=*single-effect* or START=(*single-effect*), then the starting model consists of these specified effects.

STB

produces standardized regression coefficients. A standardized regression coefficient is computed by dividing a parameter estimate by the ratio of the sample standard deviation of the dependent variable to the sample standard deviation of the regressor. If you use the INCLUDE= option to force some effects to be in the model, then the QTRSELECT procedure computes the sample standard deviation against all the effects that are forced in, as follows. Let $\mathbf{X}_1$ denote the design submatrix of p_1 regressors that consists of all the effects that are forced in, and let $\mathbf{z}$ denote the dependent variable or any regressor. Then the sample standard deviation of $\mathbf{z}$ is computed as

$$s_z = \sqrt{\frac{\mathbf{z}'\left[\mathbf{I} - \mathbf{X}_1(\mathbf{X}_1{}'\mathbf{X}_1)^{-1}\mathbf{X}_1{}'\right]\mathbf{z}}{n - p_1}}$$

OUTPUT Statement

OUTPUT OUT=*libref.data-table*
< **COPYVARS=***(variables)* >
< *keyword* < *=name* > >. . . < *keyword* < *=name* > > **;**

The OUTPUT statement creates a data table that contains observationwise statistics, which are computed after the model is fitted. The variables in the input data table are *not* included in the output data table, in order to avoid data duplication for large data tables; however, variables that you specify in the COPYVARS= option are included.

The output statistics are computed based on the parameter estimates for the selected model. If you specify multiple quantile levels by using the QUANTILES option in the MODEL statement, then for each appropriate keyword that is specified in the OUTPUT statement, one variable is generated for each specified quantile level. For observations in which only the response variable is missing, predicted values are computed even though these observations do not affect the model fit. This enables, for example, predicted values to be computed for new observations.

You must specify the following option:

OUT=*libref.data-table*
names the output data table for PROC QTRSELECT to use. You must specify this option before any other options. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to where the data table is to be stored, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 1422.

data-table specifies the name of the output data table.

You can also specify the following syntax elements:

COPYVAR=*variable*

COPYVARS=(*variables***)**

transfers one or more *variables* from the input data table to the output data table.

keyword **<=***name***>**

specifies the statistics to include in the output data table and optionally names the new variables that contain the statistics. Specify a keyword for each desired statistic (see the following list of keywords), followed optionally by an equal sign and a variable to contain the statistic.

If you specify *keyword=name*, the new variable that contains the requested statistic has the specified name. If you omit the optional *=name* after a *keyword*, then a default name is used.

You can specify the following values for *keyword* to request statistics that are available with all selection methods:

PREDICTED<=*name***>**

PRED<=*name***>**

P<=*name***>**

requests predicted values for the response variable. The default *name* is Pred.

RESIDUAL<=*name***>**

RESID<=*name***>**

R<=*name***>**

requests the residual, calculated as ACTUAL – PREDICTED. The default *name* is Residual.

ROLE<=*name***>**

requests a numeric variable that indicates the role played by each observation in fitting the model. The default *name* is _ROLE_. For each observation, the interpretation of this variable is shown in Table 28.6.

Table 28.6 Role Interpretation

Value	Observation Role
0	Not used
1	Training
2	Validation
3	Testing

If you do not partition the input data by using a PARTITION statement, then the role variable value is 1 for observations used in fitting the model, and 0 for observations that have at least one missing or invalid value for the response, regressor, frequency, or weight variables.

If you specify multiple quantile levels by using the QUANTILES option in the MODEL statement, then for each appropriate keyword that is specified in the OUTPUT statement, one variable is generated for each specified quantile level. These variables appear in the same order as the specified quantile levels. For example, the following statements generate the mylib.out data table, which contains the two predicted quantile variables p1 and p2:

```
proc qtrselect data=mylib.one;
   model y = x1-x4 /quantiles=0.5 0.3;
   output out=mylib.out pred=p;
run;
```

The variable p1 is for quantile level 0.5, and the variable p2 is for quantile level 0.3.

By using the SORT suboption in the QUANTILES option, the following statements generate the mylib.out data table in sorted order:

```
proc qtrselect data=mylib.one;
   model y = x1-x4 /quantiles(sort)=0.5 0.3;
   output out=mylib.out pred=p;
run;
```

The variable p1 is for quantile level 0.3, and the variable p2 is for quantile level 0.5, because the sorted quantile levels are (0.3 0.5).

In addition to the preceding statistics, you can also use the *keywords* listed in Table 28.7 in the OUTPUT statement to obtain additional statistics. For computational formulas, see the section "Diagnostic Statistics" on page 1450. All the statistics available in the OUTPUT statement are conditional on the selected model and do not take into account the variability introduced when you do model selection.

Table 28.7 Keywords for OUTPUT Statement

Keyword	Description
LCLM	Lower bound of a $100(1-\alpha)\%$ confidence interval for the quantile prediction variable
STDP	Standard error of the quantile prediction variable
UCLM	Upper bound of a $100(1-\alpha)\%$ confidence interval for the quantile prediction variable

PARTITION Statement

PARTITION *partition-option* **;**

The PARTITION statement specifies how observations in the input data set are logically partitioned into disjoint subsets for model training, validation, and testing. For more information, see the section "Using Validation and Test Data" on page 101 in Chapter 2, "Shared Concepts." Either you can designate a variable in the input data table and a set of formatted values of that variable to determine the role of each observation, or you can specify proportions to use for randomly assigning observations to each role.

You must specify exactly one of the following *partition-options*:

FRACTION(< TEST=*fraction* **> < VALIDATE=***fraction* **> < SEED=***number* **>)**
randomly assigns specified proportions of the observations in the input data table to the roles. You specify the proportions for testing and validation by using the TEST= and VALIDATE= suboptions. If you specify both the TEST= and VALIDATE= suboptions, then the sum of the specified fractions must be less than 1 and the remaining fraction is the proportion of the observations assigned to the training role. The SEED= option specifies an integer that is used to start the pseudorandom number generator for random partitioning of data for training, testing, and validation. If you do not specify SEED=*number* or if *number* is less than or equal to 0, the seed is generated by reading the time of day from the computer's clock.

ROLE=*variable* **(< TEST='***value***' > < TRAIN='***value***' > < VALIDATE='***value***' >)**

ROLEVAR=*variable* **(< TEST='***value***' > < TRAIN='***value***' > < VALIDATE='***value***' >)**
names the *variable* in the input data table whose values are used to assign roles to each observation. This *variable* cannot also appear as an analysis variable in other statements or options. The TEST=, TRAIN=, and VALIDATE= suboptions specify the formatted values of this variable that are used to assign observation roles. If you do not specify the TRAIN= suboption, then all observations whose role is not determined by the TEST= or VALIDATE= suboption are assigned to the training role.

SELECTION Statement

SELECTION < **METHOD=***method* < **(***method-options***)** > >< *options* > **;**

The SELECTION statement performs model selection by examining whether effects should be added to or removed from the model according to rules that are defined by model selection methods. The statement is fully documented in the section "SELECTION Statement" on page 42 in Chapter 2, "Shared Concepts."

The QTRSELECT procedure supports the following values of the METHOD= option in the SELECTION statement:

BACKWARD specifies the backward elimination method, which starts with all effects in the model and deletes effects.

FORWARD specifies the forward selection method, which starts with no effects in the model and adds effects.

NONE specifies no model selection.

STEPWISE specifies the stepwise regression method, which is similar to the forward selection method except that effects already in the model do not necessarily stay there.

By default, METHOD=STEPWISE.

The DETAILS=ALL and DETAILS=STEPS options produce the "Fit Statistics," and "Parameter Estimates" tables, which provide information about the model that is selected at each step of the selection process.

By default, the QTRSELECT procedure displays the stop horizon steps in the "Selection Summary" table, and uses these steps for choosing the final model of the model selection process. Therefore, it is possible that the final model occurs after the step that has the best stop criterion value. To exclude the stop horizon steps for choosing the final model, you can specify the following option:

HIDESTOPSTEPS

HSS

hides the stop horizon steps in the "Selection Summary" table, and excludes these steps in choosing the final model.

If you specify the PLOTS=CRITERIA or PLOTS=ALL option, then a plot of the fit criteria by the selection step is created for the AIC, AICC, SBC, R1, and adjusted R1 statistics. If you also specify a PARTITION statement, then the same type of plot is created for the ACL criteria for training, validation, and testing roles. If you specify the PLOTS=FITBYROLE option and a PARTITION statement, then a plot of the ACL criterion by the selection step for each role is created.

STORE Statement

STORE < **OUT=** >*libref.data-table* **;**

The STORE statement saves the context and results of the statistical analysis. The resulting analytic store has a binary file format that cannot be modified. An example of analytic store technology is to perform a time-consuming analysis and to store its results by using the STORE statement. Later you can perform specific statistical analysis tasks based on the saved results of the previous analysis, without having to fit the model again. The contents of the analytic store can be processed by the ASTORE procedure and the **`score`** action in the **`aStore`** action set.

libref.data-table is a two-level name, where *libref* refers to the library, and *data-table* specifies the name of the output data table. For more information about this two-level name, see the section "Using CAS Sessions and CAS Engine Librefs" on page 1422.

WEIGHT Statement

WEIGHT *variable* **;**

The *variable* in the WEIGHT statement is used as a weight to perform a weighted analysis of the data. Observations that have nonpositive or missing weights are not included in the analysis. The values of the WEIGHT variable can be nonintegral and are not truncated. When you omit the WEIGHT statement, all observations that are used in the analysis are assigned a weight of 1.

The QTRSELECT procedure uses each valid weight as the scale factor of its relevant observation. Let $\mathbf{X}$ denote the design matrix before weights are used, and let $\mathbf{W}$ denote the diagonal matrix whose diagonal elements are the weights. Then PROC QTRSELECT uses the weighted design matrix $\mathbf{WX}$ for its computation. For example, the weighted version of $\mathbf{X}'\mathbf{X}$ is $\mathbf{X}'\mathbf{WWX}$ but not $\mathbf{X}'\mathbf{WX}$.

Details: QTRSELECT Procedure

Quantile Regression

This section describes the basic concepts and notations for quantile regression and quantile regression model selection.

Let $\{(y_i, \mathbf{x}_i) : i = 1, \ldots, n\}$ denote a data set of observations, where y_i are responses and $\mathbf{x}_i$ are regressors. Koenker and Bassett (1978) define the *regression quantile* at quantile level $\tau \in (0, 1)$ as any solution to the minimization problem

$$\min_{\boldsymbol{\beta} \in \mathbf{R}^p} \sum_{i=1}^{n} \rho_\tau \left(y_i - \mathbf{x}_i{}' \boldsymbol{\beta} \right)$$

where $\rho_\tau(r) = \tau r^+ + (1 - \tau) r^-$ is a check loss function in which $r^+ = \max(r, 0)$ and $r^- = \max(-r, 0)$.

If you specify weights $w_i, i = 1, \ldots, n$, in the WEIGHT statement, then weighted quantile regression is carried out by solving

$$\min_{\boldsymbol{\beta} \in \mathbf{R}^p} \sum_{i=1}^{n} \rho_\tau \left(w_i (y_i - \mathbf{x}_i{}' \boldsymbol{\beta}) \right)$$

The QTRSELECT procedure fits a quantile regression model by using a predictor-corrector interior point algorithm, which was originally designed to solve support vector machine classifiers for large data sets (Gertz and Griffin 2005, 2010).

Linear Model with Independent and Identically Distributed Errors

A linear model with independent and identically distributed (iid) errors assumes that the distribution of the response Y_i conditional on $\mathbf{x}_i$ follows the linear model,

$$Y_i = \mathbf{x}_i{}' \boldsymbol{\beta} + \epsilon_i$$

where ϵ_i for $i = 1, \ldots, n$ are iid in the distribution function F. General linear regression uses this model for deriving its statistical inferences and model selection criteria. Although quantile regression can analyze heterogeneous data without the iid errors assumption, this model is helpful for deriving some quantile regression model selection criteria. For more information about using a linear model with iid errors in deriving the quasi-likelihood information criteria, see the section "Criteria Used in Model Selection" on page 1447.

Linear-in-Parameter Model with Non-iid Settings

The more general form of a linear quantile regression model is

$$Q_Y(\tau|\mathbf{x}) = \mathbf{x}'\boldsymbol{\beta}(\tau)$$

where the iid assumption is not necessary.

Let $s_i(\tau) = 1/f_i(F_i^{-1}(\tau))$ denote the sparsity function of the ith observation. Under some regularity conditions, the asymptotic distribution of the general form of quantile regression estimates is

$$\sqrt{n}(\hat{\boldsymbol{\beta}}(\tau) - \boldsymbol{\beta}(\tau)) \rightarrow N(0, \tau(1-\tau)\mathbf{H}^{-}\boldsymbol{\Omega}\mathbf{H}^{-})$$

where $\mathbf{H} = \lim_{n\rightarrow\infty} n^{-1}\sum \mathbf{x}_i\mathbf{x}_i'/s_i(\tau)$ and $\boldsymbol{\Omega} = \lim_{n\rightarrow\infty} n^{-1}\sum \mathbf{x}_i\mathbf{x}_i'$.

The sparsity function of the ith observation, $s_i(\tau)$, can be estimated as

$$\hat{s}_i(\tau) = \frac{\hat{F}_i^{-1}(\tau + h_n) - \hat{F}_i^{-1}(\tau - h_n)}{2h_n}$$

where $\hat{F}_i^{-1}(\tau \pm h_n) = \mathbf{x}_i'\hat{\boldsymbol{\beta}}(\tau \pm h_n)$ are the quantile predictions of the ith observation at quantile levels $(\tau \pm h_n)$.

Accordingly, the covariance matrix of $\hat{\boldsymbol{\beta}}(\tau)$ can be estimated as

$$\widehat{\mathrm{COV}}\left(\hat{\boldsymbol{\beta}}(\tau)\right) = n^{-1}\tau(1-\tau)\hat{\mathbf{H}}_n^{-}\hat{\boldsymbol{\Omega}}_n\hat{\mathbf{H}}_n^{-}$$

where $\hat{\mathbf{H}}_n = n^{-1}\sum(\mathbf{x}_i\mathbf{x}_i'/\hat{s}_i(\tau))$ and $\hat{\boldsymbol{\Omega}}_n = n^{-1}\sum \mathbf{x}_i\mathbf{x}_i'$.

You can specify the bandwidth method for computing h_n by using either the COV=SPARSITY(BF) option for the Bofinger bandwidth method or the COV=SPARSITY(HF) option for the Hall-Sheather bandwidth method:

- The Bofinger bandwidth optimizes the mean square error for standard density estimation:

  $$h_n = n^{-1/5}\left(4.5v^2(\tau)\right)^{1/5}$$

- The Hall-Sheather bandwidth is based on Edgeworth expansions for studentized quantiles:

  $$h_n = n^{-1/3}{z_\alpha}^{2/3}(1.5v(\tau))^{1/3}$$

  z_α satisfies $T(z_\alpha, df) = 1 - \alpha/2$ for the construction of $1-\alpha$ confidence intervals, where T is the cumulative distribution function for the t distribution and df is the residual degrees of freedom.

The quantity

$$v(\tau) = \frac{s(\tau)}{s^{(2)}(\tau)} = \frac{f^2}{2(f^{(1)}/f)^2 + [(f^{(1)}/f)^2 - f^{(2)}/f]}$$

is not sensitive to f and can be estimated by assuming f is Gaussian as

$$\hat{v}(\tau) = \frac{\exp(-q^2)}{2\pi(q^2+1)}$$

where $q = \Phi^{-1}(\tau)$.

More Statistics for Parameter Estimates

Let $\hat{\beta}_j(\tau)$ denote the jth parameter estimate, and let $\hat{\boldsymbol{\beta}}(\tau) = \left(\hat{\beta}_1(\tau), \ldots, \hat{\beta}_p(\tau)\right)$ denote the vector of the parameter estimates.

PROC QTRSELECT outputs the standard error, t value, and Pr > $|t|$ probability for each $\hat{\beta}_j(\tau)$ in the parameter estimates table. If you specify the CLB option in the MODEL statement, PROC QTRSELECT also outputs confidence limits in the parameter estimates table. Table 28.8 summarizes these statistics for $\hat{\beta}_j(\tau)$.

Table 28.8 More Statistics for $\hat{\beta}_j(\tau)$

Statistic	Definition
Standard error: $\hat{\sigma}_j$	$\sqrt{\widehat{\mathrm{COV}}\left(\hat{\boldsymbol{\beta}}(\tau)\right)_{jj}}$
$(1-\alpha)\%$ confidence limits	$\left(\hat{\beta}_j(\tau) \pm t_{1,1-\frac{\alpha}{2}}\hat{\sigma}_j\right)$
t value	$\hat{\beta}_j(\tau)/\hat{\sigma}_j$
Pr > $\lvert t\rvert$ probability	p-value of the t value

Here $\widehat{\mathrm{COV}}\left(\hat{\boldsymbol{\beta}}(\tau)\right)_{jj}$ is the (j, j) element of $\widehat{\mathrm{COV}}\left(\hat{\boldsymbol{\beta}}(\tau)\right)$, and $t_{1,1-\frac{\alpha}{2}}$ denotes the $\left(1-\frac{\alpha}{2}\right)$-level student's t score with 1 degree of freedom.

Criteria Used in Model Selection

The QTRSELECT procedure supports the following fit statistics that you can use as criteria for the CHOOSE=, SELECT=, and STOP= options in the SELECTION statement:

ADJR1	specifies the adjusted R1 statistic.
AIC	specifies Akaike's information criterion (Akaike 1969; Koenker 2005).
AICC	specifies the corrected Akaike's information criterion (Hurvich and Tsai 1989).
BIC | SBC	specifies the Schwarz Bayesian information criterion (Schwarz 1978; Koenker 2005).
R1	specifies the R1 statistic (Koenker and Machado 1999). The R1 statistic is not valid for the STOP= or CHOOSE= option.
SL	specifies the significance level that is used to assess an effect's contribution to the fit when it is added to or removed from a model. SL is not valid for the CHOOSE= option.
VALIDATE	specifies the average check loss over the validation data.

Table 28.9 provides formulas and definitions for these fit statistics.

Table 28.9 Formulas and Definitions for Model Fit Summary Statistics for Single Quantile Effect Selection

Statistic	Definition or Formula
n	Number of observations
p	Number of parameters, including the intercept
$r_i(\tau)$	Residual for the ith observation; $r_i(\tau) = y_i - \mathbf{x}_i'\hat{\boldsymbol{\beta}}(\tau)$
$D(\tau)$	Total sum of check losses; $D(\tau) = \sum_{i=1}^{n} \rho_\tau(r_i)$. $D(\tau)$ is labeled as Objective Function in the "Fit Statistics" table.
$D_0(\tau)$	Total sum of check losses for intercept-only model if the intercept is a forced-in effect; otherwise for empty model.
$\mathrm{ACL}(\tau)$	Average check loss; $\mathrm{ACL}(\tau) = \dfrac{D(\tau)}{n}$
$\mathrm{R1}(\tau)$	Counterpart of linear regression R square for quantile regression; $\mathrm{R1}(\tau) = 1 - \dfrac{D(\tau)}{D_0(\tau)}$
$\mathrm{ADJR1}(\tau)$	Adjusted R1; $1 - \dfrac{(n-1)D(\tau)}{(n-p)D_0(\tau)}$ if intercept is a forced-in effect; otherwise $1 - \dfrac{nD(\tau)}{(n-p)D_0(\tau)}$.
$\mathrm{AIC}(\tau)$	$2n \ln(\mathrm{ACL}(\tau)) + 2p$
$\mathrm{AICC}(\tau)$	$2n \ln(\mathrm{ACL}(\tau)) + \dfrac{2pn}{n-p-1}$
$\mathrm{SBC}(\tau)$	$2n \ln(\mathrm{ACL}(\tau)) + p \ln(n)$

The $\mathrm{ADJR1}(\tau)$ criterion is equivalent to the generalized approximate cross validation (GACV) criterion for quantile regression (Yuan 2006). The GACV criterion is defined as

$$\mathrm{GACV}(\tau) = D(\tau)/(n-p)$$

which is proportional to $1 - \mathrm{ADJR1}(\tau)$.

Quasi-likelihood Information Criteria

Given the quantile level τ, assume that the distribution of Y_i conditional on $\mathbf{x}_i$ follows the linear model

$$Y_i = \mathbf{x}_i'\boldsymbol{\beta} + \epsilon_i$$

where ϵ_i for $i = 1, \ldots, n$ are iid in distribution F. Further assume that F is an asymmetric Laplace distribution whose density function is

$$f_\tau(r) = \frac{\tau(1-\tau)}{\sigma} \exp\left(-\frac{\rho_\tau(r)}{\sigma}\right)$$

where σ is the scale parameter. Then, the negative log-likelihood function is

$$l_\tau(\boldsymbol{\beta}, \sigma) = n \log(\sigma) + \sigma^{-1} \sum_{i=1}^{n} \rho_\tau(y_i - \mathbf{x}_i'\boldsymbol{\beta}) - n \log(\tau(1-\tau))$$

Under these settings, the maximum likelihood estimate (MLE) of $\boldsymbol{\beta}$ is the same as the relevant level-τ quantile regression solution $\hat{\boldsymbol{\beta}}(\tau)$, and the MLE for σ is

$$\hat{\sigma}(\tau) = n^{-1} \sum_{i=1}^{n} \rho_\tau \left(y_i - \mathbf{x}_i' \hat{\boldsymbol{\beta}}(\tau) \right)$$

where $\hat{\sigma}(\tau)$ equals the level-τ average check loss $\mathrm{ACL}(\tau)$ for the quantile regression solution.

Because the general form of Akaike's information criterion (AIC) is $\mathrm{AIC} = (-2l + 2p)$, the quasi-likelihood AIC for quantile regression is

$$\mathrm{AIC}(\tau) = 2n \ln\left(\mathrm{ACL}(\tau)\right) + 2p$$

where p is the degrees of freedom for the fitted model.

Similarly, the quasi-likelihood AICC (corrected AIC) and SBC (Schwarz Bayesian information criterion) can be formulated as follows:

$$\mathrm{AICC}(\tau) = 2n \ln\left(\mathrm{ACL}(\tau)\right) + \frac{2pn}{n - p - 1}$$

$$\mathrm{SBC}(\tau) = 2n \ln\left(\mathrm{ACL}(\tau)\right) + p \ln(n)$$

In fact, the quasi-likelihood AIC, AICC, and SBC are fairly robust, and you can use them to select effects for data sets without the iid assumption in asymmetric Laplace distribution. For a simulation study that applies SBC for effect selection, see "Example 28.1: Simulation Study" on page 1461. The study generates a data table by using a naive instrumental model (Chernozhukov and Hansen 2008).

Statistical Tests for Significance Level

The QTRSELECT procedure supports the significance level (SL) criterion for effect selection. Consider the general form of a quantile regression model:

$$Q_Y(\tau | \mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1' \boldsymbol{\beta}_1(\tau) + \mathbf{x}_2' \boldsymbol{\beta}_2(\tau)$$

At each step of an effect-selection process, a candidate effect can be represented as $\mathbf{x}_2$, and the significance level of the candidate effect can be calculated by testing the null hypothesis: $H_0 : \boldsymbol{\beta}_2(\tau) = \mathbf{0}$.

The QTRSELECT procedure supports the Wald test for computing candidate significance levels. Let $\hat{\boldsymbol{\beta}}(\tau) = \left(\hat{\boldsymbol{\beta}}_1'(\tau), \hat{\boldsymbol{\beta}}_2'(\tau) \right)'$ be the parameter estimates for the extended model, and denote the estimated covariance matrix of $\hat{\boldsymbol{\beta}}(\tau)$ as

$$\hat{\Sigma}(\tau) = \begin{bmatrix} \hat{\Sigma}_{11}(\tau) & \hat{\Sigma}_{12}(\tau) \\ \hat{\Sigma}_{21}(\tau) & \hat{\Sigma}_{22}(\tau) \end{bmatrix}$$

where $\hat{\Sigma}_{22}(\tau)$ is the covariance matrix for $\hat{\boldsymbol{\beta}}_2(\tau)$. Then the Wald test score is defined as

$$\hat{\boldsymbol{\beta}}_2'(\tau) \hat{\Sigma}_{22}^{-1}(\tau) \hat{\boldsymbol{\beta}}_2(\tau)$$

Under the null hypothesis that the reduced model is the true model, the Wald score follows a χ^2 distribution with degrees of freedom $df = df_2 - df_1$, where df_1 and df_2 are the degrees of freedom for the reduced model and the extended model, respectively .

When you use SL as a criterion for effect selection, the algorithm for estimating sparsity function depends on whether an effect is being considered as an add or a drop candidate. For testing an add candidate effect, the sparsity values, $\hat{s}_i(\tau)$'s, are estimated by using the reduced model that does not include the add candidate effect. For testing a drop candidate effect, the sparsity values are estimated on the extended model that does not exclude the drop candidate effect. Then, these estimated sparsity function values are used to compute the covariance matrix of the parameter estimates for the extended model. However, for the model that is selected at each step, the sparsity function for estimating standard errors and confidence limits of the parameter estimates is estimated on that model itself, but not on the model that was selected at the preceding step.

Because the null hypotheses usually do not hold, the SLENTRY and SLSTAY values cannot reliably be viewed as probabilities. One way to address this difficulty is to replace hypothesis testing with a means of selecting a model with information criteria or out-of-sample prediction criteria.

Diagnostic Statistics

This section gathers the formulas for the statistics available in the OUTPUT statement. All the statistics available in the OUTPUT statement are conditional on the selected model and do not take into account the variability that is introduced by doing model selection.

The model to be fit is $Q_Y(\tau|\mathbf{x}) = \mathbf{x}'\boldsymbol{\beta}(\tau)$, and the parameter estimate $\hat{\boldsymbol{\beta}}(\tau)$ is the solution that minimizes $\sum_{i=1}^{n} \rho_\tau\left(y_i - \mathbf{x}_i'\boldsymbol{\beta}\right)$. The subscript i is for the ith observation. The subscript j is for the jth quantile level of the specified QUANTILES levels in the MODEL statement. $\hat{\Sigma}(\tau)$ denotes the covariance estimation for $\hat{\boldsymbol{\beta}}(\tau)$.

The ALPHA= option in the PROC QTRSELECT statement sets the α value for the confidence limit statistics. The degrees of freedom for $t_{\frac{\alpha}{2}}$ are $n - p$.

Table 28.10 contains the diagnostic statistics and their formulas. Each statistic is computed for each observation.

Table 28.10 Formulas and Definitions for Diagnostic Statistics

MODEL Option or Statistic	Formula
PREDj	$\widehat{y}_{ji} = \mathbf{x}_i'\hat{\boldsymbol{\beta}}(\tau_j)$
RESj	$y_i - \widehat{y}_{ji}$
STDPj	$\sqrt{\mathbf{x}_i'\hat{\Sigma}(\tau_j)\mathbf{x}_i}$
LCLMj	$\widehat{y}_{ji} - t_{\frac{\alpha}{2}}\mathrm{STDP}j_i$
UCLMj	$\widehat{y}_{ji} + t_{\frac{\alpha}{2}}\mathrm{STDP}j_i$

Classification Variables and the SPLIT Option

PROC QTRSELECT supports the ability to split classification variables when you are doing model selection. You use the SPLIT option in the CLASS statement to specify that the columns of the design matrix that correspond to effects that contain a split classification variable can enter or leave a model independently of the other design columns of that effect. The following statements illustrate the use of the SPLIT option:

```
data mylib.splitExample;
   length c2 $6;
   drop i;
   do i=1 to 1000;
     c1 = 1 + mod(i,6);
     if      i < 200 then c2 = 'low';
     else if i < 500 then c2 = 'medium';
     else                 c2 = 'high';
     x1 = ranuni(1);
     x2 = ranuni(1);
     y = x1+3*(c2 ='low')  + 10*(c1=3) +5*(c1=5) + rannor(1);
     output;
   end;
run;

proc qtrselect data=mylib.splitExample;
   class c1(split) c2(order=freq);
   model y = c1 c2 x1 x2;
   selection method=forward;
run;
```

The "Class Level Information" table shown in Figure 28.12 is produced by default whenever you specify a CLASS statement.

Figure 28.12 Class Levels

The QTRSELECT Procedure

Class Level Information		
Class	**Levels**	**Values**
c1	6	1 2 3 4 5 6
c2	3	high medium low

The SPLIT option has been specified for the classification variable c1. This permits the parameters associated with the effect c1 to enter or leave the model individually. The "Parameter Estimates" table in Figure 28.13 shows that for this example the parameters that correspond only to levels 3 and 5 of c1 are in the selected model.

Figure 28.13 Parameter Estimates

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	2.84133	0.11898	23.88	<.0001
c1_3	1	10.15300	0.10121	100.31	<.0001
c1_5	1	5.02683	0.09847	51.05	<.0001
c2 high	1	-3.10623	0.09550	-32.52	<.0001
c2 medium	1	-2.88624	0.11896	-24.26	<.0001
c2 low	0	0	.	.	.
x1	1	1.23941	0.14059	8.82	<.0001

Using Validation and Test Data

When you have sufficient data, you can subdivide your data into three parts called the training, validation, and test data. The selection process fits models to the training data and uses the validation data to find the prediction errors for the models that are obtained in this way. This prediction error on the validation data can be used to decide when to terminate the selection process or to decide what effects to include as the selection process proceeds. Finally, after a selected model has been obtained, the test data can be used to assess how the selected model generalizes on data that played no role in selecting the model.

In some cases you might want to use only training and test data. For example, you might want to use an information criterion to decide what effects to include and when to terminate the selection process. In this case no validation data are required, but test data can still help you assess the predictive performance of the selected model. In other cases you might decide to use validation data during the selection process but forgo assessing the selected model on test data. Hastie, Tibshirani, and Friedman (2001) note that it is difficult to give a general rule for how many observations you should assign to each role. They state that a typical split might be 50% for training and 25% each for validation and testing.

You use a PARTITION statement to logically subdivide the DATA= data table into separate roles. You can name the fractions of the data that you want to reserve as test data and validation data. For example, the following statements randomly subdivide the mylib.inData data table, reserving 50% for training and 25% each for validation and testing:

```
proc qtrselect data=mylib.inData;
  partition fraction(test=0.25 validate=0.25);
  ...
run;
```

In some cases you might need to exercise more control over the partitioning of the input data table. You can do this by naming both a variable in the input data table and a formatted value of that variable that correspond to each role. For example, the following statements assign roles to the observations in the mylib.inData data table based on the value of the variable group in that data table. Observations in which the value of group is “group 1” are assigned to testing, and those whose value is “group 2” are assigned to training. All other observations are ignored.

```
proc qtrselect data=mylib.inData;
  partition roleVar=group(test='group 1' train='group 2')
  ...
run;
```

After you reserve observations for training, validation, and testing, a model fit of the training data is scored on the validation and test data, and the average check loss (ACL) is computed separately for each of these subsets. The ACL for each data role is the sum of check losses for observations in that role divided by the number of observations in that role.

The following statements illustrate the use of the ROLEVAR= option:

```
%let seed=321;
%let n=600;
%let p=10;

data mylib.roleExample;
   array x{&p} x1-x&p;
   length r $8;
   drop i j k;

   do i=1 to &n;
      do j=1 to &p;
         x{j} = ranuni(&seed);
      end;

      y  = x1 + x2 + x3 + ranuni(&seed);

      k = mod(i,3);
      if      k=0 then r = 'train';
      else if k=1 then r = 'validate';
      else if k=2 then r = 'test';
      output;
   end;
run;


proc qtrselect data=mylib.roleExample;
   model y = x1-x&p;
   selection method=forward(select=validate stop=sbc);
   partition rolevar=r(train='train' validate='validate' test='test');
run;
```

The “Number of Observations” table shown in Figure 28.14 displays the number of observations used for training, validation, and testing.

Figure 28.14 Number of Observations

The QTRSELECT Procedure

Number of Observations Read	600
Number of Observations Used	600
Number of Observations Used for Training	200
Number of Observations Used for Validation	200
Number of Observations Used for Testing	200

The "Selection Summary" table shown in Figure 28.15 displays the validation ACL values for each step of the selection process.

Figure 28.15 Selection Summary

The QTRSELECT Procedure

Quantile Level = 0.5
Selection Details

Selection Summary				
Step	**Effect Entered**	**Number Effects In**	**SBC**	**Validation ACL**
0	**Intercept**	1	-588.4130	0.2150
1	**x3**	2	-645.5774	0.1826
2	**x1**	3	-730.3921	0.1478
3	**x2**	4	-831.1414*	0.1241
4	**x4**	5	-827.0261	0.1226*
5	**x6**	6	-821.8472	0.1227
6	**x8**	7	-816.8005	0.1229
*** Optimal Value Of Criterion**				

The "Fit Statistics" table shown in Figure 28.16 displays the training ACL, the validation ACL, and the testing ACL for the final model.

Figure 28.16 Fit Statistics

The QTRSELECT Procedure

Quantile Level = 0.5
Selected Model

Objective Function	23.74758
R1	0.47615
Adj R1	0.46814
AIC	-844.33471
AICC	-844.12958
SBC	-831.14144
ACL (Train)	0.11874
ACL (Validate)	0.12409
ACL (Test)	0.12282

Using the Validation ACL as the SELECT= Criterion

If you provided observations for validation and specified a model selection method that uses the SELECT= criterion, then you can specify SELECT=VALIDATE as a suboption of the METHOD= option in the SELECTION statement. After each step, the selection process computes the validation ACL values for all the current candidate models. The candidate model that has the smallest validation ACL value usually serves as the model for the next selection step.

Using the Validation ACL as the STOP= Criterion

If you provided observations for validation, then you can specify STOP=VALIDATE as a suboption of the METHOD= option in the SELECTION statement. At step k of the selection process, the best candidate effect to enter or leave the current model is determined. Here, "best candidate" means the effect that gives the best value of the SELECT= criterion; this criterion does not need to be based on the validation data. The validation ACL for the model with this candidate effect added or removed is computed. If this validation ACL is greater than the validation ACL for the model at step k, then the selection process terminates at step k.

Using the Validation ACL as the CHOOSE= Criterion

When you specify the CHOOSE=VALIDATE suboption of the METHOD= option in the SELECTION statement, the validation ACL is computed for the models at each step of the selection process. The smallest model at any step that yields the smallest validation ACL is selected.

Computational Method

Multithreading

The QTRSELECT procedure allocates data to different threads and calculates crossproduct matrices by accumulating the contributions from all threads. PROC QTRSELECT also uses multiple threads to compute matrix inverses and select candidates during model selection. For more information about how PROC QTRSELECT uses threads, see the section "Multithreading" on page 102 in Chapter 2, "Shared Concepts."

Displayed Output

The following sections describe the output produced by PROC QTRSELECT. The output is organized into various tables, which are discussed in their order of appearance.

Selection Information

When you specify the SELECTION statement, the QTRSELECT procedure produces by default a series of tables that display information about the model selection. The "Selection Information" table informs you about the model selection method; select, stop, and choose criteria; and other parameters that govern the selection. You can suppress this table by specifying DETAILS=NONE in the SELECTION statement.

Number of Observations

The "Number of Observations" table displays the number of observations read from the input data table and the number of observations used in the analysis. If you use a PARTITION statement, the table also displays the number of observations used for each data role.

Class Level Information

The "Class Level Information" table lists the levels of every variable that you specify in the CLASS statement. You should check this information to make sure that the data are correct. You can adjust the order of the CLASS variable levels by specifying the ORDER= option in the CLASS statement. You can suppress the "Class Level Information" table completely or partially by specifying the NOCLPRINT= option in the PROC QTRSELECT statement.

If the classification variables use a nonsingular parameterization, the "Class Level Information" table also displays the reference value for each variable.

Dimensions

The "Dimensions" table displays the number of effects and the number of parameters from which the selected model is chosen. If you use split classification variables, then this table also displays the number of effects after splitting is taken into account.

Entry and Removal Candidates

When you specify the DETAILS=ALL or DETAILS=STEPS option in the SELECTION statement, the QTRSELECT procedure produces "Entry Candidates" and "Removal Candidates" tables that display the effect names and values of the criterion used to select entering or departing effects at each step of the selection process. The effects are displayed in sorted order from best to worst of the selection criterion.

Selection Summary

When you specify the SELECTION statement, the QTRSELECT procedure produces the "Selection Summary" table, which displays information about the sequence of steps of the selection process. For each step, the effect that was entered or dropped is displayed along with the statistics used to select the effect, stop the selection, and choose the selected model. For all criteria that you can use for model selection, the steps at which the optimal values of these criteria occur are also indicated.

You can suppress the display of the "Selection Summary" table by specifying DETAILS=NONE in the SELECTION statement.

Stop Reason

The "Stop Reason" table displays the reason why the selection stopped. To facilitate programmatic use of this table, an integer code is assigned to each reason and is included if you use an ODS OUTPUT statement or a DISPLAYOUT statement to output this table. The reasons and their associated codes follow:

Code	Stop Reason
1	All eligible effects are in the model.
2	All eligible effects have been removed.
3	Specified maximum number of steps done.
4	The model contains the specified maximum number of effects.
5	The model contains the specified minimum number of effects (for backward selection).
6	The stopping criterion is at a local optimum.
7	No suitable add or drop candidate could be found.
8	Adding or dropping any effect does not improve the selection criterion.
9	No candidate meets the appropriate SLE or SLS significance level.
10	Stepwise selection is cycling.
11	The model is an exact fit.
12	Dropping an effect would result in an empty model.

You can suppress the display of the "Stop Reason" table by specifying DETAILS=NONE in the SELECTION statement.

Selection Reason

When you specify the SELECTION statement, the QTRSELECT procedure produces a simple table that explains why the final model was selected.

You can suppress the display of the "Selection Reason" table by specifying DETAILS=NONE in the SELECTION statement.

Selected Effects

When you specify the SELECTION statement, the QTRSELECT procedure produces a simple table that lists which effects were selected for the final model.

Fit Statistics

The "Fit Statistics" table displays fit statistics for the selected model. The statistics include the following:

- Objective Function, total sum of check losses. Objective Function is denoted as $D(\tau)$ in Table 28.9.
- R1, a measure between 0 and 1 that indicates the portion of the (corrected) total check losses attributed to the fit rather than left to residuals. It is calculated as $1 - \frac{D(\tau)}{D_0(\tau)}$. It is the quantile regression counterpart of the linear regression R square.
- Adj R1, the adjusted R1, a version of R1 that has been adjusted for degrees of freedom. It is calculated as

$$\text{Adj R1} = 1 - \frac{n-i}{n-p}(1 - \text{R1})$$

where $i = 1$ if the intercept is forced in and $i = 0$ otherwise , n is the number of observations used to fit the model, and p is the number of parameters in the model.

- the fit criteria AIC, AICC, and SBC if they are used in the selection process. For the formulas to evaluate these criteria, see Table 28.9.
- the average check loss (ACL) on the training, validation, and test data

You can request the "Fit Statistics" tables for the model at each step of the selection process by specifying the DETAILS= option in the SELECTION statement.

Parameter Estimates

The "Parameter Estimates" table displays the parameters in the selected model and their estimates. The information displayed for each parameter in the selected model includes the following:

- the parameter label that includes the effect name and level information for effects that contain classification variables
- the degrees of freedom (DF) for the parameter. There is one degree of freedom unless the model is not full rank.
- the parameter estimate
- the standard error, which is the estimate of the standard deviation of the parameter estimate
- t Value, the *t* test that the parameter is 0. This is computed as the parameter estimate divided by the standard error.
- the Pr > |t|, the probability that a *t* statistic would obtain a greater absolute value than that observed given that the true parameter is 0. This is the two-tailed significance probability.

  When you do model selection, these *p*-values are usually liberal because they are not adjusted for the fact that the terms in the model have been selected.

You can request a "Parameter Estimates" table for the model at each step of the selection process by specifying the DETAILS= option in the SELECTION statement.

Timing

The "Timing" table displays the amount of time (in seconds) and the percentage of the time that PROC QTRSELECT required to perform different tasks in the analysis.

OutputCasTables Table

The OutputCasTables table is a special table that has information about each CAS table that is created during a CAS action execution. The information for each CAS table consists of the CAS table name, the caslib in which the table resides, and the number of columns and rows in the CAS table. Because this table is not a typical ODS table that contains analytical results, you cannot include it in the *table-spec-list* in the DISPLAYOUT statement.

ODS Table Names

Each table that the QTRSELECT procedure creates has a name associated with it. You must use this name to refer to the table when you use ODS statements. These names are listed in Table 28.11.

NOTE: The EFFECT statement also create tables which are not listed in this section. For information about these tables, see "ODS Table Names" on page 33 in Chapter 2, "Shared Concepts."

Table 28.11 ODS Tables Produced by PROC QTRSELECT

Table Name	Description	Required Statement / Option
Candidates	Swap candidates at step	SELECTION DETAILS=ALL | STEPS
ClassInfo	Level information from the CLASS statement	CLASS
Dimensions	Model dimensions	Default output
EntryCandidates	Candidates for entry at step	SELECTION DETAILS=ALL | STEPS
FitStatistics	Fit statistics	Default output
ModelInfo	Information about the modeling environment	Default output
NObs	Number of observations read and used	Default output
OutputCasTables	See the section "OutputCasTables Table" on page 1458	OUTPUT DISPLAYOUT
ParameterEstimates	Solutions for the parameter estimates associated with effects in the MODEL statement	Default output
RemovalCandidates	Candidates for removal at step	SELECTION DETAILS=ALL | STEPS
SelectedEffects	List of selected effects	SELECTION
SelectionInfo	Information about selection settings	Default output
SelectionReason	Reason for selecting the final model	SELECTION
SelectionSummary	Summary information about the model selection steps	SELECTION
StopReason	Reason selection was terminated	SELECTION
Timing	Timing breakdown by task	SELECTION DETAILS

ODS Graphics

Statistical procedures use ODS Graphics to create graphs as part of their output. ODS Graphics is described in detail in Chapter 24, "Statistical Graphics Using ODS" (*SAS/STAT User's Guide*).

When ODS Graphics is enabled, then the SELECTION statement can produce plots to help evaluate the selection process. For information about these plots, see the section "SELECTION Statement" on page 42 in Chapter 2, "Shared Concepts."

PROC QTRSELECT assigns a name to each graph that it creates using ODS. You can use these names to reference the graphs when using ODS. The names are listed in Table 28.12.

Table 28.12 Graphs Produced by PROC QTRSELECT

ODS Graph Name	Plot Description	PLOTS Option
CoefficientPanel	Coefficients and CHOOSE= criterion by step	COEFFICIENTS
ChooseCriterionPlot	CHOOSE= criterion by step	COEFFICIENTS(UNPACK)
CoefficientPlot	Coefficients by step	COEFFICIENTS(UNPACK)
CriterionPanel	Fit criteria by step	CRITERIA
AdjR1Plot	Adjusted R1 by step	CRITERIA(UNPACK)
AICCPlot	Corrected Akaike's information criterion by step	CRITERIA(UNPACK)
AICPlot	Akaike's information criterion by step	CRITERIA(UNPACK)
SBCPlot	Schwarz Bayesian information criterion by step	CRITERIA(UNPACK)
TEST_ACLPlot	Average check loss on testing data by step	CRITERIA(UNPACK)
TRAIN_ACLPlot	Average check loss on training data by step	CRITERIA(UNPACK)
VAL_ACLPlot	Average check loss on validation data by step	CRITERIA(UNPACK)
FitByRolePlot	Progression of average check loss by role	FITBYROLE

Examples: QTRSELECT Procedure

The examples in this chapter are available in the GitHub repository located at https://github.com/sascommunities/doc-supplement-casstat.

NOTE: Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections "Using CAS Sessions and CAS Engine Librefs" on page 7 and "Loading a SAS Data Set onto a CAS Server" on page 7 in Chapter 2, "Shared Concepts."

Example 28.1: Simulation Study

(View the complete code for this example (qtrselex1.sas) in the example repository.)

This example is based on the section "Simulation Study" (Examples: QUANTSELECT Procedure, *SAS/STAT User's Guide*). This simulation study shows how you can use the forward selection method to select quantile regression models for single quantile levels. The following statements simulate a data set from a naive instrumental model (Chernozhukov and Hansen 2008):

```
%let seed=321;
%let p=20;
%let n=3000;

data analysisData;
   array x{&p} x1-x&p;
   do i=1 to &n;
      U  = ranuni(&seed);
      x1 = ranuni(&seed);
      x2 = ranexp(&seed);
      x3 = abs(rannor(&seed));
      y  = x1*(U-0.1) + x2*(U*U-0.25) + x3*(exp(U)-exp(0.9));
      do j=4 to &p;
         x{j} = ranuni(&seed);
      end;
      output;
   end;
run;
```

Variable U in the data set indicates the true quantile level of the response y conditional on $\mathbf{x} = (x_1, \ldots, x_p)'$.

Let $Q_Y(\tau|\mathbf{x}) = \mathbf{x}'\boldsymbol{\beta}(\tau)$ denote the underlying quantile regression model, where $\boldsymbol{\beta}(\tau) = (\beta_1(\tau), \ldots, \beta_p(\tau))'$. Then, the true parameter functions are

$$\begin{aligned}
\beta_1(\tau) &= \tau - 0.1 \\
\beta_2(\tau) &= \tau^2 - 0.25 \\
\beta_3(\tau) &= \exp(\tau) - \exp(0.9) \\
\beta_4(\tau) &= \cdots = \beta_p(\tau) = 0
\end{aligned}$$

It is easy to see that, at $\tau = 0.1$, only $\beta_2(0.1) = -0.24$ and $\beta_3(0.1) = \exp(0.1) - \exp(0.9) \approx -1.354432$ are nonzero parameters. Therefore, an effective effect-selection method should select x2 and x3 and drop all the other effects in this data set at $\tau = 0.1$. By the same rationale, x1 and x3 should be selected at $\tau = 0.5$ with $\beta_1(0.5) = 0.4$ and $\beta_3(0.5) \approx -0.810882$, and x1 and x2 should be selected at $\tau = 0.9$ with $\beta_1(0.9) = 0.8$ and $\beta_2(0.9) = 0.56$.

You can load the analysisData data set into your CAS session by specifying your CAS engine libref in the first statement in the following DATA step:

```
data mylib.analysisData;
   set analysisData;
run;
```

The following statements use PROC QTRSELECT with the forward selection method. The STB option and the CLB option in the MODEL statement request the standardized parameter estimates and the confidence limits of parameter estimates, respectively.

```
proc qtrselect data=mylib.analysisData;
   model y= x1-x&p / quantile=0.1 0.5 0.9 clb;
   selection method=forward;
   output out=mylib.out p=pred;
run;
```

Output 28.1.1 shows that, by default, the CHOOSE= and STOP= options are both set to SBC.

Output 28.1.1 Selection Information

The QTRSELECT Procedure

Selection Information	
Selection Method	Forward
Select Criterion	SBC
Stop Criterion	SBC
Effect Hierarchy Enforced	None
Stop Horizon	3

Output 28.1.2, Output 28.1.3, and Output 28.1.4 display the selected effects and the parameter estimates for $\tau = 0.1$, $\tau = 0.5$, and $\tau = 0.9$, respectively. You can see that the forward selection method correctly selects active effects for all three quantile levels.

Output 28.1.2 Parameter Estimates at $\tau = 0.1$

Selected Effects:	Intercept x2 x3

Parameter Estimates

Parameter	DF	Estimate	Standard Error	95% Confidence Limits		t Value	Pr > |t|
Intercept	1	0.01179	0.01192	-0.01158	0.03516	0.99	0.3225
x2	1	-0.22871	0.00946	-0.24725	-0.21017	-24.19	<.0001
x3	1	-1.37991	0.01556	-1.41042	-1.34939	-88.67	<.0001

Output 28.1.3 Parameter Estimates at $\tau = 0.5$

Selected Effects: Intercept x1 x3

Parameter Estimates

Parameter	DF	Estimate	Standard Error	95% Confidence Limits		t Value	Pr > |t|
Intercept	1	0.01178	0.03418	-0.05524	0.07879	0.34	0.7304
x1	1	0.42584	0.06237	0.30355	0.54814	6.83	<.0001
x3	1	-0.86332	0.04765	-0.95674	-0.76989	-18.12	<.0001

Output 28.1.4 Parameter Estimates at $\tau = 0.9$

Selected Effects: Intercept x1 x2

Parameter Estimates

Parameter	DF	Estimate	Standard Error	95% Confidence Limits		t Value	Pr > |t|
Intercept	1	-0.00774	0.03292	-0.07228	0.05680	-0.24	0.8142
x1	1	0.78294	0.05134	0.68228	0.88360	15.25	<.0001
x2	1	0.57644	0.03422	0.50935	0.64354	16.85	<.0001

Example 28.2: Growth Charts for Body Mass Index

(View the complete code for this example (qtrselex2.sas) in the example repository.)

This example is modeled on an example in the section "Getting Started: QUANTREG Procedure" in the *SAS/STAT User's Guide*. This example highlights the use of the QTRSELECT procedure for multiple-level quantile regression by creating growth charts for men's body mass index (BMI).

BMI, which is defined as the ratio of weight (kg) to squared height (m^2), is a standard measure for categorizing individuals as overweight or underweight. The percentiles of BMI for specified ages are of particular interest. This example draws smooth BMI quantile curves conditional on Age, which can serve as BMI growth charts in medical diagnoses to identify BMI percentiles for subjects.

The Sashelp.BMIMen data set records 1999–2000 and 2001–2002 survey results for men's BMI. These survey results are published by the National Center for Health Statistics. The data set contains 3,264 observations for two variables, BMI and Age.

The following statements display the variables in the data set. Output 28.2.1 shows the results.

```
proc contents varnum data=sashelp.BMIMen;
   ods select position;
run;
```

Output 28.2.1 Sashelp.BMIMen Data Set

The CONTENTS Procedure

Variables in Creation Order			
#	Variable	Type	Len
1	Age	Num	8
2	BMI	Num	8

You can load the Sashelp.BMIMen data set into your CAS session by specifying your CAS engine libref in the first statement in the following DATA step:

```
data mylib.BMIMen;
   set sashelp.BMIMen;
   SqrtAge = sqrt(Age);
   InveAge = 1/Age;
   LogBMI  = log(BMI);
run;
```

The logarithm of BMI is used as the response. The following statements fit quantile regression models for the mylib.BMIMen data table at 10 quantile levels:

```
%let quantile=0.03 0.05 0.1 0.25 0.5 0.75 0.85 0.90 0.95 0.97;
%let nq=10;

proc qtrselect data=mylib.BMIMen;
   model logBMI = InveAge SqrtAge Age SqrtAge*Age Age*Age Age*Age*Age
         / quantile=&quantile;
   output out=mylib.BMIOut copyvars=(BMI Age) pred=P_LogBMI;
run;
```

The OUTPUT statement outputs the mean predicted quantiles for the 10 specified quantile levels. The PRED= option in the OUTPUT statement specifies the variable names for the quantile predictions. For example, p1 is for quantile level 0.03, and p2 is for quantile level 0.05.

The following statements define and apply a SAS macro function to create a quantile curves plot for the mylib.BMIOut data table:

```
%let BMIcolor=red olive orange blue brown gray violet black gold green;

%macro plotBMI;
   data BMIPred;
      set mylib.BMIOut;
      %do j=1 %to &nq;
         predBMI&j = exp(P_LogBMI&j);
      %end;
      label %do j=1 %to &nq;
               predBMI&j=%qscan(&quantile,&j,%str( ))
            %end;;
   run;

   proc sort data=BMIPred;
      by Age;
```

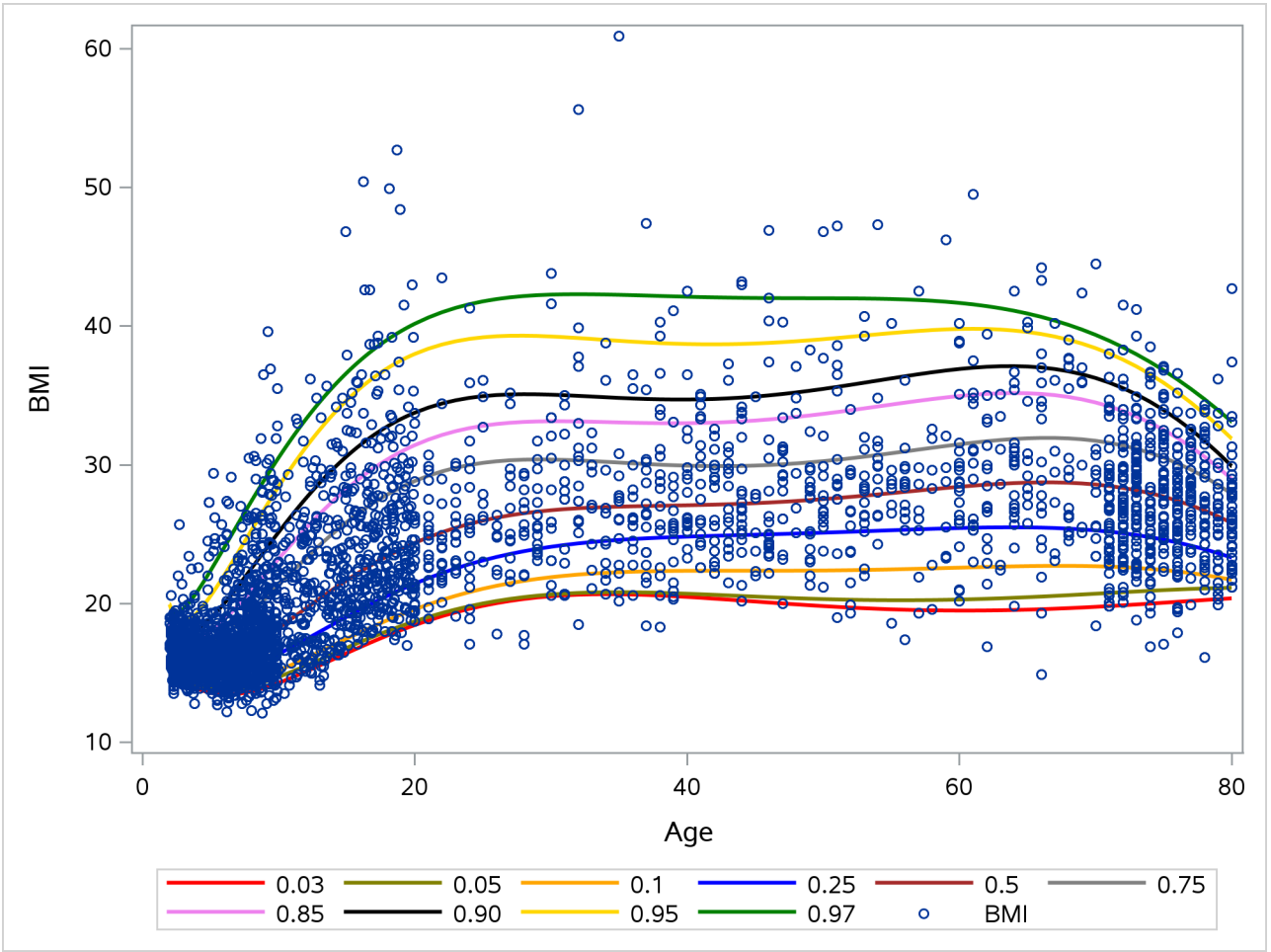
```
   run;

   proc sgplot data=BMIPred;
      %do j=1 %to &nq;
         series y=predBMI&j x=Age/lineattrs=(thickness=2
         color=%qscan(&BMIcolor,&j,%str( )));
      %end;
      scatter y=BMI x=Age/markerattrs=(size=5);
   run;
%mend;

%plotBMI;
```

Output 28.2.2 shows the BMI quantile curves, which can serve as BMI growth charts. For example, the percentiles of any observations (small blue circles) that are located between the top 0.95 quantile (gold) curve and the 0.97 quantile (green) curve are between the 95th percentile and the 97th percentile. By using this rule, you can measure the percentile range for any observations of interest.

Output 28.2.2 Growth Chart for Body Mass Index



Example 28.3: Pollution and Mortality

(View the complete code for this example (qtrselex3.sas) in the example repository.)

This example shows how you can produce the ODS graphical summary for the effect selection process, and how you can use the PARTITION statement and other options to control the effect selection process. The data for this example come from a study by McDonald and Schwing (1973). The data set contains 60 observations, 15 covariates, and one response variable. The response variable is the total age-adjusted mortality rate for Standard Metropolitan Statistical Areas in the United States in 1959–1961.

The following DATA step creates the Mortality data set:

```
data mortality;
   input index aap ajant ajult size65 nph nsch25 nfek ppsm snwp nowk nin3k
   hpi nopi sdpi datm DeathRate;
   label index="the index"
      aap="Average Annual Precipitation"
      ajant="Average January Temperature"
      ajult="Average July Temperature"
      size65="Size of Population Older Than 65"
      nph="Number of Members per Household"
      nsch25="Number of Years of Schooling for Persons over 25"
      nfek="Number of Households with Fully Equipped Kitchens"
      ppsm="Population per Square Mile"
      snwp="Size of the Nonwhite Population"
      nowk="Number of Office Workers"
      nin3k="Number of Families with an Income Less Than $3000"
      hpi="Hydrocarbon Pollution Index"
      nopi="Nitric Oxide Pollution Index"
      sdpi="Sulfur Dioxide Pollution Index"
      datm="Degree of Atmospheric Moisture"
      DeathRate="Age-Adjusted Death Rate: Deaths per 100,000 Population";
   datalines;
 1  36  27  71   8.1  3.34  11.4  81.5  3243   8.8  42.6  11.7   21
    15   59  59   921.870
 2  35  23  72  11.1  3.14  11.0  78.8  4281   3.6  50.7  14.4    8
    10   39  57   997.875
 3  44  29  74  10.4  3.21   9.8  81.6  4260   0.8  39.4  12.4    6
     6   33  54   962.354
 4  47  45  79   6.5  3.41  11.1  77.5  3125  27.1  50.2  20.6   18
     8   24  56   982.291
 5  43  35  77   7.6  3.44   9.6  84.6  6441  24.4  43.7  14.3   43

   ... more lines ...

    11   42  56  1003.502
58  45  24  70  11.8  3.25  11.1  79.8  3678   1.0  44.8  14.0    7
     3    8  56   895.696
59  42  83  76   9.7  3.22   9.0  76.2  9699   4.8  42.2  14.5    8
     8   49  54   911.817
60  38  28  72   8.9  3.48  10.7  79.8  3451  11.7  37.5  13.0   14
    13   39  58   954.442
;
```

The following statements load the mylib.Mortality data set into your CAS session.

```
data mylib.Mortality;
   set Mortality;
run;
```

The following statements fit a median model for mortality rate conditional on a set of climate, demographic, and pollution covariates by using the forward selection method:

```
ods graphics on;
proc qtrselect data=mylib.Mortality;
   partition fraction(validate=0.3 seed=8);
   model DeathRate = aap aap*aap ajant ajant*ajant ajult
      ajult*ajult size65 size65*size65 nph nph*nph nsch25
      nsch25*nsch25 nfek nfek*nfek  ppsm ppsm*ppsm snwp snwp*snwp
      nowk nowk*nowk nin3k nin3k*nin3k hpi hpi*hpi nopi
      nopi*nopi sdpi sdpi*sdpi datm datm*datm
      / quantile=0.5;
   selection method=forward(choose=validate) sh=8 hier=single
      plot=all;
   output out=mylib.OutData copyvar=DeathRate p=Pred role=Role;
run;

proc print data=mylib.OutData(obs=10); run;
```

Because linear terms alone might not be sufficient to fit this model, quadratic terms are also added in the MODEL statement. The FRACTION option of the PARTITION statement requests that 30% of the observations be used for validation and the remaining 70% of the observations be used for training. The SEED suboption in the FRACTION option specifies the random seed for partitioning the data. The HIER=SINGLE option in the MODEL statement forces the effect selection process to ignore quadratic effect candidates if their corresponding main effects are not in the model. The OUTPUT statement creates a SAS data set named mylib.OutData, which contains the variable Role for observation roles and the variable Pred for median prediction conditional on final selected effects.

Output 28.3.1 shows the selection summary. You can see that the best model is at step 10 for validation ACL and step 14 for the SBC.

Output 28.3.1 Selection Summary

The QTRSELECT Procedure

Quantile Level = 0.5
Selection Details

Selection Summary				
Step	Effect Entered	Number Effects In	SBC	Validation ACL
0	Intercept	1	276.4532	27.5142
1	snwp	2	252.1691	22.0040
2	sdpi	3	241.4058	19.2247
3	aap	4	229.3885	17.1180
4	nfek	5	227.7900	17.8873
5	snwp*snwp	6	228.6916	17.2618
6	ajant	7	229.7307	15.9206
7	ppsm	8	231.0825	13.9243*
8	aap*aap	9	230.7470	14.7606
9	ajult	10	224.4233*	15.5726
10	nfek*nfek	11	226.7077	14.0834
11	size65	12	226.8650	15.3115
12	datm	13	228.7751	16.8086
13	sdpi*sdpi	14	231.3286	15.6029
14	ppsm*ppsm	15	234.0065	16.2998
15	ajult*ajult	16	236.2907	15.3831
16	nph	17	238.7730	15.6733
17	nin3k	18	240.6393	16.4520
* Optimal Value Of Criterion				

Output 28.3.2 shows the selected effects and the relevant estimates.

Output 28.3.2 Parameter Estimates

Selected Effects: Intercept aap ajant nfek ppsm snwp snwp*snwp sdpi

The QTRSELECT Procedure

Quantile Level = 0.5
Selected Model

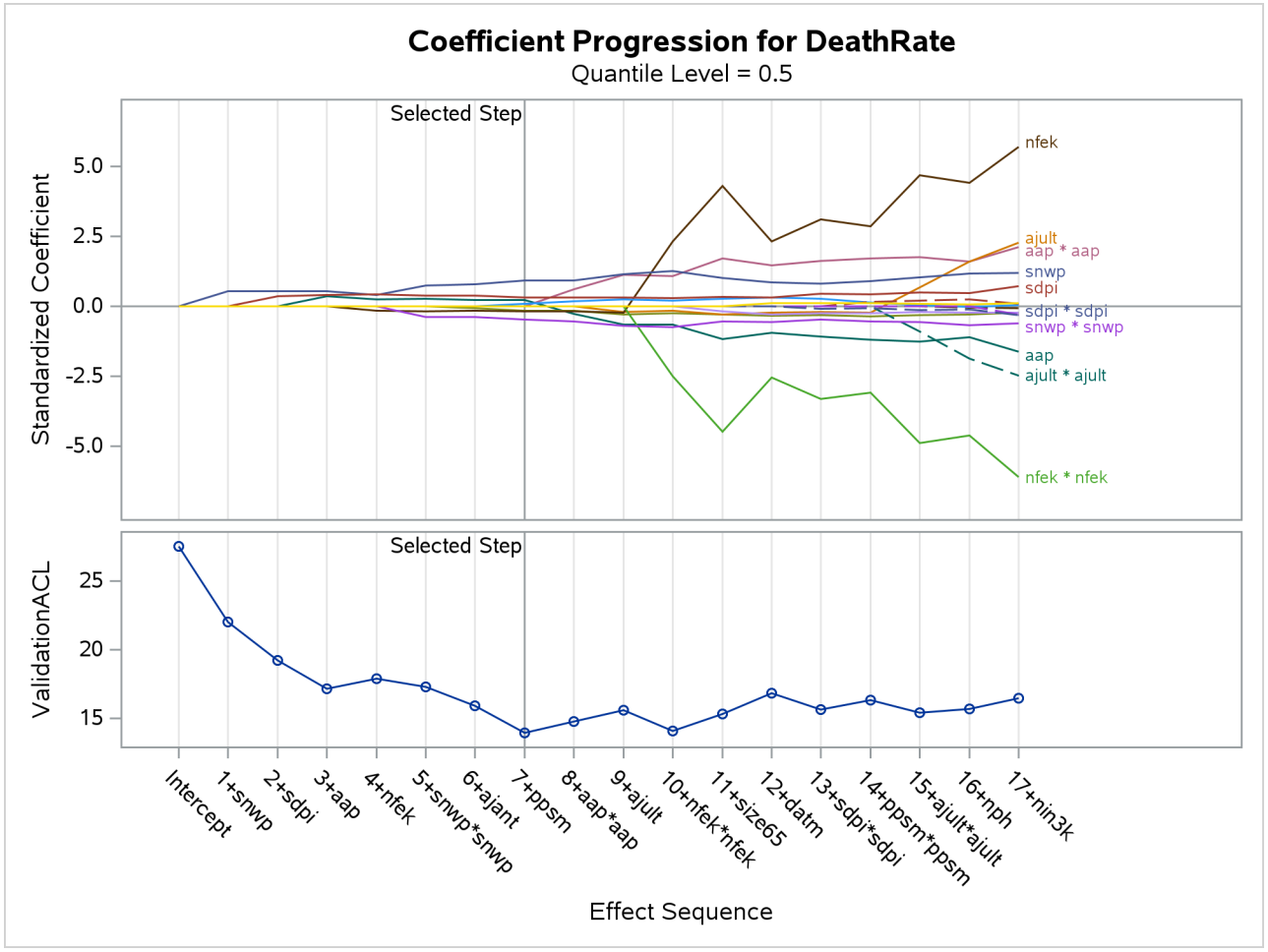
Objective Function	445.11257
R1	0.56557
Adj R1	0.47868
AIC	216.99293
AICC	221.22823
SBC	231.08253
ACL (Train)	10.35146
ACL (Validate)	13.92430

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	1023	77.77882	13.15	<.0001
aap	1	1.48497	0.48729	3.05	0.0044
ajant	1	-0.78146	0.40199	-1.94	0.0600
nfek	1	-2.37270	0.80270	-2.96	0.0055
ppsm	1	0.00327	0.00379	0.86	0.3939
snwp	1	6.00729	2.02197	2.97	0.0053
snwp * snwp	1	-0.09451	0.07153	-1.32	0.1949
sdpi	1	0.28182	0.03472	8.12	<.0001

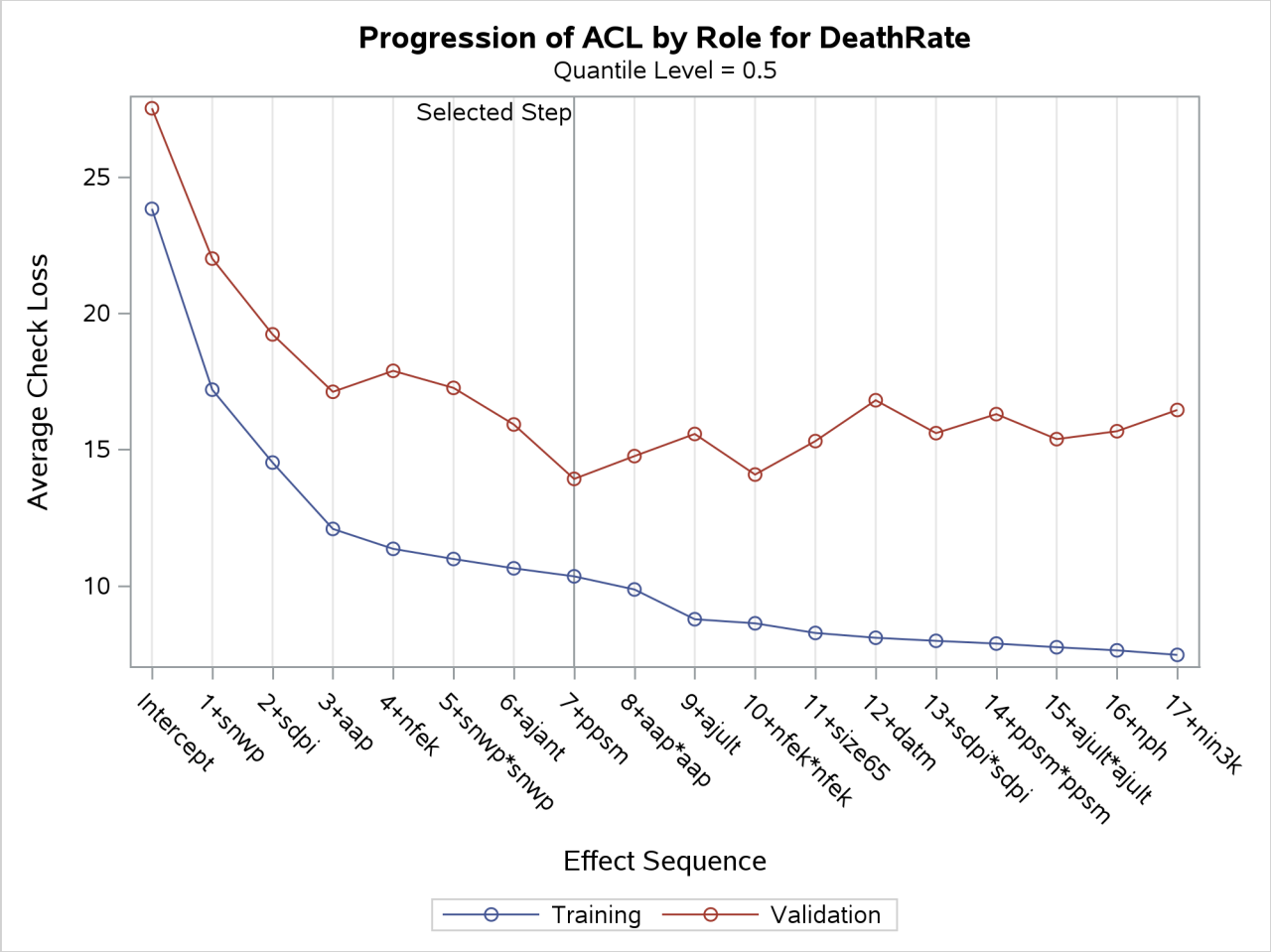
Output 28.3.3 shows the progression of the standardized parameter estimates as the selection process proceeds.

Output 28.3.3 Coefficient Panel



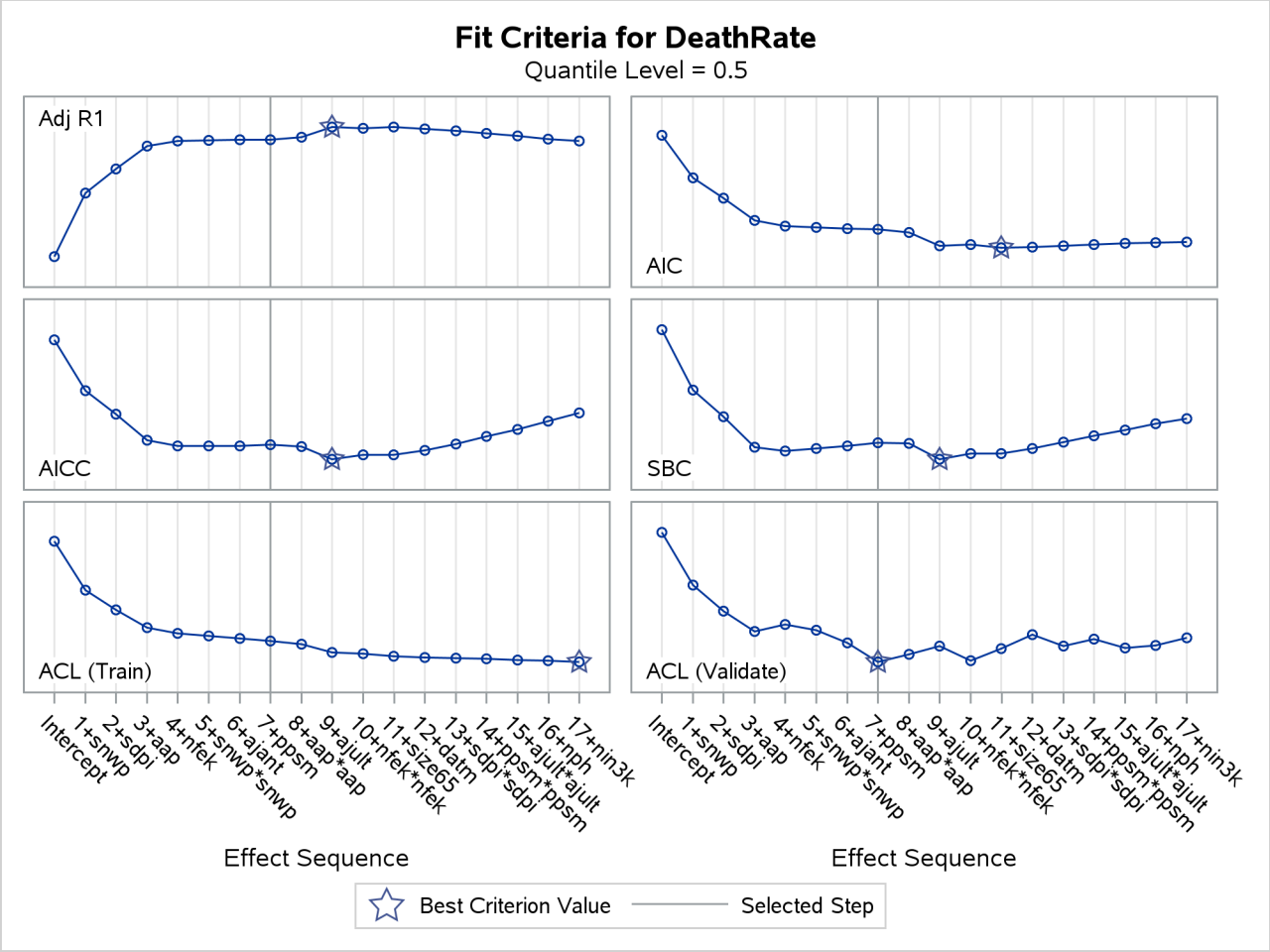
Output 28.3.4 shows the progression of the average check losses for training data and validation data as the selection process proceeds.

Output 28.3.4 Average Check Loss Plot



Output 28.3.5 shows the progression of six effect selection criteria as the selection process proceeds.

Output 28.3.5 Criterion Panel



Output 28.3.6 shows the first 10 observations of the OUTPUT data set.

Output 28.3.6 OUTPUT Data Set

Obs	Role	Pred	DeathRate
1	2	934.659	921.870
2	1	984.009	982.291
3	2	916.885	934.700
4	1	922.525	912.347
5	1	989.710	970.467
6	1	922.537	860.101
7	2	963.488	959.221
8	2	895.263	871.338
9	1	952.529	952.529
10	1	909.981	844.053

References

Akaike, H. (1969). "Fitting Autoregressive Models for Prediction." *Annals of the Institute of Statistical Mathematics* 21:243–247.

Chernozhukov, V., and Hansen, C. (2008). "Instrumental Variable Quantile Regression: A Robust Inference Approach." *Journal of Econometrics* 142:379–398.

Collier Books (1987). *The 1987 Baseball Encyclopedia Update*. New York: Macmillan.

Gertz, E. M., and Griffin, J. D. (2005). *Support Vector Machine Classifiers for Large Data Sets*. Technical Report ANL/MCS-TM-289, Mathematics and Computer Science Division, Argonne National Laboratory.

Gertz, E. M., and Griffin, J. D. (2010). "Using an Iterative Linear Solver in an Interior-Point Method for Generating Support Vector Machines." *Computational Optimization and Applications* 47:431–453.

Hastie, T. J., Tibshirani, R. J., and Friedman, J. H. (2001). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York: Springer-Verlag.

Hurvich, C. M., and Tsai, C.-L. (1989). "Regression and Time Series Model Selection in Small Samples." *Biometrika* 76:297–307.

Koenker, R. (2005). *Quantile Regression*. New York: Cambridge University Press.

Koenker, R., and Bassett, G. W. (1978). "Regression Quantiles." *Econometrica* 46:33–50.

Koenker, R., and Machado, A. F. (1999). "Goodness of Fit and Related Inference Processes for Quantile Regression." *Journal of the American Statistical Association* 94:1296–1310.

McDonald, G. C., and Schwing, R. C. (1973). "Instabilities of Regression Estimates Relating Air Pollution to Mortality." *Technometrics* 15:463–481.

Reichler, J. L., ed. (1987). *The 1987 Baseball Encyclopedia Update*. New York: Macmillan.

Schwarz, G. (1978). "Estimating the Dimension of a Model." *Annals of Statistics* 6:461–464.

Time Inc. (1987). "What They Make." *Sports Illustrated* (April 20): 54–81.

Yuan, M. (2006). "GACV for Quantile Smoothing Splines." *Computational Statistics and Data Analysis* 50:813–829. http://econpapers.repec.org/article/eeecsdana/v_3a50_3ay_3a2006_3ai_3a3_3ap_3a813-829.htm.

Chapter 29
REGSELECT Procedure

Contents

Overview: REGSELECT Procedure **1476**
PROC REGSELECT Features 1476
PROC REGSELECT Compared with Other SAS Procedures 1478
Using CAS Sessions and CAS Engine Librefs 1479
Getting Started: REGSELECT Procedure **1480**
Syntax: REGSELECT Procedure **1488**
PROC REGSELECT Statement 1488
BY Statement 1490
CLASS Statement 1490
CODE Statement 1491
DISPLAY Statement 1491
DISPLAYOUT Statement 1492
EFFECT Statement 1493
FREQ Statement 1495
MODEL Statement 1495
OUTPUT Statement 1497
PARTITION Statement 1500
SELECTION Statement 1500
STORE Statement 1502
WEIGHT Statement 1502
Details: REGSELECT Procedure **1503**
Criteria Used in Model Selection 1503
Diagnostic Statistics 1505
Classification Variables and the SPLIT Option 1506
Using Validation and Test Data 1507
Computational Method 1508
Ridge Regression 1508
Joint Tests and Type 3 Tests 1509
Displayed Output 1510
ODS Table Names 1515
ODS Graphics 1517
Examples: REGSELECT Procedure **1519**
Example 29.1: Stepwise Selection with Validation 1519
Example 29.2: LASSO and Elastic Net Selection 1528
Example 29.3: SCAD and MCP Selection 1544
References **1550**

Overview: REGSELECT Procedure

The REGSELECT procedure fits and performs model selection for ordinary least squares regression models in SAS Viya.

The models that PROC REGSELECT supports can contain main effects that consist of both continuous and classification variables and interaction effects of these variables. The models can also include constructed effects such as splines. The procedure offers a number of effect-selection methods, including stepwise methods and modern LASSO methods. It also offers extensive capabilities for customizing the model selection by using a wide variety of selection and stopping criteria, from computationally efficient significance-level-based criteria to modern, computationally intensive validation-based criteria. PROC REGSELECT also provides a variety of regression diagnostics that are conditional on the selected model.

PROC REGSELECT Features

The main features of the REGSELECT procedure are as follows:

- **Model specification**
  - supports multiple parameterizations for classification effects
  - supports any degree of interaction (crossed effects) and nested effects
  - supports a hierarchy among effects
  - supports partitioning of data into training, validation, and testing roles
  - provides a FREQ statement for grouped analysis
  - provides a WEIGHT statement for weighted analysis
- **Selection control**
  - provides multiple effect-selection methods
  - offers selection of individual levels of classification effects
  - provides effect selection based on a variety of selection criteria
  - provides stopping rules based on a variety of model evaluation criteria
- **Display and output**
  - produces output data tables that contain predicted values, residuals, studentized residuals, confidence limits, and influence statistics
  - uses ODS Graphics to create model selection plots as part of its output. For more information about ODS Graphics, see the section "ODS Graphics" on page 1517.
  - saves fitted models to analytic stores so that you can use them to make predictions later

The REGSELECT procedure supports the following effect-selection methods. For a more detailed description of these methods, see the section "SELECTION Statement" on page 42 in Chapter 2, "Shared Concepts."

- Forward selection starts with no effects in the model and adds effects.
- Backward elimination starts with all effects in the model and deletes effects.
- Stepwise regression is similar to forward selection, except that effects already in the model do not necessarily stay there.
- Forward-swap selection is an extension of the forward selection method. Before any addition step, PROC REGSELECT makes all pairwise swaps of one effect in the model and one effect out of the current model that improve the selection criterion. When the selection criterion is R-square, this method is the same as the MAXR method in the REG procedure in SAS/STAT software.
- Least angle regression (LAR), like forward selection, starts with no effects in the model and adds effects. The parameter estimates at any step are "shrunk" when compared to the corresponding least squares estimates.
- LASSO adds and deletes parameters based on a version of ordinary least squares in which the sum of the absolute regression coefficients is constrained. PROC REGSELECT also supports adaptive LASSO selection, in which weights are applied to each of the parameters in forming the LASSO constraint.
- Elastic net is an extension of LASSO that estimates parameters by using a version of ordinary least squares in which both the sum of the absolute regression coefficients and the sum of the squared regression coefficients are constrained.
- Best-subset selection uses the branch-and-bound algorithm (Furnival and Wilson 1974) to efficiently find subsets of model effects that best predict the dependent variable without examining all possible subsets.
- Smoothly clipped absolute deviation (SCAD) and minimax concave penalty (MCP) selection methods use folded concave penalties (FCP) in regularized linear regression to produce shrinkage estimators.

Hybrid versions of the LAR and LASSO are also supported. They use the LAR or LASSO to select the model, but they estimate the regression coefficients by ordinary weighted least squares.

Because the REGSELECT procedure runs on CAS, it also does the following:

- enables you to run on a cluster of machines that distribute the data and the computations
- enables you to run in single-machine mode on CAS
- exploits all the available cores and concurrent threads. For information about how PROC REGSELECT uses threads, see the section "Multithreading" on page 102 in Chapter 2, "Shared Concepts."

PROC REGSELECT Compared with Other SAS Procedures

The REGSELECT procedure provides regression modeling functionality that is comparable to that of the HPREG, GLMSELECT, and REG procedures in SAS/STAT software.

PROC REGSELECT Compared with the HPREG Procedure

The functionality of the REGSELECT procedure closely resembles that of the HPREG procedure, which is a high-performance procedure. The REGSELECT procedure is the next generation of the HPREG procedure, and it was developed specifically for SAS Viya. Both procedures are designed to run on a cluster of machines that distribute the data and the computations.

Both the REGSELECT and HPREG procedures fit and perform model selection for ordinary least squares regression models, which you can specify as general linear models that include classification variables. You request model selection by using the SELECTION statement.

The REGSELECT procedure offers all the methods of effect selection that the HPREG procedure supports; in addition, PROC REGSELECT supports the elastic net selection method, the best-subset selection method, the SCAD method, and the MCP method. PROC REGSELECT also produces selection plots by using ODS Graphics. The REGSELECT and HPREG procedures offer the same methods of customizing the model selection, and the same set of regression diagnostics. The main differences in functionality are that the REGSELECT procedure provides the ability to specify constructed effects with the EFFECT statement, and it computes Type 3 tests of effects.

PROC REGSELECT Compared with the GLMSELECT Procedure

The functionality of the REGSELECT procedure also closely resembles that of the GLMSELECT procedure. Both procedures fit and perform model selection for ordinary least squares regression models, which you can specify as general linear models that include classification variables. Both procedures offer the same methods of effect selection (including the LAR, LASSO, and elastic net methods) and extensive options to customize the selection process. The REGSELECT procedure supports the best-subset selection method, the SCAD method, and the MCP method, none of which the GLMSELECT procedure supports. Both procedures provide the ability to specify constructed effects with the EFFECT statement.

Unlike the GLMSELECT procedure, the REGSELECT procedure does not perform model selection by default. If you request model selection by using the SELECTION statement, then the default selection method is stepwise selection based on the Schwarz Bayesian information criterion (SBC). This default matches the default method in PROC GLMSELECT. The REGSELECT procedure does not support using cross validation as a model selection criterion.

With the REGSELECT procedure—but not with the GLMSELECT procedure—you can request observationwise residual and influence diagnostics in the OUTPUT statement and variance inflation and tolerance statistics for the parameter estimates. If the fitted model has been obtained by performing model selection, then these statistics are conditional on the selected model and do not take into account the variability introduced by the selection process.

PROC REGSELECT Compared with the REG Procedure

A major functional difference between the REGSELECT procedure and the REG procedure is that the REGSELECT procedure enables you to specify general linear models that include classification variables. In this respect it is similar to the GLM and GLMSELECT procedures in SAS/STAT.

Unlike the REG procedure, the REGSELECT procedure supports the LAR, LASSO and elastic net methods, the ability to use external validation data as selection criteria, and extensive options to customize the selection process.

Like the REG procedure, PROC REGSELECT does not perform model selection by default. If you request model selection by using the SELECTION statement in PROC REGSELECT, then the default selection method is stepwise selection based on the Schwarz Bayesian information criterion (SBC).

With the REGSELECT procedure, as with the REG procedure, you can request observationwise residual and influence diagnostics in the OUTPUT statement, and variance inflation and tolerance statistics for the parameter estimates. If the fitted model has been obtained by performing model selection, then these statistics are conditional on the selected model and do not take into account the variability introduced by the selection process.

Using CAS Sessions and CAS Engine Librefs

SAS Cloud Analytic Services (CAS) is the analytic server and associated cloud services in SAS Viya. This section describes how to create a CAS session and set up a CAS engine libref that you can use to connect to the CAS session. It assumes that you have a CAS server already available; contact your system administrator if you need help starting and terminating a server. This CAS server is identified by specifying the host on which it runs and the port on which it listens for communications. To simplify your interactions with this CAS server, the host information and port information for the server are stored as SAS option values that are retrieved automatically whenever this CAS server needs to be accessed. You can examine the host and port values for the server at your site by using the following statements:

```
proc options option=(CASHOST CASPORT);
run;
```

In addition to starting a CAS server, your system administrator might also have created a CAS session and a CAS engine libref for your use. You can define your own sessions and CAS engine librefs that connect to the CAS server as shown in the following statements:

```
cas mysess;
libname mylib cas sessref=mysess;
```

The CAS statement creates the CAS session named mysess, and the LIBNAME statement creates the mylib CAS engine libref that you use to connect to this session. It is not necessary to explicitly name the CASHOST and CASPORT of the CAS server in the CAS statement, because these values are retrieved from the corresponding SAS option values.

If you have created the mysess session, you can terminate it by using the TERMINATE option in the CAS statement as follows:

```
cas mysess terminate;
```

For more information about the CAS and LIBNAME statements, see the section "Introduction to Shared Concepts" on page 6 in Chapter 2, "Shared Concepts."

Getting Started: REGSELECT Procedure

NOTE: Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections "Using CAS Sessions and CAS Engine Librefs" on page 7 and "Loading a SAS Data Set onto a CAS Server" on page 7 in Chapter 2, "Shared Concepts."

(View the complete code for this example at `https://github.com/sascommunities/doc-supplement-casstat/tree/main/Examples/regselectgs.sas`.)

The following example is closely modeled on the example in the section "Getting Started: GLMSELECT Procedure" in the *SAS/STAT User's Guide*.

The Sashelp.Baseball data set contains salary and performance information for Major League Baseball players who played at least one game in both the 1986 and 1987 seasons, excluding pitchers. The salaries (Time Inc. 1987) are from the 1987 season, and the performance measures are from 1986 (Collier Books 1987). The following step displays (in Figure 29.1) the variables in the data set:

```
proc contents varnum data=sashelp.baseball;
   ods select position;
run;
```

Suppose you want to investigate whether you can model the players' salaries from the 1987 season based on performance measures for the previous season. The aim is to obtain a parsimonious model that does not overfit these particular data, making the model useful for prediction. This example shows how you can use PROC REGSELECT as a starting point for such an analysis. Because the variation of salaries is much greater for the higher salaries, it is appropriate to apply a log transformation to the salaries before you do the model selection.

You can load the Sashelp.Baseball data set into your CAS session by specifying your CAS engine libref in the first statement in the following DATA step:

```
data mylib.baseball;
   set sashelp.baseball;
run;
```

These statements assume that your CAS engine libref is named mylib, as in the section "Using CAS Sessions and CAS Engine Librefs" on page 1479, but you can substitute any appropriately defined CAS engine libref.

Figure 29.1 Sashelp.Baseball Data Set

The CONTENTS Procedure

Variables in Creation Order				
#	Variable	Type	Len	Label
1	Name	Char	18	Player's Name
2	Team	Char	14	Team at the End of 1986
3	nAtBat	Num	8	Times at Bat in 1986
4	nHits	Num	8	Hits in 1986
5	nHome	Num	8	Home Runs in 1986
6	nRuns	Num	8	Runs in 1986
7	nRBI	Num	8	RBIs in 1986
8	nBB	Num	8	Walks in 1986
9	YrMajor	Num	8	Years in the Major Leagues
10	CrAtBat	Num	8	Career Times at Bat
11	CrHits	Num	8	Career Hits
12	CrHome	Num	8	Career Home Runs
13	CrRuns	Num	8	Career Runs
14	CrRbi	Num	8	Career RBIs
15	CrBB	Num	8	Career Walks
16	League	Char	8	League at the End of 1986
17	Division	Char	8	Division at the End of 1986
18	Position	Char	8	Position(s) in 1986
19	nOuts	Num	8	Put Outs in 1986
20	nAssts	Num	8	Assists in 1986
21	nError	Num	8	Errors in 1986
22	Salary	Num	8	1987 Salary in $ Thousands
23	Div	Char	16	League and Division
24	logSalary	Num	8	Log Salary

The following statements select a model by using the default settings for stepwise selection. ODS Graphics must be enabled before you can request plots. For more information about ODS Graphics, see the section "ODS Graphics" on page 1517.

```
ods graphics on;

proc regselect data=mylib.baseball;
   class league division;
   model logSalary = nAtBat nHits nHome nRuns nRBI nBB
                     yrMajor crAtBat crHits crHome crRuns crRbi
                     crBB league division nOuts nAssts nError;
   selection method=stepwise plots=all;
run;
```

The output and graphics from this analysis are presented in Figure 29.2 through Figure 29.6.

Figure 29.2 Selection Information, Number of Observations, Class Level Information, and Dimensions

The REGSELECT Procedure

Selection Information	
Selection Method	Stepwise
Select Criterion	SBC
Stop Criterion	SBC
Effect Hierarchy Enforced	None
Stop Horizon	3

Number of Observations Read	322
Number of Observations Used	263

Class Level Information

Class	Levels	Values
League	2	American National
Division	2	East West

Dimensions	
Number of Effects	19
Number of Parameters	21

The "Selection Information" table provides details about the method and criteria used to perform the model selection. The requested selection method is a variant of the traditional stepwise selection in which the decisions about what effects to add or drop at any step and when to terminate the selection are both based on the Schwarz Bayesian information criterion (SBC). The effect in the current model whose removal yields the maximal decrease in the SBC value is dropped, provided that this lowers the SBC value. When no further decrease in the SBC value can be obtained by dropping an effect in the model, the effect whose addition to the model yields the lowest SBC value is added and the whole process is repeated. The method terminates when dropping or adding any effect increases the SBC value.

Figure 29.2 displays the "Number of Observations," "Class Level Information," and "Dimensions" tables. The "Number of Observations" table shows that of the 322 observations in the input data, only 263 observations are used in the analysis because there are observations that contain incomplete data. The "Class Level Information" table lists the levels of the classification variables division and league. When you specify effects that contain classification variables, the number of parameters is usually larger than the number of effects. The "Dimensions" table shows the number of effects and the number of parameters that are considered.

The "Stepwise Selection Summary" table in Figure 29.3 shows the effect that was added or dropped at each step of the selection process together with fit statistics for the model at each step. In this case, both selection and stopping are based on the SBC.

Figure 29.4 displays the "Stop Reason," "Selection Reason," and "Selected Effects" tables. Note that these tables are displayed without any titles. The "Stop Reason" table indicates that selection stopped because adding or removing any effect would worsen the SBC value that is used as the selection criterion. In this case, because no CHOOSE= criterion is specified in the SELECTION statement, the final model is the selected model; this is indicated in the "Selection Reason" table. The "Selected Effects" table lists the effects in the selected model.

Figure 29.3 Selection Summary Table

The REGSELECT Procedure

Selection Details

Selection Summary

Step	Effect Entered	Effect Removed	Number Effects In	SBC
0	Intercept		1	-57.2041
1	CrRuns		2	-194.3166
2	nHits		3	-252.5794
3	YrMajor		4	-262.7322
4		CrRuns	3	-262.8353
5	nBB		4	-269.7804*

*** Optimal Value Of Criterion**

Figure 29.4 Stopping and Selection Reasons

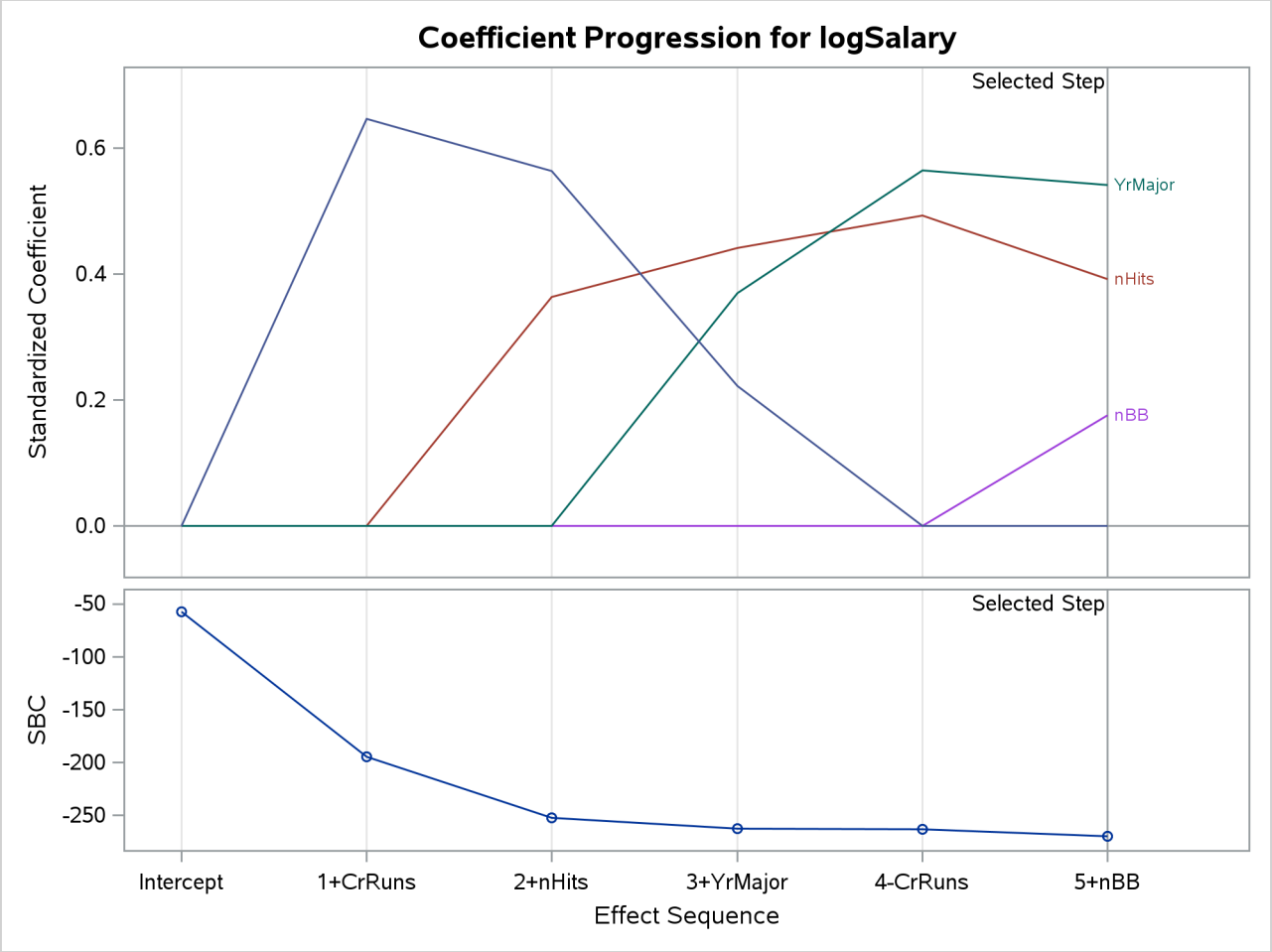
Stepwise selection stopped because adding or removing an effect does not improve the SBC criterion.

The model at step 5 is selected.

Selected Effects: Intercept nHits nBB YrMajor

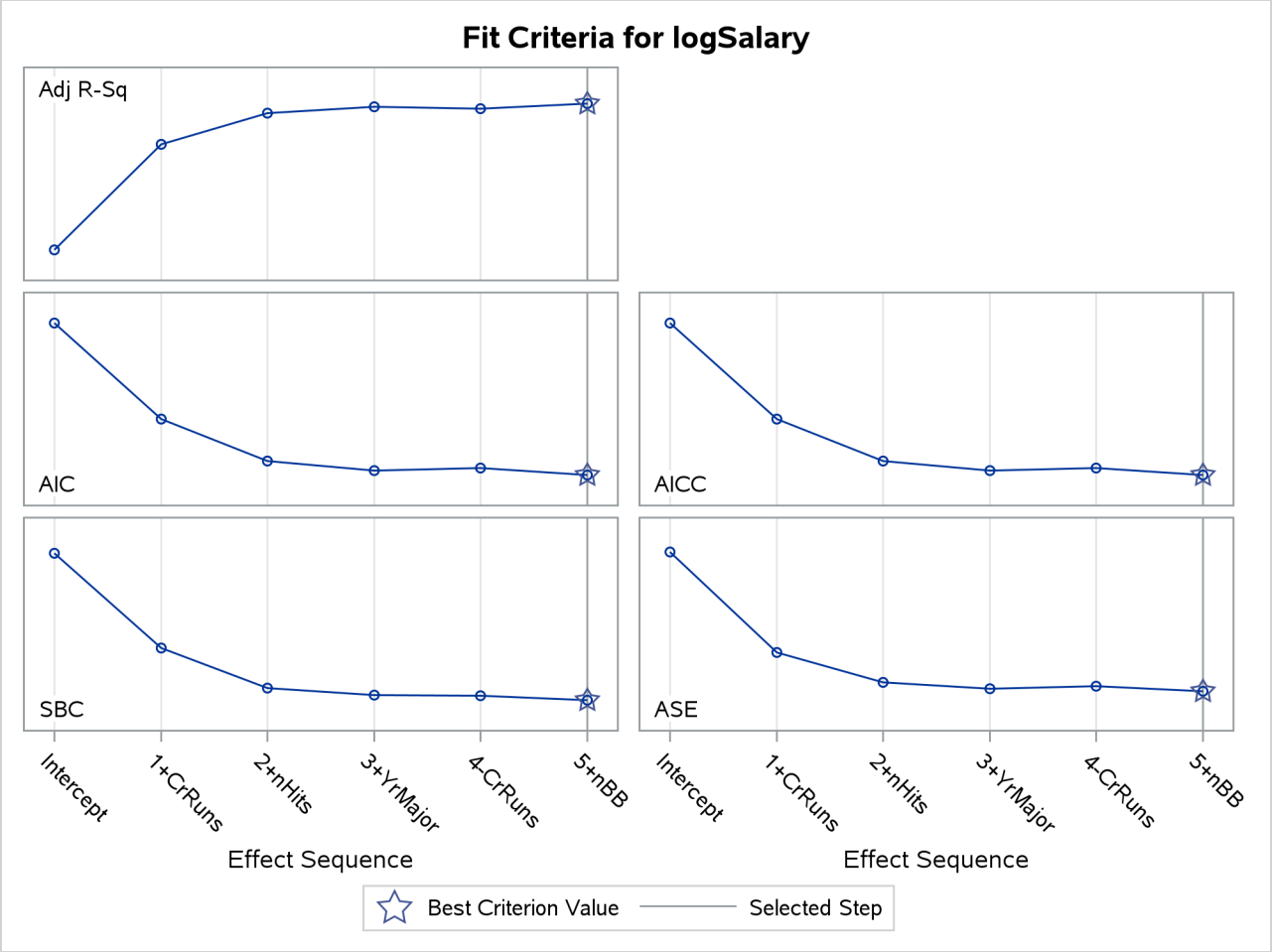
The coefficient panel in Figure 29.5 enables you to visualize the selection process. In this plot, standardized coefficients of all the effects that are selected at some step of the stepwise method are plotted as a function of the step number. This enables you to assess the relative importance of the effects that are selected at any step of the selection process and to know when effects entered the model. The lower plot in the panel shows how the criterion that is used to choose the selected model changes as effects enter or leave the model.

Figure 29.5 Coefficient Progression



The criterion panel in Figure 29.6 provides a graphical view of the progression of the fit criteria as the selection process evolves.

Figure 29.6 Criterion Panel



The "Analysis of Variance," "Fit Statistics," and "Parameter Estimates" tables shown in Figure 29.7 display details of the selected model.

Figure 29.7 Details of the Selected Model

The REGSELECT Procedure

Selected Model

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	3	120.52553	40.17518	120.12	<.0001
Error	259	86.62820	0.33447		
Corrected Total	262	207.15373			

Figure 29.7 *continued*

Root MSE	0.57834
R-Square	0.58182
Adj R-Sq	0.57697
AIC	-19.06903
AICC	-18.83557
SBC	-269.78041
ASE	0.32938

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	4.013911	0.111290	36.07	<.0001
nHits	1	0.007929	0.000994	7.98	<.0001
nBB	1	0.007280	0.002049	3.55	0.0005
YrMajor	1	0.100663	0.007551	13.33	<.0001

Finally, a table is displayed that shows the amount of time (in seconds) that PROC REGSELECT required to perform the different tasks in the analysis.

Figure 29.8 Procedure Timing

Task Timing

Task	Seconds	Percent
Setup and Parsing	0.03	48.84%
Levelization	0.02	32.62%
Model Initialization	0.00	3.51%
SSCP Computation	0.01	8.02%
Model Selection	0.00	3.65%
Cleanup	0.00	2.49%
Total	0.06	100.00%

You might want to examine regression diagnostics for the selected model to investigate whether collinearity among the selected parameters or the presence of outlying or high-leverage observations might be affecting the fit. The following statements include some options and statements to obtain these diagnostics:

```
proc regselect data=mylib.baseball;
  class league division;
  model logSalary = nAtBat nHits nHome nRuns nRBI nBB
                    yrMajor crAtBat crHits crHome crRuns crRbi
                    crBB league division nOuts nAssts nError / vif clb;
  selection method=stepwise;
  output out=mylib.baseballOut
         p=predictedLogSalary r h cookd rstudent copyvars=(name);
run;
```

The VIF and CLB options in the MODEL statement request variance inflation factors and 95% confidence limits, respectively, for the parameter estimates. Figure 29.9 shows the "Parameter Estimates" table, which displays these requested statistics. The variance inflation factors (VIF) measure the inflation in the variances of the parameter estimates due to collinearities that exist among the regressor (independent) variables.

Although there are no formal criteria for deciding whether a VIF is large enough to affect the predicted values, the VIF values for the selected effects in this example are small enough to indicate that there are no collinearity issues among the selected regressors.

Figure 29.9 Parameter Estimates with Additional Statistics

The REGSELECT Procedure

Selected Model

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Pr > |t|	Variance Inflation	95% Confidence Limits	
Intercept	1	4.013911	0.111290	36.07	<.0001	0	3.79476	4.23306
nHits	1	0.007929	0.000994	7.98	<.0001	1.49642	0.00597	0.00989
nBB	1	0.007280	0.002049	3.55	0.0005	1.52109	0.00325	0.01131
YrMajor	1	0.100663	0.007551	13.33	<.0001	1.02488	0.08579	0.11553

By default, SAS Viya statistical procedures do not include all variables from the input data table in output data tables. The COPYVARS= option in the OUTPUT statement specifies that the variable name in the input data table be added as an identification variable in the baseballOut data table that is produced by the OUTPUT statement. In addition to this variable, the OUTPUT statement requests that predicted values, raw residuals, leverage values, Cook's D statistics, and studentized residuals be added to the output data table. Note that default names are used for these statistics, except for the predicted values for which a specified name, predictedLogSalary, is supplied. The following statements use PROC PRINT to display five observations in this output data table:

```
proc print data=mylib.baseballOut(obs=5);
run;
```

Figure 29.10 First 5 Observations of the baseballOut Data Set

Obs	predictedLogSalary	Residual	COOKD	H	RSTUDENT	Name
1	4.73980	.	.	0.016087	.	Allanson, Andy
2	6.50852	-0.29392	.000730178	0.011060	-0.51031	Dawson, Andre
3	4.66148	-0.41299	.002516874	0.018999	-0.72031	Newman, Al
4	6.52518	0.47788	.003069150	0.017361	0.83308	Thornton, Andre
5	5.65468	0.65524	.002020832	0.006219	1.13715	Van Slyke, Andy

Syntax: REGSELECT Procedure

The following statements are available in the REGSELECT procedure:

PROC REGSELECT < *options* > **;**
BY *variables* **;**
CLASS *variable* < **(***options***)** >. . . < *variable* < **(***options***)** > > < / *global-options* > **;**
CODE < *options* > **;**
DISPLAY < *table-list* > < / *options* > **;**
DISPLAYOUT *table-spec-list* < / *options* > **;**
EFFECT *name***=***effect-type***(***variables* < / *options* >**)** **;**
FREQ *variable* **;**
MODEL *dependent* **=** < *effects* > < / *model-options* > **;**
OUTPUT OUT=*libref.data-table* < *keyword* < **=***name* > >. . . < *keyword* < **=***name* > > **;**
PARTITION *partition-options* **;**
SELECTION < **METHOD=***method* < **(***method-options***)** > >< *options* > **;**
STORE < **OUT=** >*libref.data-table* **;**
WEIGHT *variable* **;**

The PROC REGSELECT statement and a single MODEL statement are required. All other statements are optional. The CLASS statement can appear multiple times. If a CLASS statement is specified, it must precede the MODEL statement.

The rest of this section provides detailed syntax information about each of the preceding statements, beginning with the PROC REGSELECT statement. The remaining statements are described in alphabetical order.

PROC REGSELECT Statement

PROC REGSELECT < *options* > **;**

The PROC REGSELECT statement invokes the procedure. Table 29.1 summarizes the options in the PROC REGSELECT statement by function.

Table 29.1 PROC REGSELECT Statement Options

Option	Description
Basic Option	
DATA=	Specifies the input data set
Other Options	
ALPHA=	Sets the significance level to be used for the construction of confidence intervals
NOCLPRINT	Limits or suppresses the display of class levels
PLOTS=	Produces ODS graphical displays

You can specify the following *options*:

ALPHA=*number*

sets the significance level to be used for the construction of confidence intervals. The value must be between 0 and 1; the default value of 0.05 results in 95% intervals. This option affects the keywords LCL, LCLM, UCL, and UCLM in the OUTPUT statement and the CLB option in the MODEL statement.

DATA=*libref.data-table*

names the input data table for PROC REGSELECT to use. The default is the most recently created data table. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to the data, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 1479.

data-table specifies the name of the input data table.

NOCLPRINT< =*number* **>**

suppresses the display of the "Class Level Information" table if you do not specify *number*. If you specify *number*, the values of the classification variables are displayed for only those variables whose number of levels is less than *number*. Specifying *number* helps reduce the size of the "Class Level Information" table if some classification variables have a large number of levels.

PLOTS< (UNPACK) > = RIDGE | RIDGEPANEL | RIDGEPLOT < (*ridge-plot-options***) >**

creates panels of variance inflation factor (VIF) values and standardized ridge estimates by ridge values for each coefficient. The VIF values for each coefficient are connected by lines and are displayed in the upper plot in each panel. The points that correspond to the standardized estimates of each coefficient are connected by lines and are displayed in the lower plot in each panel.

You can specify the following *ridge-plot-options*:

COEF

plots the standardized ridge estimates by ridge values for each coefficient.

COMMONAXES

specifies that the same VIF axis and the same standardized estimate axis be used in all panels when multiple panels are needed. By default, these axes are chosen independently for the regressors shown in each panel.

MAXVARS=ALL | *number*

specifies the maximum number of regressors to display in each panel or in each plot if you also specify the UNPACK option. If you specify MAXVARS=ALL, then the VIF values and ridge traces for all regressors are displayed in a single panel. By default, MAXVARS=10.

RIDGEAXIS=LINEAR | LOG

specifies the axis type to use for displaying the ridge parameters. By default, RIDGEAXIS=LINEAR. Note that the point whose ridge parameter value is 0 is not displayed if you specify RIDGEAXIS=LOG.

UNPACK | UNPACKPANEL
suppresses paneling. The traces of the VIF statistics and standardized estimates are shown in separate plots.

VIF
plots the VIF statistics by ridge values for each coefficient.

VIFAXIS=LINEAR | LOG
specifies the axis type to use for displaying the VIF statistics. By default, VIFAXIS=LINEAR.

BY Statement

BY *variables* **;**

You can specify a BY statement in PROC REGSELECT to obtain separate analyses of observations in groups that are defined by the values of the BY variables. If you specify more than one BY statement, only the last one specified is used. For more information, see the discussion of BY-group processing in the "Grouping Data" section of *SAS Programmers Guide: Essentials*.

CLASS Statement

CLASS *variable* < **(***options***)** >. . . < *variable* < **(***options***)** > > < / *global-options* > **;**

The CLASS statement names the classification variables to be used as explanatory variables in the analysis. You can list the response variable for binary models in the CLASS statement, but this is not required. Table 29.2 summarizes the values that you can use for either an *option* or a *global-option*. The options are fully documented in the section "CLASS Statement" on page 9 in Chapter 2, "Shared Concepts."

Table 29.2 CLASS Statement Options

Option	Description
DESCENDING	Reverses the sort order
MISSING	Treats missing values as valid levels
ORDER=	Specifies the sort order for the levels
PARAM=	Specifies the parameterization of the variable
REF=	Specifies the reference level of the variable
SPLIT	Splits levels of CLASS variables into independent effects

CODE Statement

CODE < *options* > **;**

The CODE statement writes SAS DATA step code for computing predicted values of the fitted model to a file or to a table. To score new data, you can then include the file in a DATA step, or you can specify the table in the **runCodeTable** action in the **dataStep** action set (for more information, see *SAS Viya: System Programming Guide*).

Table 29.3 summarizes the *options* available in the CODE statement.

Table 29.3 CODE Statement Options

Option	Description
COMMENT	Adds comments to the generated code
FILE=	Names the file in which to save the generated code
FORMATWIDTH=	Specifies the numeric format width for the regression coefficients
INDENTSIZE=	Specifies the number of spaces to indent the generated code
LABELID=	Specifies a number used to construct names and labels
LINESIZE=	Specifies the line size for the generated code
NOTRIM	Compares formatted values, including blank padding
OUT=	Names an output table in which to save the generated code

For more information about the syntax of the CODE statement, see the section "CODE Statement" on page 13 in Chapter 2, "Shared Concepts."

This statement is ignored if you perform best-subset selection.

DISPLAY Statement

DISPLAY < *table-list* > < / *options* > **;**

The DISPLAY statement enables you to specify a list of display tables to display or exclude. This statement is similar to the ODS SELECT, ODS EXCLUDE, and ODS TRACE statements. However, the DISPLAY statement can improve performance when a large number of tables could be generated (such as in BY-group processing). The procedure processes the DISPLAY statement on a CAS server and thus sends only a subset of ODS tables to the SAS client. Because ODS statements are processed on a SAS client, first all the generated display tables are sent to the client, and then the client creates a subset.

If you use both DISPLAY and ODS statements together, the DISPLAY statement takes precedence over the ODS statements. Note that the ODS EXCLUDE statement processes tables that are sent to the client after they have been filtered by the DISPLAY statement. In some cases, it might appear that the ODS EXCLUDE statement is taking precedence because it can further filter the tables. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

You can specify the *table-list* as a list of table names, paths, partial pathnames, and regular expressions.

The table names that you can specify are listed in the section "ODS Table Names" on page 1515. A path is a table name that is prefixed with dot-separated grouping information. For example, a

SelectionSummary table that a procedure produces during a selection routine might have the path *Bygroup1.Summary.SelectionSummary*. A partial pathname does not include all groups; for example, *SelectionSummary* and *Summary.SelectionSummary* are partial pathnames for *Bygroup1.Summary.SelectionSummary*.

When you specify a table name or partial pathname, all display tables whose paths end in the specified name are selected for display or exclusion. For example, both *SelectionSummary* and *Summary.SelectionSummary* select *Bygroup1.Summary.SelectionSummary*.

A regular expression is enclosed in forward slashes (/). For example, specifying "/tions/" selects all pathnames that contain the substring "tions"; in particular, the *Bygroup1.Summary.SelectionSummary* table is selected. Specifying "!/tions/" selects all pathnames that do not contain the substring "tions"; in particular, the *Bygroup1.Summary.SelectionSummary* table is not selected.

You can specify the following *options* after a slash (/):

CASESENSITIVE
performs a case-sensitive comparison of table names in the *table-list* to display table names when tables are subsetted for display. To preserve case, you must enclose table names in the *table-list* in quotation marks.

EXCLUDE
displays all display tables except those that you specify in the *table-list*.

EXCLUDEALL
suppresses display of all tables. This option takes precedence over the other options.

TRACE
displays the display table names, labels, and paths.

DISPLAYOUT Statement

DISPLAYOUT *table-spec-list* < / *options* > **;**

The DISPLAYOUT statement enables you to create output tables from your displayed output. This statement is similar to the ODS OUTPUT statement. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

The *table-spec-list* specifies a list of output tables to create. Each entry in the list has either a *key=value* format or a *key* format:

key=value specifies *key* as the ODS table name, path, or partial pathname, and specifies *value* as the output table name.

key specifies *key* as the ODS table name and also as the output table name.

The ODS table names that you can specify are listed in the section "ODS Table Names" on page 1515. You cannot specify the ODS table named OutputCasTables in the *table-spec-list*.

Table names and partial pathnames are discussed under the DISPLAY statement. The DISPLAYOUT statement does not support regular expressions.

You can specify the following *options* after a slash (/):

INCLUDEALL
creates output tables for all display tables. The name of the created output table is the same as the name of the corresponding display table. If you specify this option, the *table-spec-list* specification is ignored.

NOREPLACE
does not replace any existing output table of the same name.

REPEATED
replicates all output tables on all nodes.

EFFECT Statement

EFFECT *name*=*effect-type* **(***variables* < / *options* >**) ;**

The EFFECT statement enables you to construct special collections of columns for design matrices. These collections are referred to as *constructed effects* to distinguish them from the usual model effects that are formed from continuous or classification variables, as discussed in the section "GLM Parameterization of Classification Variables and Effects" on page 67 in Chapter 2, "Shared Concepts."

You can specify the following *effect-types*:

COLLECTION specifies a collection effect that defines one or more variables as a single effect that has multiple degrees of freedom. The variables in a collection are considered as a unit for purposes of estimation and inference.

MULTIMEMBER | MM specifies a multimember classification effect whose levels are determined by one or more variables that appear in a CLASS statement.

POLYNOMIAL | POLY specifies a multivariate polynomial effect in the specified numeric variables.

SPLINE specifies a regression spline effect whose columns are univariate spline expansions of one or more variables. A spline expansion replaces the original variable with an expanded or larger set of new variables.

Table 29.4 summarizes the *options* available in the EFFECT statement.

Table 29.4 EFFECT Statement Options

Option	Description
Collection Effects Options	
DETAILS	Displays the constituents of the collection effect

Table 29.4 *continued*

Option	Description
Multimember Effects Options	
DETAILS	Displays the levels of the multimember effect
NOEFFECT	Specifies that observations whose levels are all missing for the multimember variables should have 0 values in the corresponding design matrix columns
STDIZE	Standardizes the design matrix entries so that each observation has a sum of 1
WEIGHT=	Specifies the weight variable for the contributions of each classification effect
Polynomial Effects Options	
DEGREE=	Specifies the degree of the polynomial
DETAILS	Displays details of the specified polynomial
MDEGREE=	Specifies the maximum degree of any variable in a term of the polynomial
NOSEPARATE	Treats the polynomial as a single effect with multiple degrees of freedom
STANDARDIZE=	Specifies centering and scaling suboptions for the variables that define the polynomial
Spline Effects Options	
BASIS=	Specifies the type of basis (B-spline basis or truncated power function basis) for the spline effect
DATABOUNDARY	Uses the extremes of the data as boundary knots for a B-spline basis
DEGREE=	Specifies the degree of the spline effect
DETAILS	Displays the knots and locations for each spline basis function
KNOTMAX=	Requests equally spaced right-side boundary knots starting at the variables' maximum and ending at the KNOTMAX= value
KNOTMETHOD=	Specifies how to construct the knots for the spline effect
KNOTMIN=	Requests equally spaced left-side boundary knots starting at the KNOTMIN= value and ending at the variables' minimum value
NATURALCUBIC	Specifies a natural cubic spline basis for the spline effect
SEPARATE	Treats the spline basis for each variable as a separate effect when multiple variables are specified
SPLIT	Treats each design matrix column as a separate effect for selection methods

For more information about the syntax of these *effect-types* and how columns of constructed effects are computed, see the section "EFFECT Statement" on page 18 in Chapter 2, "Shared Concepts."

FREQ Statement

FREQ *variable* **;**

The *variable* in the FREQ statement identifies a numeric variable in the input data table that contains the frequency of occurrence of each observation. PROC REGSELECT treats each observation as if it appears f times, where f is the value of the FREQ *variable* for the observation. If f is not an integer, it is truncated to an integer. If f is less than 1 or missing, the observation is not used in the analysis. When you omit the FREQ statement, each observation is assigned a frequency of 1.

MODEL Statement

MODEL *dependent*=< *effects* > / < *model-options* > **;**

The MODEL statement names the dependent variable and the explanatory effects, including covariates, main effects, interactions, and nested effects. If you omit the explanatory effects, the procedure fits an intercept-only model.

After the keyword MODEL, the dependent (response) variable is specified, followed by an equal sign. The explanatory effects follow the equal sign.

For information about constructing the model effects, see the section "Specification and Parameterization of Model Effects" on page 64 in Chapter 2, "Shared Concepts."

The *model-options* control other aspects of model formation and inference. Table 29.5 summarizes these options.

Table 29.5 MODEL Statement Options

Option	Description
Model Options	
CLB	Requests confidence limits
INCLUDE=	Includes effects in all models for model selection
INFORMATIVE	Models missing values by using extra indicator variables
NOINT	Suppresses the intercept
RIDGE=	Performs ridge regression analysis
SS3	Displays the Type 3 or joint tests of effects
START=	Includes effects in the initial model for model selection
STB	Displays standardized estimates
TOL	Displays tolerance values for the estimates
VIF	Displays variance inflation factors for the estimates
XPX	Displays sums-of-squares and crossproducts matrix

Model Options

You can specify the following *options* in the MODEL statement after a slash (/):

CLB
requests the $100(1-\alpha)\%$ upper and lower confidence limits for the parameter estimates. By default, the 95% limits are computed; you can use the ALPHA= option in the PROC REGSELECT statement to change the α level. The CLB option is not available when you specify METHOD=LAR, METHOD=LASSO, METHOD=ELASTICNET, or METHOD=BESTSUBSET in the SELECTION statement.

INCLUDE=*n*

INCLUDE=*single-effect*

INCLUDE=(*effects***)**
forces effects to be included in all models. If you specify INCLUDE=*n*, then the first *n* effects that are listed in the MODEL statement are included in all models. If you specify INCLUDE=*single-effect* or INCLUDE=(*single-effect*), then the specified effects are forced into all models. The INCLUDE= option is not available when you specify METHOD=LAR, METHOD=LASSO, or METHOD=ELASTICNET in the SELECTION statement. For METHOD=BESTSUBSET, only the first *n* effects that are listed in the MODEL statement are allowed.

INFORMATIVE
models missing values by using extra model effects. These effects consist of dummy variables that take the value 1 when the value of a continuous model variable involved in the effect is missing, and take the value 0 otherwise. The missing value in the original model effect is replaced by the average value of the effect for the nonmissing values. For continuous-by-class effects, such as A*x, where A is a classification variable and x is a continuous variable, informative missingness creates multiple dummy columns and substitutes the effect mean of x that corresponds to the respective level of A. Missing values for classification variables are treated as valid levels. For more information about informative missingness, see the section "Informative Missingness" on page 99 in Chapter 2, "Shared Concepts."

NOINT
suppresses the intercept term that is otherwise included in the model.

RIDGE=*value(s)*
performs a ridge regression analysis and specifies the values of the ridge constant *k*. Each value of *k* produces a set of ridge regression estimates that are displayed in the RidgeParmEst table and a set of ridge fit statistics that are displayed in the RidgeFitStats table. You can specify only nonnegative numbers in this option. Ridge regression is ignored when you perform model selection.

SS3
displays a "Model Anova" table that contains tests computed with Type III sums of squares for each effect. For more information, see the section "Joint Tests and Type 3 Tests" on page 1509. This option is not available when you specify METHOD=BESTSUBSET in the SELECTION statement.

START=*n*

START=*single-effect*

START=(*effects***)**

is used to begin the selection process in the FORWARD, FORWARDSWAP, and STEPWISE selection methods from the initial model that you designate. If you specify START=*n*, then the starting model consists of the first *n* effects that are listed in the MODEL statement. If you specify START=*single-effect* or START=(*single-effect*), then the starting model consists of these specified effects. The START= option is not available when you specify METHOD=BACKWARD, METHOD=LAR, METHOD=LASSO, METHOD=ELASTICNET, or METHOD=BESTSUBSET in the SELECTION statement.

STB

produces standardized regression coefficients. A standardized regression coefficient is computed by dividing a parameter estimate by the ratio of the sample standard deviation of the dependent variable to the sample standard deviation of the regressor. This option is not available when you specify METHOD=BESTSUBSET in the SELECTION statement.

TOL

produces tolerance values for the estimates. Tolerance for a parameter is defined as $1 - R^2$, where R^2 is obtained from the regression of the parameter on all other parameters in the model. This option is not available when you specify METHOD=LAR, METHOD=LASSO, METHOD=ELASTICNET, or METHOD=BESTSUBSET in the SELECTION statement.

VIF

produces variance inflation factors along with the parameter estimates. Variance inflation is the reciprocal of tolerance. This option is not available when you specify METHOD=LAR, METHOD=LASSO, METHOD=ELASTICNET, or METHOD=BESTSUBSET in the SELECTION statement.

XPX

displays the $X'X$ crossproducts matrix for the model. The crossproducts matrix is bordered by the $X'Y$ and $Y'Y$ matrices.

OUTPUT Statement

OUTPUT OUT=*libref.data-table*
< **COPYVARS=***(variables)* >
< *keyword* < *=name* > >. . . < *keyword* < *=name* > > **;**

The OUTPUT statement creates a data table that contains observationwise statistics, which are computed after the model is fitted. The variables in the input data table are *not* included in the output data table, in order to avoid data duplication for large data tables; however, variables that you specify in the COPYVARS= option are included.

The output statistics are computed based on the parameter estimates for the selected model. For observations in which only the response variable is missing, predicted values are computed even though these observations do not affect the model fit. This enables, for example, predicted values to be computed for new observations.

You must specify the following option:

OUT=*libref.data-table*
names the output data table for PROC REGSELECT to use. You must specify this option before any other options. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to where the data table is to be stored, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 1479.

data-table specifies the name of the output data table.

You can also specify the following syntax elements:

COPYVAR=*variable*

COPYVARS=(*variables***)**
transfers one or more *variables* from the input data table to the output data table.

keyword **<** **=***name* **>**
specifies the statistics to include in the output data table and optionally names the new variables that contain the statistics. Specify a keyword for each desired statistic (see the following list of keywords), followed optionally by an equal sign and a variable to contain the statistic.

If you specify *keyword=name*, the new variable that contains the requested statistic has the specified name. If you omit the optional *=name* after a *keyword*, then a default name is used.

You can specify the following values for *keyword* to request statistics that are available with all selection methods:

PREDICTED
PRED
P
requests predicted values for the response variable. The default is Pred.

RESIDUAL
RESID
R
requests the residual, calculated as ACTUAL – PREDICTED. The default is Residual.

ROLE
requests a numeric variable that indicates the role played by each observation in fitting the model. The default is _ROLE_. For each observation, the interpretation of this variable is shown in Table 29.6.

Table 29.6 Role Interpretation

Value	Observation Role
0	Not used
1	Training
2	Validation
3	Testing

If you do not partition the input data by using a PARTITION statement, then the role variable value is 1 for observations used in fitting the model, and 0 for observations that have at least one missing or invalid value for the response, regressor, frequency, or weight variables.

In addition to the preceding statistics, you can also use the *keywords* listed in Table 29.7 in the OUTPUT statement to obtain additional statistics. These statistics are not available if you specify METHOD=LAR or METHOD=LASSO or METHOD=ELASTICNET in the SELECTION statement. For computational formulas, see the section "Diagnostic Statistics" on page 1505. All the statistics available in the OUTPUT statement are conditional on the selected model and do not take into account the variability introduced when you do model selection.

Table 29.7 Keywords for OUTPUT Statement

Keyword	Description
COOKD	Cook's D influence statistic
COVRATIO	Standard influence of observation on covariance of betas
DFFIT	Standard influence of observation on predicted value
H	Leverage, $\mathbf{x}_i(\mathbf{X}'\mathbf{X})^{-}\mathbf{x}_i'$
LCL	Lower bound of a $100(1-\alpha)\%$ confidence interval for an individual prediction. This includes the variance of the error and the variance of the parameter estimates.
LCLM	Lower bound of a $100(1-\alpha)\%$ confidence interval for the expected value (mean) of the dependent variable
LIKEDIST	Likelihood displacement, which measures the change in the error sum of squares when the current observation is deleted
PRESS	ith residual divided by $(1-h)$, where h is the leverage and where the model has been refit without the ith observation
RSTUDENT	A studentized residual with the current observation deleted
STDI	Standard error of the individual predicted value
STDP	Standard error of the mean predicted value
STDR	Standard error of the residual
STUDENT	Studentized residuals, which are the residuals divided by their standard errors
UCL	Upper bound of a $100(1-\alpha)\%$ confidence interval for an individual prediction
UCLM	Upper bound of a $100(1-\alpha)\%$ confidence interval for the expected value (mean) of the dependent variable

This statement is ignored if you perform best-subset selection.

PARTITION Statement

PARTITION *partition-option* **;**

The PARTITION statement specifies how observations in the input data set are logically partitioned into disjoint subsets for model training, validation, and testing. For more information, see the section "Using Validation and Test Data" on page 101 in Chapter 2, "Shared Concepts." Either you can designate a variable in the input data table and a set of formatted values of that variable to determine the role of each observation, or you can specify proportions to use for randomly assigning observations to each role.

You must specify exactly one of the following *partition-options*:

FRACTION(<TEST=*fraction***> <VALIDATE=***fraction***> <SEED=***number***>)**
randomly assigns specified proportions of the observations in the input data table to the roles. You specify the proportions for testing and validation by using the TEST= and VALIDATE= suboptions. If you specify both the TEST= and VALIDATE= suboptions, then the sum of the specified fractions must be less than 1 and the remaining fraction is the proportion of the observations assigned to the training role. The SEED= option specifies an integer that is used to start the pseudorandom number generator for random partitioning of data for training, testing, and validation. If you do not specify SEED=*number* or if *number* is less than or equal to 0, the seed is generated by reading the time of day from the computer's clock.

ROLE=*variable* **(<TEST='***value***'> <TRAIN='***value***'> <VALIDATE='***value***'>)**
ROLEVAR=*variable* **(<TEST='***value***'> <TRAIN='***value***'> <VALIDATE='***value***'>)**
names the *variable* in the input data table whose values are used to assign roles to each observation. This *variable* cannot also appear as an analysis variable in other statements or options. The TEST=, TRAIN=, and VALIDATE= suboptions specify the formatted values of this variable that are used to assign observation roles. If you do not specify the TRAIN= suboption, then all observations whose role is not determined by the TEST= or VALIDATE= suboption are assigned to the training role.

SELECTION Statement

SELECTION <**METHOD=***method* <**(***method-options***)**> >< *options* > **;**

The SELECTION statement performs model selection by examining whether effects should be added to or removed from the model according to rules that are defined by model selection methods. The statement is fully documented in the section "SELECTION Statement" on page 42 in Chapter 2, "Shared Concepts."

The REGSELECT procedure supports the following values of the METHOD= option in the SELECTION statement:

BACKWARD specifies the backward elimination method, which starts with all effects in the model and deletes effects.

BESTSUBSET specifies the best-subset selection method, which uses the branch-and-bound algorithm (Furnival and Wilson 1974) to efficiently search candidate models without examining all possible subsets.

ELASTICNET specifies the elastic net method, which is an extension of LASSO that estimates parameters by using a version of ordinary least squares in which both the sum of the absolute regression coefficients and the sum of the squared regression coefficients are constrained.

FORWARD specifies the forward selection method, which starts with no effects in the model and adds effects.

FORWARDSWAP specifies forward-swap selection, which is an extension of the forward selection method. Before any addition step, PROC REGSELECT makes all pairwise swaps of one effect in the model and one effect out of the current model that improve the selection criterion. When the selection criterion is R-square, this method is the same as the MAXR method in the REG procedure in SAS/STAT software.

LAR specifies the least angle regression method. Like forward selection, this method starts with no effects in the model and adds effects. The parameter estimates at any step are "shrunk" when compared to the corresponding least squares estimates. If the model contains classification variables, then these classification variables are split. For more information about split classification variables, see the section "Classification Variables and the SPLIT Option" on page 1506.

LASSO specifies the LASSO method, which adds and deletes parameters based on a version of ordinary least squares in which the sum of the absolute regression coefficients is constrained. If the model contains classification variables, then these classification variables are split. For more information about split classification variables, see the section "Classification Variables and the SPLIT Option" on page 1506.

MCP specifies the MCP method (Zhang 2010), which uses the minimax concave penalty as the penalty function in penalized least squares estimation.

NONE specifies no model selection.

SCAD specifies the SCAD method (Fan and Li 2001), which uses smoothly clipped absolute deviation as the penalty function in penalized least squares estimation.

STEPWISE specifies the stepwise regression method, which is similar to the forward selection method except that effects already in the model do not necessarily stay there. ??citation??

By default, METHOD=STEPWISE.

The SELECT=, CHOOSE=, and STOP= *method-options* default to the Schwarz Bayesian information criterion (SBC) except in best-subset selection. For the best-subset selection method, the default SELECT= criterion is ADJRSQ. These defaults differ from their corresponding defaults in the HPLOGISTIC procedure in SAS/STAT software.

The DETAILS=ALL and DETAILS=STEPS options produce the "ANOVA," "Fit Statistics," and "Parameter Estimates" tables, which provide information about the model that is selected at each step of the selection process.

If you specify the PLOTS=CRITERIA or PLOTS=ALL option, then a plot of the fit criteria by the selection step is created. The default fit criteria that are shown in this plot are the ADJRSQ, AIC, AICC, ASE, and SBC statistics. If you also specify a PARTITION statement, then the ASE statistics for all roles that are defined in the PARTITION statement are included. If you use the CP or PRESS statistic as the SELECT=, STOP=, or CHOOSE= criterion in the SELECTION statement, then these statistics are also added to the criterion panel.

If you specify the PLOTS=FITBYROLE or PLOTS=ALL option and a PARTITION statement, then a plot of the ASE by the selection step for each role is created.

For best-subset selection, only a plot of the fit criteria is created if you specify the PLOTS=CRITERIA or PLOTS=ALL option. The fit criteria that are shown in the plot are the ADJRSQ, AIC, BIC, CP, RSQUARE, and SBC statistics.

For SCAD and MCP selection, no plot is produced in this release.

STORE Statement

STORE < **OUT=** >*libref.data-table* **;**

The STORE statement saves the context and results of the statistical analysis. The resulting analytic store has a binary file format that cannot be modified. An example of analytic store technology is to perform a time-consuming analysis and to store its results by using the STORE statement. Later you can perform specific statistical analysis tasks based on the saved results of the previous analysis, without having to fit the model again. The contents of the analytic store that PROC REGSELECT produces can be processed by the ASTORE procedure, the **`glmScore`** action in the **`regression`** action set, and the **`score`** action in the **`aStore`** action set.

libref.data-table is a two-level name, where *libref* refers to the library, and *data-table* specifies the name of the output data table. For more information about this two-level name, see the section "Using CAS Sessions and CAS Engine Librefs" on page 1479.

This statement is ignored if you perform best-subset selection.

WEIGHT Statement

WEIGHT *variable* **;**

The *variable* in the WEIGHT statement is used as a weight to perform a weighted analysis of the data. Observations that have nonpositive or missing weights are not included in the analysis. The values of the WEIGHT variable can be nonintegral and are not truncated. When you omit the WEIGHT statement, all observations that are used in the analysis are assigned a weight of 1.

Details: REGSELECT Procedure

Criteria Used in Model Selection

The REGSELECT procedure supports a variety of fit statistics that you can specify as criteria for the CHOOSE=, SELECT=, and STOP= options in the SELECTION statement, *method-options*. The following statistics are available:

ADJRSQ	Adjusted R-square statistic (Darlington 1968; Judge et al. 1985)
AIC	Akaike's information criterion (Akaike 1969; Judge et al. 1985)
AICC	Corrected Akaike's information criterion (Hurvich and Tsai 1989)
BIC | SBC	Schwarz Bayesian information criterion (Schwarz 1978; Judge et al. 1985)
CP	Mallows' C_p statistic (Mallows 1973; Hocking 1976)
PRESS	Predicted residual sum of squares statistic
RSQUARE	R-square statistic (Darlington 1968; Judge et al. 1985)
SL	Significance used to assess the contribution of an effect to the fit when it is added to or removed from a model
VALIDATE	Average square error over the validation data

NOTE: The RSQUARE statistic is supported only in the SELECT= option.

When you use SL as a criterion for effect selection, the definition depends on whether an effect is being considered as a drop candidate or an add candidate. Assume that the current model has p parameters, excluding the intercept. If you denote its residual sum of squares by RSS_p, add an effect with k degrees of freedom, and denote the residual sum of squares of the resulting model by RSS_{p+k}, then the F statistic for entry with k numerator degrees of freedom and $n-(p+k)-1$ denominator degrees of freedom is given by

$$F = \frac{(\mathrm{RSS}_p - \mathrm{RSS}_{p+k})/k}{\mathrm{RSS}_{p+k}/(n-(p+k)-1)}$$

where n is number of observations used in the analysis. The significance level for adding an effect is the p-value of this F statistic, and the effect is deemed significant if it is smaller than the SLENTRY limit. (For more information about the SLENTRY= option, see the section "SELECTION Statement" on page 42 in Chapter 2, "Shared Concepts.") Among several such add candidates, the effect with the smallest p-value (most significant) is deemed best.

If you drop an effect that has k degrees of freedom and denote the residual sum of squares of the resulting model by RSS_{p-k}, then the F statistic for removal with k numerator degrees of freedom and $n-p-k$ denominator degrees of freedom is given by

$$F = \frac{(\mathrm{RSS}_{p-k} - \mathrm{RSS}_p)/k}{\mathrm{RSS}_p/(n-p-k)}$$

where n is number of observations used in the analysis. The significance level for removal of an effect is the p-value of this F statistic, and the effect is deemed not significant if this p-value is larger than the

SLSTAY limit. (For more information about the SLSTAY= option, see the section "SELECTION Statement" on page 42 in Chapter 2, "Shared Concepts.") Among several such removal candidates, the effect with the largest p-value (least significant) is deemed the best removal candidate.

It is known that the "F-to-enter" and "F-to-delete" statistics do not follow an F distribution (Draper, Guttman, and Kanemasu 1971). Hence the SLENTRY and SLSTAY values cannot reliably be viewed as probabilities. One way to address this difficulty is to replace hypothesis testing as a means of selecting a model with information criteria or out-of-sample prediction criteria. Although Harrell (2001) points out that information criteria were developed for comparing only prespecified models, Burnham and Anderson (2002) note that AIC criteria have routinely been used for several decades to perform model selection in time series analysis.

Table 29.8 provides formulas and definitions for these fit statistics.

Table 29.8 Formulas and Definitions for Model Fit Summary Statistics

Statistic	Definition or Formula
n	Number of observations
p	Number of parameters, including the intercept
$\hat{\sigma}^2$	Estimate of pure error variance from fitting the full model
SST	Total sum of squares corrected for the mean for the dependent variable
SSE	Error sum of squares
ASE	$\frac{\text{SSE}}{n}$
MSE	$\frac{\text{SSE}}{n-p}$
R^2	$1-\frac{\text{SSE}}{\text{SST}}$
ADJRSQ	$1-\frac{(n-1)(1-R^2)}{n-p}$
AIC	$n\log\left(\frac{\text{SSE}}{n}\right)+2p+n+2$
AICC	$n\log\left(\frac{\text{SSE}}{n}\right)+\frac{n(n+p)}{n-p-2}$
CP (C_p)	$\frac{\text{SSE}}{\hat{\sigma}^2}+2p-n$
PRESS	$\sum_{i=1}^{n}\frac{r_i^2}{(1-h_i)^2}$ where r_i = residual at observation i and h_i = leverage of observation $i = \mathbf{x}_i(\mathbf{X}'\mathbf{X})^{-}\mathbf{x}_i'$
RMSE	$\sqrt{\text{MSE}}$
SBC	$n\ln\left(\frac{\text{SSE}}{n}\right)+p\ln(n)$

Diagnostic Statistics

This section gathers the formulas for the statistics available in the OUTPUT statement. All the statistics available in the OUTPUT statement are conditional on the selected model and do not take into account the variability introduced by doing model selection.

The model to be fit is $\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$, and the parameter estimate is denoted by $\mathbf{b} = (\mathbf{X}'\mathbf{X})^{-}\mathbf{X}'\mathbf{Y}$. The subscript i denotes values for the ith observation, and the parenthetical subscript (i) means that the statistic is computed by using all observations except the ith observation.

The ALPHA= option in the PROC REGSELECT statement is used to set the α value for the confidence limit statistics.

Table 29.9 contains the diagnostic statistics and their formulas. Each statistic is computed for each observation.

Table 29.9 Formulas and Definitions for Diagnostic Statistics

MODEL Option or Statistic	Formula
PRED ($\widehat{\mathbf{Y}}_i$)	$\mathbf{X}_i\mathbf{b}$
RES (r_i)	$\mathbf{Y}_i - \widehat{\mathbf{Y}}_i$
H (h_i)	$\mathbf{x}_i(\mathbf{X}'\mathbf{X})^{-}\mathbf{x}_i'$
STDP	$\sqrt{h_i\widehat{\sigma}^2}$
STDI	$\sqrt{(1+h_i)\widehat{\sigma}^2}$
STDR	$\sqrt{(1-h_i)\widehat{\sigma}^2}$
LCL	$\widehat{Y}_i - t_{\frac{\alpha}{2}}\text{STDI}$
LCLM	$\widehat{Y}_i - t_{\frac{\alpha}{2}}\text{STDP}$
UCL	$\widehat{Y}_i + t_{\frac{\alpha}{2}}\text{STDI}$
UCLM	$\widehat{Y}_i + t_{\frac{\alpha}{2}}\text{STDP}$
STUDENT	$\dfrac{r_i}{\text{STDR}_i}$
RSTUDENT	$\dfrac{r_i}{\hat{\sigma}_{(i)}\sqrt{1-h_i}}$
COOKD	$\dfrac{1}{p}\text{STUDENT}^2\dfrac{\text{STDP}^2}{\text{STDR}^2}$
COVRATIO	$\dfrac{\det(\hat{\sigma}^2_{(i)}(\mathbf{x}'_{(i)}\mathbf{x}_{(i)})^{-1}}{\det(\hat{\sigma}^2(\mathbf{X}'\mathbf{X})^{-1})}$
DFFITS	$\dfrac{(\widehat{\mathbf{Y}}_i - \widehat{\mathbf{Y}}_{(i)})}{(\hat{\sigma}_{(i)}\sqrt{h_i})}$
PRESS(predr$_i$)	$\dfrac{r_i}{1-h_i}$
LIKEDIST	$(n-p)\log\left(\dfrac{\hat{\sigma}^2_{(i)}}{\text{MSE}}\right) + \dfrac{r_i^2}{(1-h_i)^2\hat{\sigma}^2_{(i)}} - 1$

Classification Variables and the SPLIT Option

(View the complete code for this example (regselectdt.sas) in the example repository.)

PROC REGSELECT supports the ability to split classification variables when you are doing model selection. You use the SPLIT option in the CLASS statement to specify that the columns of the design matrix that correspond to effects that contain a split classification variable can enter or leave a model independently of the other design columns of that effect. The following statements illustrate the use of the SPLIT option:

```
data mylib.splitExample;
   length c2 $6;
   drop i;
   do i=1 to 1000;
     c1 = 1 + mod(i,6);
     if      i < 200 then c2 = 'low';
     else if i < 500 then c2 = 'medium';
     else                 c2 = 'high';
     x1 = ranuni(1);
     x2 = ranuni(1);
     y = x1+3*(c2 ='low')  + 10*(c1=3) +5*(c1=5) + rannor(1);
     output;
   end;
run;

proc regselect data=mylib.splitExample;
   class c1(split) c2(order=freq);
   model y = c1 c2 x1 x2;
   selection method=forward;
run;
```

The "Class Level Information" table shown in Figure 29.11 is produced by default whenever you specify a CLASS statement.

Figure 29.11 Class Levels

The REGSELECT Procedure

Class Level Information		
Class	**Levels**	**Values**
c1	6	1 2 3 4 5 6
c2	3	high medium low

The SPLIT option has been specified for the classification variable c1. This permits the parameters associated with the effect c1 to enter or leave the model individually. The "Parameter Estimates" table in Figure 29.12 shows that for this example the parameters that correspond only to levels 3 and 5 of c1 are in the selected model.

Figure 29.12 Parameter Estimates

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	2.827805	0.094738	29.85	<.0001
c1_3	1	10.161378	0.087606	115.99	<.0001
c1_5	1	5.018482	0.087593	57.29	<.0001
c2 high	1	-3.134740	0.084808	-36.96	<.0001
c2 medium	1	-2.923335	0.092537	-31.59	<.0001
c2 low	0	0	.	.	.
x1	1	1.315202	0.109440	12.02	<.0001

Using Validation and Test Data

When you have sufficient data, you can subdivide your data into three parts called the training, validation, and test data. The selection process fits models to the training data and uses the validation data to find the prediction errors for the models that are obtained in this way. This prediction error on the validation data can be used to decide when to terminate the selection process or to decide what effects to include as the selection process proceeds. Finally, after a selected model has been obtained, the test data can be used to assess how the selected model generalizes on data that played no role in selecting the model.

In some cases you might want to use only training and test data. For example, you might want to use an information criterion to decide what effects to include and when to terminate the selection process. In this case no validation data are required, but test data can still help you assess the predictive performance of the selected model. In other cases you might decide to use validation data during the selection process but forgo assessing the selected model on test data. Hastie, Tibshirani, and Friedman (2001) note that it is difficult to give a general rule for how many observations you should assign to each role. They state that a typical split might be 50% for training and 25% each for validation and testing.

You use a PARTITION statement to logically subdivide the DATA= data table into separate roles. You can name the fractions of the data that you want to reserve as test data and validation data. For example, the following statements randomly subdivide the inData data table, reserving 50% for training and 25% each for validation and testing:

```
proc regselect data=mylib.inData;
  partition fraction(test=0.25 validate=0.25);
  ...
run;
```

In some cases you might need to exercise more control over the partitioning of the input data table. You can do this by naming both a variable in the input data table and a formatted value of that variable that correspond to each role. For example, the following statements assign roles to the observations in the inData data table based on the value of the variable group in that data table. Observations in which the value of group is "group 1" are assigned to testing, and those whose value is "group 2" are assigned to training. All other observations are ignored.

```
proc regselect data=mylib.inData;
  partition roleVar=group(test='group 1' train='group 2')
  ...
run;
```

After you reserve observations for training, validation, and testing, a model fit of the training data is scored on the validation and test data, and the average squared error (ASE) is computed separately for each of these subsets. The ASE for each data role is the error sum of squares for observations in that role divided by the number of observations in that role.

Using the Validation ASE as the STOP= Criterion

If you have provided observations for validation, then you can specify STOP=VALIDATE as a suboption of the METHOD= option in the SELECTION statement. At step k of the selection process, the best candidate effect to enter or leave the current model is determined. Here, "best candidate" means the effect that gives the best value of the SELECT= criterion; this criterion does not need to be based on the validation data. The validation ASE for the model with this candidate effect added or removed is computed. If this validation ASE is greater than the validation ASE for the model at step k, then the selection process terminates at step k.

Using the Validation ASE as the CHOOSE= Criterion

When you specify the CHOOSE=VALIDATE suboption of the METHOD= option in the SELECTION statement, the validation ASE is computed for the models at each step of the selection process. The smallest model at any step that yields the smallest validation ASE is selected.

Computational Method

Multithreading

The REGSELECT procedure allocates data to different threads and calculates crossproduct matrices by accumulating the contributions from all threads. PROC REGSELECT also uses multiple threads to compute matrix inverses and select candidates during model selection. For more information about how PROC REGSELECT uses threads, see the section "Multithreading" on page 102 in Chapter 2, "Shared Concepts."

Ridge Regression

Ridge regression is a technique for analyzing multiple regression data that suffer from multicollinearity. In ridge regression analysis, the crossproduct matrix for the independent variables is centered (the NOINT option is ignored if it is specified) and scaled to one on the diagonal elements. The ridge constant k (which you specify in the RIDGE= option) is then added to each diagonal element of the crossproduct matrix. The ridge regression estimates are the least squares estimates that are obtained by using the new crossproduct matrix. More specifically, let $\mathbf{X} = (x_1, x_2, \ldots, x_p)$ denote the matrix of covariates, and let $\mathbf{Y}$ denote the response, where the x_i have been centered and scaled to have unit standard deviation and mean 0 and $\mathbf{Y}$ has mean 0. Then for a particular parameter k, the ridge regression coefficients $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_p)$ are obtained by solving

$$\left(\mathbf{X}'\mathbf{X} + k\mathbf{I}\right)\boldsymbol{\beta} = \mathbf{X}'\mathbf{Y}$$

For information about ridge regression and variance inflation factors, see Rawlings, Pantula, and Dickey (1998); Neter, Wasserman, and Kutner (1990); Marquardt and Snee (1975).

Joint Tests and Type 3 Tests

Linear hypotheses for $\boldsymbol{\beta}$ are expressed in matrix form as

$$H_0: \mathbf{L}\boldsymbol{\beta} = \mathbf{c}$$

where $\mathbf{L}$ is a matrix of coefficients for the linear hypotheses and $\mathbf{c}$ is a vector of constants. The vector of regression coefficients $\boldsymbol{\beta}$ includes slope parameters and intercept parameters. The F statistic for testing H_0 is computed as

$$F = (\mathbf{L}\widehat{\boldsymbol{\beta}} - \mathbf{c})'[\mathbf{L}\widehat{\mathbf{V}}(\widehat{\boldsymbol{\beta}})\mathbf{L}']^{-1}(\mathbf{L}\widehat{\boldsymbol{\beta}} - \mathbf{c})/r$$

where $\widehat{\mathbf{V}}(\widehat{\boldsymbol{\beta}}) = \hat{\sigma}^2(X'WX)^{-1}$ is the estimated covariance matrix of $\widehat{\boldsymbol{\beta}}$, with $\hat{\sigma}^2$ being the weighted residual sum of squares divided by the residual degrees of freedom d. Under H_0, F has an asymptotic F distribution with r and d degrees of freedom, where r is the rank of $\mathbf{L}$.

For models that use less-than-full-rank parameterization (as specified by the PARAM=GLM option in the CLASS statement), a Type 3 test of an effect of interest (main effect or interaction) is a test of the Type III estimable functions that are defined for that effect. When the model contains no missing cells, the Type 3 test of a main effect is equivalent to testing the hypothesis of equal marginal means. For more information about Type III estimable functions, see the chapter "The GLM Procedure" and the section "The Four Types of Estimable Functions" in *SAS/STAT User's Guide*. Also see Littell, Freund, and Spector (1991).

For models that use full-rank parameterization, all parameters are estimable when there are no missing cells, so it is unnecessary to define estimable functions. The standard test of an effect of interest in this case is the joint test that the values of the parameters associated with that effect are 0. For a model that uses effects parameterization (as specified by the PARAM=EFFECT option in the CLASS statement), the joint test for a main effect is equivalent to testing the equality of marginal means. For a model that uses reference parameterization (as specified by the PARAM=REF option in the CLASS statement), the joint test is equivalent to testing the equality of cell means at the reference level of the other model effects. For more information about the coding scheme and the associated interpretation of results, see Muller and Fetterman (2002, Chapter 14).

If there is no interaction term, the Type 3 test of an effect for a model that uses GLM parameterization is the same as the joint test of the effect for the model that uses full-rank parameterization. In this situation, the joint test is also called the Type 3 test. For a model that contains an interaction term and no missing cells, the Type 3 test of a component main effect under GLM parameterization is the same as the joint test of the component main effect under effect parameterization. Both test the equality of cell means. But this Type 3 test differs from the joint test under reference parameterization, which tests the equality of cell means at the reference level of the other component main effect. If some cells are missing, you can obtain meaningful tests only by testing a Type III estimation function, so in this case you should use GLM parameterization.

The results of a Type 3 test or a joint test do not depend on the order in which you specify the terms in the MODEL statement.

Displayed Output

The following sections describe the output produced by PROC REGSELECT. The output is organized into various tables, which are discussed in their order of appearance.

Selection Information

When you specify the SELECTION statement, the REGSELECT procedure produces by default a series of tables that display information about the model selection. The "Selection Information" table informs you about the model selection method; select, stop, and choose criteria; and other parameters that govern the selection. You can suppress this table by specifying DETAILS=NONE in the SELECTION statement.

Number of Observations

The "Number of Observations" table displays the number of observations read from the input data table and the number of observations used in the analysis. If you specify a FREQ statement, this table also displays the sum of frequencies read and used. If you use a PARTITION statement, the table also displays the number of observations used for each data role.

Class Level Information

The "Class Level Information" table lists the levels of every variable that you specify in the CLASS statement. You should check this information to make sure that the data are correct. You can adjust the order of the CLASS variable levels by specifying the ORDER= option in the CLASS statement. You can suppress the "Class Level Information" table completely or partially by specifying the NOCLPRINT= option in the PROC REGSELECT statement.

If the classification variables use a nonsingular parameterization, the "Class Level Information" table also displays the reference value for each variable.

Dimensions

The "Dimensions" table displays the number of effects and the number of parameters from which the selected model is chosen. If you use split classification variables, then this table also displays the number of effects after splitting is taken into account.

Entry and Removal Candidates

When you specify the DETAILS=ALL or DETAILS=STEPS option in the SELECTION statement, the REGSELECT procedure produces "Entry Candidates" and "Removal Candidates" tables that display the effect names and values of the criterion used to select entering or departing effects at each step of the selection process. The effects are displayed in sorted order from best to worst of the selection criterion.

Elastic Net Summary

When you specify METHOD=ELASTICNET without specifying an L2 value in the SELECTION statement, the REGSELECT procedure produces the "Elastic Net Summary" table, which displays the ridge regularization parameter L2 (in the ratio scaling) and CHOOSE criterion values in each L2 search step.

Relaxed LASSO Summary

When you specify METHOD=LASSO(RELAXED), the REGSELECT procedure produces the "Relaxed LASSO Summary" table, which displays the mixing parameter kappa and CHOOSE criterion values in each kappa search step.

Selection Summary

When you specify the SELECTION statement, the REGSELECT procedure produces the "Selection Summary" table, which displays information about the sequence of steps of the selection process. For each step, the effect that was entered or dropped is displayed along with the statistics used to select the effect, stop the selection, and choose the selected model. For all criteria that you can use for model selection, the steps at which the optimal values of these criteria occur are also indicated.

When you specify the METHOD=SCAD or METHOD=MCP option in the SELECTION statement, the "Selection Summary" table produces the information from each selection step. This information includes current regularization parameters, number of effects, CHOOSE criterion value, value of the objective function, and optimization convergence status.

You can suppress the display of the "Selection Summary" table by specifying DETAILS=NONE in the SELECTION statement.

Subset Selection Summary

When you specify the METHOD=BESTSUBSET option in the SELECTION statement, the REGSELECT procedure produces the "Subset Selection Summary" table, which displays information about the selected subset models that are found by the best-subset selection method. For each model, the model comparison criteria and selected parameters are displayed. You can request other model statistics and parameter estimates for each model by specifying appropriate suboptions in the METHOD=BESTSUBSET option.

Stop Reason

The "Stop Reason" table displays the reason why the selection stopped. To facilitate programmatic use of this table, an integer code is assigned to each reason and is included if you use an ODS OUTPUT statement or a DISPLAYOUT statement to output this table. The reasons and their associated codes follow:

Code	Stop Reason
1	All eligible effects are in the model.
2	All eligible effects have been removed.
3	Specified maximum number of steps done.
4	The model contains the specified maximum number of effects.
5	The model contains the specified minimum number of effects (for backward selection).
6	The stopping criterion is at a local optimum.
7	No suitable add or drop candidate could be found.
8	Adding or dropping any effect does not improve the selection criterion.
9	No candidate meets the appropriate SLE or SLS significance level.
10	Stepwise selection is cycling.
11	The model is an exact fit.
12	Dropping an effect would result in an empty model.

You can suppress the display of the "Stop Reason" table by specifying DETAILS=NONE in the SELECTION statement.

Selection Reason

When you specify the SELECTION statement, the REGSELECT procedure produces a simple table that explains why the final model was selected.

You can suppress the display of the "Selection Reason" table by specifying DETAILS=NONE in the SELECTION statement.

Selected Effects

When you specify the SELECTION statement, the REGSELECT procedure produces a simple table that lists which effects were selected for the final model.

ANOVA

The "ANOVA" table displays an analysis of variance for the selected model. This table includes the following:

- the Source of the variation, Model for the fitted regression, Error for the residual error, and C Total for the total variation after correcting for the mean. The Uncorrected Total Variation is produced when you specify the NOINT option.
- the degrees of freedom (DF) associated with the source
- the Sum of Squares for the term
- the Mean Square, which is the sum of squares divided by the degrees of freedom
- the *F* Value for testing the hypothesis that all parameters are 0 except for the intercept. This is formed by dividing the mean square for Model by the mean square for Error.
- the Prob>*F*, which is the probability of getting a greater *F* statistic than that observed if the hypothesis is true. When you do model selection, these *p*-values are usually liberal because they are not adjusted for the fact that the terms in the model have been selected.

You can request an "ANOVA" table for the model at each step of the selection process by specifying the DETAILS= option in the SELECTION statement.

Fit Statistics

The "Fit Statistics" table displays fit statistics for the selected model. The statistics include the following:

- Root MSE, an estimate of the standard deviation of the error term. It is calculated as the square root of the mean square error.
- R-square, a measure between 0 and 1 that indicates the portion of the (corrected) total variation attributed to the fit rather than left to residual error. It is calculated as SS(Model) divided by SS(Total) and is also called the *coefficient of determination*. It is the square of the multiple correlation—in other words, the square of the correlation between the dependent variable and the predicted values.
- Adj R-Sq, the adjusted R-square, which is a version of R-square that has been adjusted for degrees of freedom. It is calculated as

$$\bar{R}^2 = 1 - \frac{(n-i)(1-R^2)}{n-p}$$

  where i is equal to 1 if there is an intercept and 0 otherwise, n is the number of observations used to fit the model, and p is the number of parameters in the model.
- the fit criteria AIC, AICC, BIC, CP, and PRESS if they are used in the selection process. See Table 29.8 for the formulas to evaluate these criteria.
- the average square errors (ASE) on the training, validation, and test data

You can request a "Fit Statistics" table for the model at each step of the selection process by specifying the DETAILS= option in the SELECTION statement.

Parameter Estimates

The "Parameter Estimates" table displays the parameters in the selected model and their estimates. The information displayed for each parameter in the selected model includes the following:

- the parameter label that includes the effect name and level information for effects that contain classification variables
- the degrees of freedom (DF) for the parameter. There is one degree of freedom unless the model is not full rank.
- the parameter estimate
- the standard error, which is the estimate of the standard deviation of the parameter estimate
- t Value, the t test that the parameter is 0. This is computed as the parameter estimate divided by the standard error.

- the Pr > |t|, the probability that a t statistic would obtain a greater absolute value than that observed given that the true parameter is 0. This is the two-tailed significance probability.

  When you do model selection, these p-values are usually liberal because they are not adjusted for the fact that the terms in the model have been selected.

You can request a "Parameter Estimates" table for the model at each step of the selection process by specifying the DETAILS= option in the SELECTION statement.

Model Analysis of Variance (Type III)

When you specify the SS3 option in the MODEL statement, the REGSELECT procedure produces the "Model Analysis of Variance (Type III)" table. This table displays tests that all parameters for a particular effect are equal to zero. These tests are computed by using a Type III sum of squares, which yields a balanced test of each effect, adjusted for every other effect. For more information, see the section "Joint Tests and Type 3 Tests" on page 1509.

Model Crossproducts $X'XX'YY'Y$

When you specify the XPX option in the MODEL statement, the REGSELECT procedure produces the "Model Crossproducts $X'XX'YY'Y$" table. This table displays the $X'X$ crossproducts matrix for the model. The crossproducts matrix is bordered by the $X'Y$ and $Y'Y$ matrices. In addition, if you specify a selection method by using the SELECTION statement and obtain a model that contains a subset of effects, the table displays the $X'X$ crossproducts matrix for the selected model effects.

Ridge Regression Parameter Estimates

When you specify the RIDGE= option in the MODEL statement, the REGSELECT procedure produces the "Ridge Regression Parameter Estimates" table. This table displays the parameters and their estimates from ridge regression. The information that is displayed for each parameter includes the following:

- values of the ridge constant k
- the parameter label, which includes the effect name and level information for effects that contain classification variables
- the parameter estimates
- values of variance inflation factors and tolerance along with the parameter estimates when you specify the VIF and TOL options in the MODEL statement.

Ridge Regression Fit Statistics

When you specify the RIDGE= option in the MODEL statement, the REGSELECT procedure produces the "Ridge Regression Fit Statistics" table. This table displays fit statistics for the ridge regression models. The statistics include root MSE, R-square, Adj R-Sq, AIC, AICC, SBC, and ASE.

Timing

The "Timing" table displays the amount of time (in seconds) and the percentage of the time that PROC REGSELECT required to perform different tasks in the analysis.

OutputCasTables Table

The OutputCasTables table is a special table that has information about each CAS table that is created during a CAS action execution. The information for each CAS table consists of the CAS table name, the caslib in which the table resides, and the number of columns and rows in the CAS table. Because this table is not a typical ODS table that contains analytical results, you cannot include it in the *table-spec-list* in the DISPLAYOUT statement.

ODS Table Names

Each table that the REGSELECT procedure creates has a name associated with it. You must use this name to refer to the table when you use the DISPLAY statement, the DISPLAYOUT statement, or ODS statements. These names are listed in Table 29.10.

NOTE: The EFFECT statement also create tables which are not listed in this section. For information about these tables, see "ODS Table Names" on page 33 in Chapter 2, "Shared Concepts."

Table 29.10 ODS Tables Produced by PROC REGSELECT

Table Name	Description	Required Statement / Option
ANOVA	Selected model ANOVA table	Default output
BSplineDetails	B-spline basis details	EFFECT SPLINE BASIS=BSPLINE and DETAILS
Candidates	Swap candidates at step	SELECTION DETAILS=ALL | STEPS
ClassInfo	Level information from the CLASS statement	CLASS
CollectionLevelInfo	Level information for collection effects	EFFECT COLLECTION DETAILS
Crossproducts	Bordered model $X'X$ matrix	MODEL XPX
Dimensions	Model dimensions	Default output
ElasticNetSummary	Summary information about the L2 search steps	SELECTION METHOD=ELASTICNET
EntryCandidates	Candidates for entry at step	SELECTION DETAILS=ALL | STEPS
FitStatistics	Fit statistics	Default output

Table 29.10 *continued*

Table Name	Description	Required Statement / Option
MMLevelInfo	Level information for multimember effects	EFFECT MULTIMEMBER DETAILS
ModelAnova	Model analysis of variance (Type III)	MODEL SS3
ModelInfo	Information about the modeling environment	Default output
NObs	Number of observations read and used	Default output
OutputCasTables	See the section "OutputCasTables Table" on page 1515	OUTPUT DISPLAYOUT
ParameterEstimates	Solutions for the parameter estimates associated with effects in the MODEL statement	Default output
PolyDetails	Number of variables and columns, polynomial degree, and standardization method	EFFECT POLYNOMIAL DETAILS
PolyScaling	Centering and scaling details	EFFECT POLYNOMIAL DETAILS and STANDARDIZE
RelaxedLassoSummary	Summary information about the kappa search steps	SELECTION METHOD=LASSO(RELAXED)
RemovalCandidates	Candidates for removal at step	SELECTION DETAILS=ALL | STEPS
RidgeFitStats	Fit statistics of ridge regression models	MODEL RIDGE=
RidgeParmEst	Parameter estimates of ridge regression models	MODEL RIDGE=
SelectedEffects	List of selected effects	SELECTION
SelectionInfo	Information about selection settings	Default output
SelectionReason	Reason for selecting the final model	SELECTION
SelectionSummary	Summary information about the model selection steps	SELECTION
SplineKnots	Knot and boundary knot values	EFFECT SPLINE BASIS=BSPLINE and DETAILS
StopReason	Reason selection was terminated	SELECTION
SubsetSelectionSummary	Summary information about selected subset models	SELECTION METHOD=BESTSUBSET

Table 29.10 *continued*

Table Name	Description	Required Statement / Option
Timing	Timing breakdown by task	Default output
TPFSplineDetails	Truncated power function spline basis details	EFFECT SPLINE BASIS=TPF or BASIS=NATURALCUBIC and DETAILS

ODS Graphics

Statistical procedures use ODS Graphics to create graphs as part of their output. ODS Graphics is described in detail in Chapter 24, "Statistical Graphics Using ODS" (*SAS/STAT User's Guide*).

When ODS Graphics is enabled, the SELECTION statement can produce plots to help evaluate the selection process. For information about these plots, see the section "SELECTION Statement" on page 42 in Chapter 2, "Shared Concepts."

PROC REGSELECT assigns a name to each graph that it creates using ODS. You can use these names to reference the graphs when using ODS. The names are listed in Table 29.11.

Table 29.11 Graphs Produced by PROC REGSELECT

ODS Graph Name	Plot Description	PLOTS= Option
CoefficientPanel	Coefficients and CHOOSE= criterion by step	COEFFICIENTS
ChooseCriterionPlot	CHOOSE= criterion by step	COEFFICIENTS(UNPACK)
CoefficientPlot	Coefficients by step	COEFFICIENTS(UNPACK)

Table 29.11 *continued*

ODS Graph Name	Plot Description	PLOTS= Option
CriterionPanel	Fit criteria by step	CRITERIA
AdjRSqPlot	Max-rescaled (adjusted) R-square by step Adjusted R-square by number of parameters for best-subset selection	CRITERIA(UNPACK)
AICCPlot	Corrected Akaike's information criterion by step	CRITERIA(UNPACK)
AICPlot	Akaike's information criterion by step Akaike's information criterion by number of parameters for best-subset selection	CRITERIA(UNPACK)
BICPlot	Bayesian information criterion by number of parameters for best-subset selection	CRITERIA(UNPACK)
CPPlot	Mallows' C_p statistic by step Mallow's C_p statistic by number of parameters for best-subset selection	CRITERIA(UNPACK)
PRESSPlot	PRESS criterion by step	CRITERIA(UNPACK)
RSQUAREPLOT	R-square by number of parameters for best-subset selection	CRITERIA(UNPACK)
SBCPlot	Schwarz Bayesian information criterion by step Schwarz Bayesian information criterion by number of parameters for best-subset selection	CRITERIA(UNPACK)
TEST_ASEPlot	Average square error on testing data by step	CRITERIA(UNPACK)
TRAIN_ASEPlot	Average square error on training data by step	CRITERIA(UNPACK)
VAL_ASEPlot	Average square error on validation data by step	CRITERIA(UNPACK)
FitByRolePlot	Progression of average square error by role	FITBYROLE

Table 29.11 *continued*

ODS Graph Name	Plot Description	PLOTS= Option
RidgePanel	Variance inflation factor (VIF) values and standardized ridge estimates by ridge values for each coefficient	RIDGE
RidgePlot	Standardized ridge estimates by ridge values for each coefficient	RIDGE(UNPACK)
VifPlot	Variance inflation factor (VIF) values by ridge values for each coefficient	RIDGE(UNPACK)

Examples: REGSELECT Procedure

The examples in this chapter are available in the GitHub repository located at https://github.com/sascommunities/doc-supplement-casstat.

NOTE: Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections "Using CAS Sessions and CAS Engine Librefs" on page 7 and "Loading a SAS Data Set onto a CAS Server" on page 7 in Chapter 2, "Shared Concepts."

Example 29.1: Stepwise Selection with Validation

(View the complete code for this example (regselectex1.sas) in the example repository.)

This example is based on the example "Using Validation and Cross Validation" in the documentation for the GLMSELECT procedure in SAS/STAT software. The current example shows how you can use validation data to monitor and control variable selection. It also demonstrates the use of split classification variables.

The following DATA step produces analysis data that contain a variable that you can use to assign observations to the training, validation, and testing roles. In this case, each role has 5,000 observations.

```
%let nTotalObs=15000;
%let seed=1;
   data mylib.analysisData;
    drop i j c3Num nObsPerThread nExtras rew;
    length c3$ 7;

    array x{20} x1-x20;
    call streamInit(&seed);
    nObsPerThread  = int(&nTotalObs/_nthreads_);
    nExtras        = mod(&nTotalObs,_nthreads_);
    if _threadid_ <= nExtras then nObsPerThread =  nObsPerThread + 1;
```

```
   do i=1 to nObsPerThread;
      id = (_threadid_-1)*nObsPerThread +i;
      if _threadid_ > nExtras then id=id+nExtras;
      rew = rand('rewind', id);

      do j=1 to 20;
         x{j} = rand('UNIFORM');
      end;

      if  rand('UNIFORM') < 0.4 then byVar='A';
      else                           byVar='B';

      c1 = 1 + mod(id,8);
      c2 = rand('BERNOULLI',0.6);

      if      id < (0.001 * &nTotalObs) then do; c3 = 'tiny';     c3Num=1;end;
      else if id < (0.3   * &nTotalObs) then do; c3 = 'small';    c3Num=1;end;
      else if id < (0.7   * &nTotalObs) then do; c3 = 'average';  c3Num=2;end;
      else if id < (0.9   * &nTotalObs) then do; c3 = 'big';      c3Num=3;end;
      else                                   do; c3 = 'huge';     c3Num=5;end;

      yTrue = 10 + x1 + 2*x5 + 3*x10 + 4*x20  + 3*x1*x7 + 8*x6*x7
                 + 5*c3Num   + 8*(c1=7) + 8*(c1=3)*c2;
      error = 5*rand('NORMAL');
      y = yTrue + error;

           if mod(id,3)=1 then Role = 'TRAIN';
      else if mod(id,3)=2 then Role = 'VAL';
      else                     Role = 'TEST';

      output;
    end;
   run;
```

By construction, the true model consists of the main effects x1, x5, x10, x20, and c1 and the interaction effects x1*x7, x6*x7, and c1*c3. Furthermore, you can see that only levels 3 and 7 of the classification variable c1 are systematically related to the response.

Because the error term for each observation is five times a value drawn from a standard normal distribution, the expected error variance is 25. For the data in each role, you can compute an estimate of this error variance by forming the average square error (ASE) for the observations in the role. Output 29.1.1 shows the ASE for each role that you can compute with the following statements:

```
proc summary data=mylib.analysisData;
   class role;
   ways 1;
   var error;
   output out=ASE uss=uss n=n;
data ASE; set ASE;
   OracleASE = uss / n;
   label OracleASE = 'Oracle ASE';
   keep Role OracleASE;
proc print data=ASE label noobs;
run;
```

```
proc print data=ASE label noobs;
run;
```

Output 29.1.1 Oracle ASE Values by Role

Role	Oracle ASE
TEST	26.0510
TRAIN	25.0375
VAL	24.9843

The ASE values shown Output 29.1.1 are labeled as "Oracle ASE" because you need to know the true underlying model if you want to compute these values from the response and underlying regressors.

In a modeling context, a good predictive model produces values that are close to these oracle values. An overfit model produces a smaller ASE for the training data but higher values for the validation and test data. An underfit model exhibits higher values for all data roles.

Suppose you suspect that the dependent variable depends on both main effects and two-way interactions. You can use the following statements to select a model:

```
ods graphics on;
proc regselect data=mylib.analysisData;
   partition roleVar=role(train='TRAIN' validate='VAL' test='TEST');
   class c1 c2 c3;
   model y =  c1|c2|c3|x1|x2|x3|x4|x5|x5|x6|x7|x8|x9|x10
              |x11|x12|x13|x14|x15|x16|x17|x18|x19|x20 @2 /stb;
   selection method = stepwise(select=sl sle=0.1 sls=0.15 choose=validate)
                      hierarchy=single details=steps plots(startstep=5)=all;
run;
```

A PARTITION statement assigns observations to training, validation, and testing roles based on the values of the input variable named role. The SELECTION statement requests stepwise selection based on significance level (SL), where the SLE and SLS values are set to use the defaults of PROC REGSELECT. The CHOOSE=VALIDATE option selects the model that yields the smallest ASE value for the validation data.

The "Number of Observations" table in Output 29.1.2 confirms that there are 5,000 observations for each data role. The "Dimensions" table shows that the selection is from 278 effects with a total of 593 parameters.

Output 29.1.2 Number of Observations, Class Levels, and Dimensions

The REGSELECT Procedure

Number of Observations Read	15000
Number of Observations Used	15000
Number of Observations Used for Training	5000
Number of Observations Used for Validation	5000
Number of Observations Used for Testing	5000

Output 29.1.2 *continued*

Class Level Information		
Class	**Levels**	**Values**
c1	8	1 2 3 4 5 6 7 8
c2	2	0 1
c3	5	average big huge small tiny

Dimensions	
Number of Effects	278
Number of Parameters	593

Output 29.1.3 shows the "Selection Summary" table. You see that 17 steps are done, at which point all effects in the model are significant at the SLS value of 0.15 and all the remaining effects if added individually would not be significant at the SLE significance level of 0.1. However, because you specified the CHOOSE=VALIDATE option, the model at step 17 is not used as the selected model. Instead the model at step 11 (where the validation ASE achieves a local minimum value) is selected. The "Stop Reason," "Selection Reason," and "Selected Effects" in Output 29.1.4 tables provide this information.

Output 29.1.3 Selection Summary

The REGSELECT Procedure

Selection Details

Selection Summary				
Step	**Effect Entered**	**Number Effects In**	**Validation ASE**	**p Value**
0	**Intercept**	1	77.3492	.
1	**c3**	2	42.8635	<.0001
2	**c1**	3	34.0015	<.0001
3	**x7**	4	31.5269	<.0001
4	**x20**	5	30.1790	<.0001
5	**x6**	6	28.9493	<.0001
6	**x10**	7	28.2364	<.0001
7	**x1**	8	27.6796	<.0001
8	**x6*x7**	9	27.2929	<.0001
9	**x5**	10	26.9240	<.0001
10	**c2**	11	26.7270	<.0001
11	**c1*c2**	12	25.0709*	<.0001
12	**x5*x5**	13	25.0900	0.0142
13	**x5*c3**	14	25.1618	0.0472
14	**x7*x20**	15	25.1852	0.0629
15	**x7*c1**	16	25.2748	0.0638
16	**x6*c2**	17	25.2736	0.0705
17	**x19**	18	25.2945	0.0748
* Optimal Value Of Criterion				

Output 29.1.4 Stopping and Selection Reasons

Selection stopped because all candidates for removal are significant at the 0.15 level and no candidate for entry is significant at the 0.1 level.

The model at step 11 is selected where Validation ASE is 25.07093.

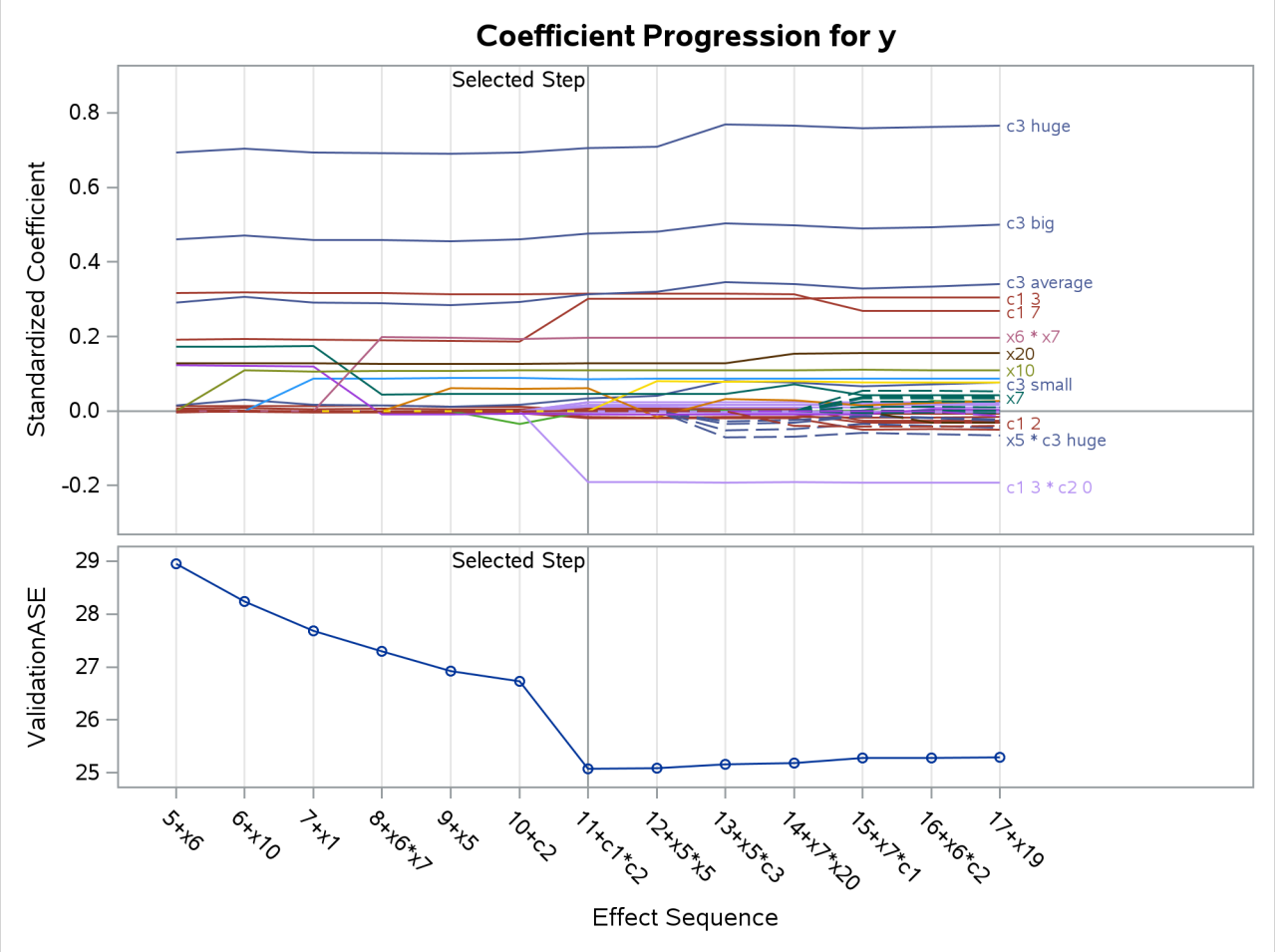
Selected Effects: Intercept c1 c2 c1*c2 c3 x1 x5 x6 x7 x6*x7 x10 x20

You can see that the selected effects include all the main effects in the true model and two of the three true interaction terms. Furthermore, the selected model does not include any variables that are not in the true model. Note that these statements are not true of the larger model at the final step of the selection process.

When you enable ODS Graphics, the PLOTS=ALL option in the SELECTION statement produces plots that illuminate the selection process. The STARTSTEP=5 suboption in the PLOTS= option requests that only steps 5 and beyond be displayed in the plots. This enables you to more easily assess how the fit criteria change near the selected step.

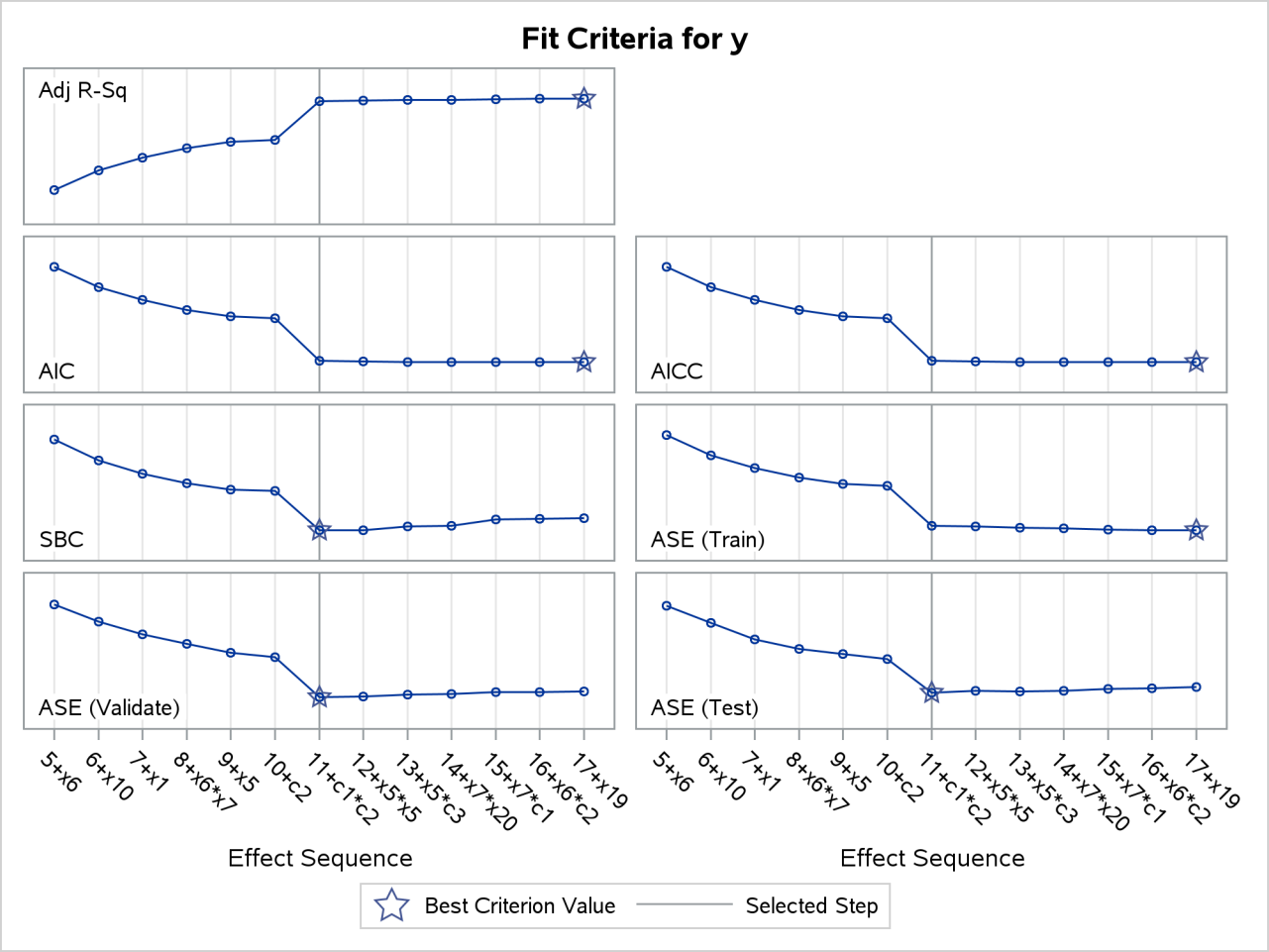
The upper plot in the coefficient panel in Output 29.1.5 shows how the standardized parameter estimates change with the selection step after step 5. The lower plot in the panel shows how the average square error on the validation data, which you specified as the CHOOSE= criterion, changes with the selection step.

Output 29.1.5 Coefficient Panel



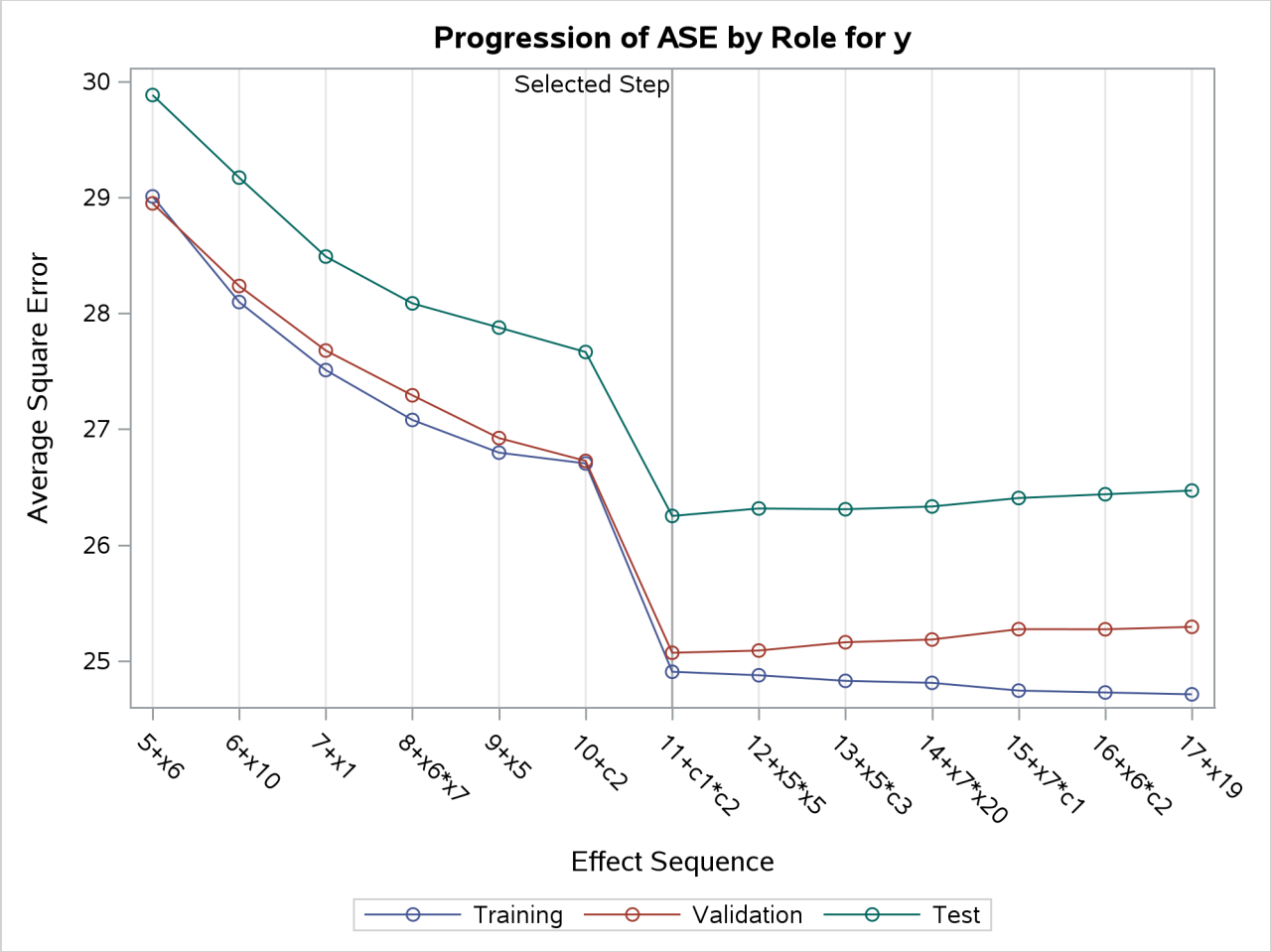
The criterion panel in Output 29.1.6 shows how the various fit criteria change as the stepwise selection method proceeds. This plot shows that if you had used the SBC statistic as the CHOOSE= criterion, then the selected model would still be the model at step 11 However, the AIC or AICC statistic continues to decline for all steps of the selection process. If you had used either of these criteria as the CHOOSE= criterion, then the final model at step 17 would have been selected.

Output 29.1.6 Criterion Panel



Output 29.1.7 shows how the average square error compares for the training, test, and validation data. You see that the average square error on both the test and validation data starts to increase beyond the selected step. This indicates that the selected model does not overfit the training data.

Output 29.1.7 Average Square Error Plot



Output 29.1.8 shows the fit statistics of the selected model. You can see that the ASE values for the training, validation, and test data are all similar, indicating a reasonable predictive model. In this case, where the true model is known, you can see that all three ASE values are close to the oracle values for the true model, as shown in Output 29.1.1.

Output 29.1.8 Fit Statistics for the Selected Model

Root MSE	5.00421
R-Square	0.67654
Adj R-Sq	0.67485
AIC	21132
AICC	21132
SBC	16306
ASE (Train)	24.90688
ASE (Validate)	25.07093
ASE (Test)	26.25177

Because you specified the DETAILS=STEPS option in the SELECTION statement, you can see the fit statistics for the model at each step of the selection process. Output 29.1.9 shows these fit statistics for the

final model at step 17. You see that for this model, the ASE value for the training data is smaller than the ASE values for the validation and test data. This indicates an overfit model that might not generalize well to new data. You see that the ASE values for the validation and test data are now worse in comparison to the oracle values than the values for the selected model at step 11.

Output 29.1.9 Fit Statistics for the Model at Step 17

Root MSE	4.99218
R-Square	0.67907
Adj R-Sq	0.67641
AIC	21123
AICC	21123
SBC	16394
ASE (Train)	24.71247
ASE (Validate)	25.29450
ASE (Test)	26.47078

Output 29.1.10 shows part of the "Parameter Estimates" table for the selected model at step 11 that includes the estimates for the main effect c1. Because you specified the STB option in the MODEL statement, this table includes standardized estimates.

Output 29.1.10 Part of the Parameter Estimates Table for the Selected Model

Parameter Estimates

Parameter	DF	Estimate	Standardized Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	13.672688	0	2.283909	5.99	<.0001
c1 1	1	-0.427892	-0.016127	0.369633	-1.16	0.2471
c1 2	1	-0.511935	-0.019294	0.365942	-1.40	0.1619
c1 3	1	7.970490	0.300395	0.363361	21.94	<.0001
c1 4	1	-0.248256	-0.009356	0.367053	-0.68	0.4989
c1 5	1	0.051535	0.001942	0.366794	0.14	0.8883
c1 6	1	0.170392	0.006422	0.363975	0.47	0.6397
c1 7	1	8.342768	0.314426	0.366337	22.77	<.0001
c1 8	0	0	0	.	.	.

The magnitudes of the standardized estimates and the *t* statistics of the parameters of the effect c1 reveal that only levels 3 and 7 of this effect contribute appreciably to the model. This suggests that you might obtain a more parsimonious model with similar or better predictive power if parameters that correspond to the levels of c1 can enter or leave the model independently. You request this by specifying the SPLIT option in the CLASS statement, as shown in the following statements:

```
proc regselect data=mylib.analysisData;
   partition roleVar=role(train='TRAIN' validate='VAL' test='TEST');
   class c1(split) c2 c3;
   model y = c1|c2|c3|x1|x2|x3|x4|x5|x5|x6|x7|x8|x9|x10
             |x11|x12|x13|x14|x15|x16|x17|x18|x19|x20 @2 /stb;
   selection method = stepwise(select=sl sle=0.1 sls=0.15 choose=validate)
             hierarchy=single details=steps;
run;
```

Output 29.1.11 shows the "Dimensions" table. You can see that because the columns in the design matrix that correspond to the levels of c1 are treated as separate effects, the selection is now from 439 effects, even though the number of parameters is unchanged.

Output 29.1.11 Dimensions with c1 Split

The REGSELECT Procedure

Dimensions	
Number of Effects	439
Number of Parameters	593

Output 29.1.12 shows the selected effects. You can see that, as anticipated, the selected model now depends on only levels 3 and 7 of c1.

Output 29.1.12 Selected Effects with c1 Split

Selected Effects:	Intercept c1_3 c1_7 c2 c1_3*c2 c3 x1 x5 x6 x7 x6*x7 x10 x20

Finally, the fit statistics for the selected model are shown in Output 29.1.13.

Output 29.1.13 Fit Statistics for the Selected Model with c1 Split

Root MSE	5.00424
R-Square	0.67582
Adj R-Sq	0.67485
AIC	21121
AICC	21121
SBC	16223
ASE (Train)	24.96224
ASE (Validate)	25.06729
ASE (Test)	26.21366

If you compare the ASE values for this model in Output 29.1.13 with the oracle values in Output 29.1.1 and the values for the model without splitting c1 in Output 29.1.8, you see that this more parsimonious model produces the best predictive performance for the test data of all the models considered in this example.

Example 29.2: LASSO and Elastic Net Selection

(View the complete code for this example (regselectEx2.sas) in the example repository.)

This example continues the investigation of the baseball data set that were introduced in the section "Getting Started: REGSELECT Procedure" on page 1480. That example uses the default stepwise selection method, which is based on the SBC criterion to select a model. This example explores model selection by using LASSO and elastic net selection methods with validation.

You can load the Sashelp.Baseball data set into your CAS session by specifying your CAS engine libref in the first statement in the following DATA step:

```
data mylib.baseball;
   set sashelp.baseball;
run;
```

The following statements perform LASSO selection with validation for the baseball data:

```
ods graphics on;
proc regselect data=mylib.baseball;
   partition roleVar = league(train='National' validate = 'American');
   class division;
   model logSalary = division nAtBat nHits nHome nRuns nRBI nBB
                     yrMajor crAtBat crHits crHome crRuns crRbi
                     crBB nOuts nAssts nError;
   selection method = lasso(choose=validate) plots=all;
run;
```

The PARTITION statement assigns observations to training and validation roles on the basis of the values of the input variable league. The CHOOSE=VALIDATE option in the SELECTION statement selects the model that yields the smallest average square error (ASE) value for the validation data.

Output 29.2.1 shows that 124 observations are used for model training, 139 observations are used for model validation, and the number of effects is 19.

Output 29.2.1 Basic Information about LASSO Selection for baseball Data

The REGSELECT Procedure

Selection Information	
Selection Method	LASSO
Choose Criterion	Validation ASE
Stop Criterion	SBC
Effect Hierarchy Enforced	None
Stop Horizon	3

Number of Observations Read	322
Number of Observations Used	263
Number of Observations Used for Training	124
Number of Observations Used for Validation	139
Number of Observations Used for Testing	0

Output 29.2.1 *continued*

Dimensions	
Number of Effects	19
Number of Parameters	19

The "Selection Summary" table in Output 29.2.2 shows the effect that was added or dropped at each step of the selection process together with the SBC and validation ASE for the model at each step. You see that the local minimum of SBC occurs at step 11 and that three more steps are completed before the procedure stops (because STOPHORIZON=3). The model that has lowest SBC is not used as the selected model because CHOOSE=VALIDATE. Instead, the model at step 14 (where the validation ASE achieves a local minimum value) is selected. Details of this selected model are shown in Output 29.2.3.

Output 29.2.2 Selection Summary of LASSO Selection for baseball Data

The REGSELECT Procedure

Selection Details

Selection Summary

Step	Effect Entered	Effect Removed	Number Effects In	SBC	Validation ASE
0	**Intercept**		1	-26.8275	0.7992
1	**CrRbi**		2	-47.5668	0.6415
2	**nRBI**		3	-50.9604	0.5958
3	**nBB**		4	-47.0607	0.5908
4	**CrAtBat**		5	-56.0563	0.5184
5	**YrMajor**		6	-71.9243	0.4330
6	**nHits**		7	-69.4898	0.4233
7	**nAtBat**		8	-78.6487	0.3766
8	**CrHits**		9	-75.8090	0.3708
9	**nHome**		10	-73.9605	0.3649
10		**CrRbi**	9	-80.4248	0.3618
11		**CrAtBat**	8	-87.0397*	0.3591
12	**nOuts**		9	-86.1885	0.3534
13	**Division_West**		10	-83.3840	0.3489
14	**nError**		11	-81.3255	0.3488*

*** Optimal Value Of Criterion**

Selection stopped at a local minimum of the SBC criterion.

The model at step 14 is selected where Validation ASE is 0.348804.

Selected Effects: Intercept Division_West nAtBat nHits nHome nBB YrMajor CrHits nOuts nError

Output 29.2.3 Details of Model Selected by LASSO for baseball Data

The REGSELECT Procedure

Selected Model

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	9	54.10354	6.01150	16.33	<.0001
Error	114	41.96456	0.36811		
Corrected Total	123	96.06810			

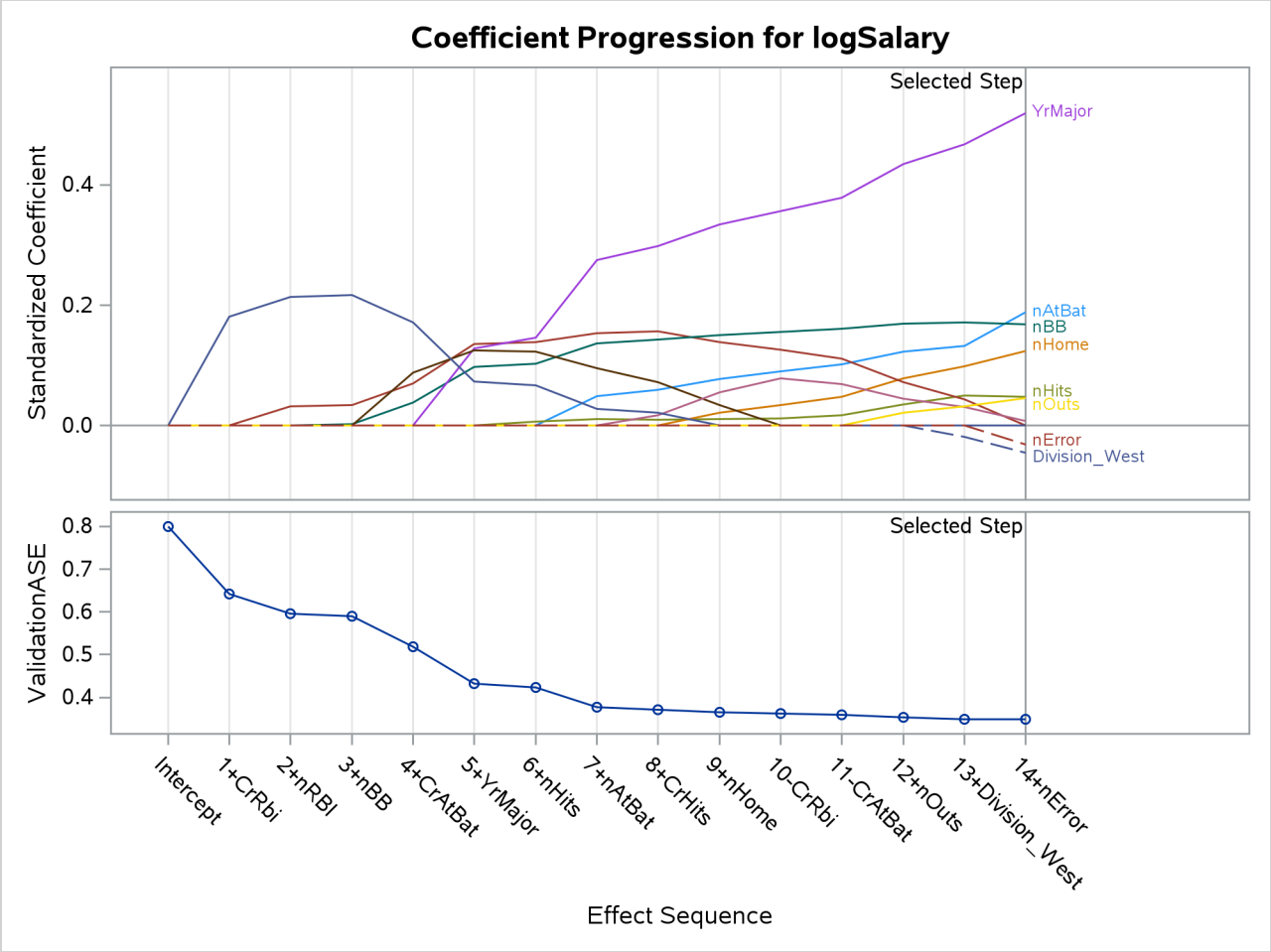
Root MSE	0.60672
R-Square	0.56318
Adj R-Sq	0.52869
AIC	11.65144
AICC	14.00859
SBC	-86.14574
ASE (Train)	0.33842
ASE (Validate)	0.34880

Parameter Estimates

Parameter	DF	Estimate
Intercept	1	4.222406
Division_West	1	-0.078574
nAtBat	1	0.001202
nHits	1	0.001017
nHome	1	0.014405
nBB	1	0.007625
YrMajor	1	0.095327
CrHits	1	0.000009670
nOuts	1	0.000141
nError	1	-0.004048

The coefficient panel in Output 29.2.4 enables you to visualize the selection process. In this plot, standardized coefficients of all the effects that are selected at some step of the LASSO method are plotted as a function of the step number. The panel's lower plot shows how the criterion that is used to choose the selected model changes as effects enter or leave the model.

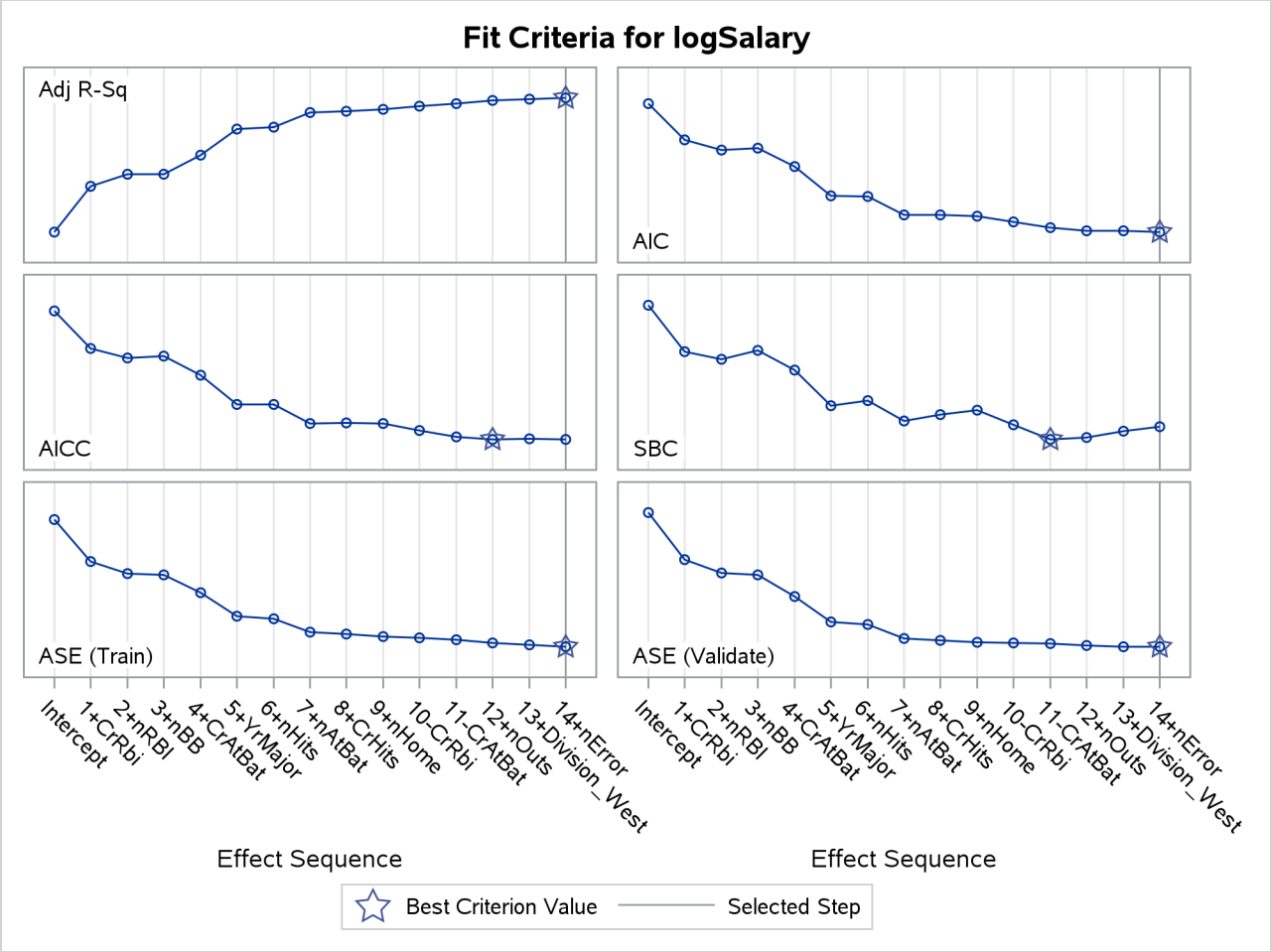
Output 29.2.4 LASSO Coefficient Progression



The criterion panel in Output 29.2.5 shows how the various fit criteria change as the LASSO selection method proceeds. This plot shows that if you had used the SBC statistic as the CHOOSE= criterion, then the selected model would be the model at step 11. However, the AIC and validation ASE continue to decline for all steps of the selection process.

The "Parameter Estimates" table in Output 29.2.3 shows that the coefficients of the parameters are in different scales. In this case, you can use the adaptive LASSO, in which adaptive weights are used for penalizing different coefficients in the L1 penalty. For more information about this selection method, see the section "Adaptive LASSO Selection" on page 82 in Chapter 2, "Shared Concepts."

Output 29.2.5 Criterion Panel



The following statements perform adaptive LASSO selection with validation for the baseball data:

```
proc regselect data=mylib.baseball;
   partition roleVar = league(train='National' validate = 'American');
   class division;
   model logSalary = division nAtBat nHits nHome nRuns nRBI nBB
                     yrMajor crAtBat crHits crHome crRuns crRbi
                     crBB nOuts nAssts nError;
   selection method = lasso(adaptive choose=validate);
run;
```

The results of adaptive LASSO selection are shown in Output 29.2.6. You can see that the model that is chosen by adaptive LASSO selection has fewer parameters and a lower validation ASE than the model chosen by LASSO selection.

Output 29.2.6 Details of Model Selected by Adaptive LASSO for baseball Data

The REGSELECT Procedure

Selected Model

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	6	53.21414	8.86902	24.21	<.0001
Error	117	42.85396	0.36627		
Corrected Total	123	96.06810			

Root MSE	0.60521
R-Square	0.55392
Adj R-Sq	0.53105
AIC	8.25204
AICC	9.50421
SBC	-98.00599
ASE (Train)	0.34560
ASE (Validate)	0.34501

Parameter Estimates

Parameter	DF	Estimate
Intercept	1	4.203747
Division_East	1	0.031888
nHits	1	0.004381
nHome	1	0.017141
nBB	1	0.009164
YrMajor	1	0.093256
CrHits	1	0.000028569

As shown in the previous example, adaptive LASSO selection can improve model fit by adaptively penalizing regression parameters. Relaxed LASSO can also achieve this goal by mixing least squares estimates with LASSO estimates. A noteworthy feature of relaxed LASSO is that you can use the least square estimates for model evaluation after variables are chosen by LASSO selection. You can use the LSCOEFFS suboption to perform the analysis. The following statements perform relaxed LASSO selection with ordinary least squares (OLS) coefficients for the baseball data:

```
proc regselect data=mylib.baseball;
   partition roleVar = league(train='National' validate = 'American');
   class division;
   model logSalary = division nAtBat nHits nHome nRuns nRBI nBB
                     yrMajor crAtBat crHits crHome crRuns crRbi
                     crBB nOuts nAssts nError;
   selection method = lasso(lscoeffs choose=validate);
run;
```

The results of LASSO selection with OLS coefficients are shown in Output 29.2.7. You can see that the selected model has fewer parameters and a lower validation ASE than the model chosen by original LASSO selection.

Output 29.2.7 Details of Model Selected by LASSO with OLS Coefficients for baseball Data

The REGSELECT Procedure

Selected Model

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	6	52.57372	8.76229	23.57	<.0001
Error	117	43.49438	0.37175		
Corrected Total	123	96.06810			

Root MSE	0.60971
R-Square	0.54725
Adj R-Sq	0.52404
AIC	10.09142
AICC	11.34360
SBC	-96.16661
ASE (Train)	0.35076
ASE (Validate)	0.34000

Parameter Estimates

Parameter	DF	Estimate
Intercept	1	4.130463
nHits	1	0.003124
nRBI	1	0.007459
nBB	1	0.009771
YrMajor	1	0.095590
CrAtBat	1	0.000025438
CrRbi	1	-0.000177

Instead of directly using the least squares estimates to evaluate candidate models, you can use a weighted combination of LASSO estimates and least squares estimates for this purpose. The following statements use this combination to perform relaxed LASSO selection with validation for the baseball data:

```
proc regselect data=mylib.baseball;
   partition roleVar = league(train='National' validate = 'American');
   class division;
   model logSalary = division nAtBat nHits nHome nRuns nRBI nBB
                     yrMajor crAtBat crHits crHome crRuns crRbi
                     crBB nOuts nAssts nError;
   selection method = lasso(relaxed choose=validate);
run;
```

The results are shown in Output 29.2.8. You can see that the model that is chosen by relaxed LASSO selection has fewer parameters and a lower validation ASE than the model chosen by LASSO selection.

The "Relaxed LASSO Selection Summary" table in Output 29.2.9 shows that the selected kappa value is 0.1. It also shows that the minimal validation ASE shown in Output 29.2.8 is lower than both the one in original LASSO selection and the one in LASSO selection with the LSCOEFFS suboption. Note that relaxed LASSO selection uses the mixing parameter kappa to control the balance between LASSO solution and

OLS solution. By comparing Output 29.2.3 with Output 29.2.9, you can see that the validation ASE from relaxed LASSO selection with a kappa value of 1 is the same as the validation ASE from original LASSO selection. By comparing Output 29.2.7 with Output 29.2.9, you can see that the validation ASE from relaxed LASSO selection with a kappa value of 0 is the same as the validation ASE from LASSO selection with OLS coefficients. For more information about this selection method, see the section "Relaxed LASSO Selection" on page 83 in Chapter 2, "Shared Concepts."

Output 29.2.8 Details of Model Selected by Relaxed LASSO for baseball Data

The REGSELECT Procedure

Selected Model

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	6	52.46932	8.74489	23.47	<.0001
Error	117	43.59878	0.37264		
Corrected Total	123	96.06810			

Root MSE	0.61044
R-Square	0.54617
Adj R-Sq	0.52289
AIC	10.38870
AICC	11.64088
SBC	-95.86932
ASE (Train)	0.35160
ASE (Validate)	0.33854

Parameter Estimates

Parameter	DF	Estimate
Intercept	1	4.226245
nHits	1	0.002826
nRBI	1	0.007235
nBB	1	0.009262
YrMajor	1	0.088724
CrAtBat	1	0.000027369
CrRbi	1	-0.000142

Output 29.2.9 Relaxed LASSO Selection Summary for baseball Data

The REGSELECT Procedure

Selection Details

Relaxed LASSO Summary			
Step	Kappa	Number Effects In	Validation ASE
1	0	7	0.339995
2	0.1	7	0.338536*
3	0.2	7	0.339253
4	0.3	7	0.342145
5	0.4	7	0.347212
6	0.5	9	0.350871
7	0.6	9	0.353167
8	0.7	9	0.356308
9	0.8	10	0.347139
10	0.9	10	0.347884
11	1	11	0.348804
* Optimal Value of Criterion			

You might be able to further improve the selection by specifying a finer grid near the current optimal kappa value 0.1, as shown in the following statements:

```
proc regselect data=mylib.baseball;
   partition roleVar = league(train='National' validate = 'American');
   class division;
   model logSalary = division nAtBat nHits nHome nRuns nRBI nBB
                     yrMajor crAtBat crHits crHome crRuns crRbi
                     crBB nOuts nAssts nError;
   selection method = lasso(relaxed(kappa=0.05 to 0.15  by 0.001) choose=validate);
run;
```

The "Relaxed LASSO Selection Summary" table in Output 29.2.10 shows that the new optimal kappa value is 0.117. Output 29.2.11 shows that the minimal validation ASE is further improved.

Output 29.2.10 Relaxed LASSO Selection Summary for baseball Data

Relaxed LASSO Summary

Step	Kappa	Number Effects In	Validation ASE
1	0.05	7	0.338994
2	0.051	7	0.338979
3	0.052	7	0.338965
.	.	.	.
	.	.	.
	.	.	.
66	0.115	7	0.338505
67	0.116	7	0.338505
68	0.117	7	0.338504*
69	0.118	7	0.338505
70	0.119	7	0.338505
.	.	.	.
	.	.	.
	.	.	.
99	0.148	7	0.338609
100	0.149	7	0.338615
101	0.15	7	0.338622

*** Optimal Value of Criterion**

Output 29.2.11 Details of Model Selected by Relaxed LASSO for baseball Data

Model Selected by Relaxed LASSO

Root MSE	0.61071
R-Square	0.54577
Adj R-Sq	0.52247
AIC	10.49819
AICC	11.75037
SBC	-95.75984
ASE (Train)	0.35191
ASE (Validate)	0.33850

Parameter Estimates

Parameter	DF	Estimate
Intercept	1	4.242527
nHits	1	0.002776
nRBI	1	0.007196
nBB	1	0.009176
YrMajor	1	0.087557
CrAtBat	1	0.000027698
CrRbi	1	-0.000136

From the previous examples of relaxed LASSO selection, you can see that relaxed LASSO usually produces sparser models with equal or lower prediction error than ordinary LASSO selection produces.

If you are interested only in the catchers' salaries, you can use the following statements to perform LASSO selection for the baseball_C data:

```
proc regselect data = mylib.baseball(where=(Position='C'));
   partition roleVar = league(train='National' validate = 'American');
   class division;
   model logSalary = division nAtBat nHits nHome nRuns nRBI nBB
                     yrMajor crAtBat crHits crHome crRuns crRbi
                     crBB nOuts nAssts nError;
   selection method = lasso(choose=validate);
run;
```

Output 29.2.12 shows that 15 observations were used for training and the number of effects is 19. This implies that you cannot obtain a good result by using LASSO selection because the number of effects is greater than the number of observations. In this case, you can use elastic net selection instead of LASSO selection. For more information about this selection method, see the section "Elastic Net Selection" on page 83 in Chapter 2, "Shared Concepts."

Output 29.2.12 Basic Information about LASSO Selection for baseball_C Data

The REGSELECT Procedure

Number of Observations Read	40
Number of Observations Used	30
Number of Observations Used for Training	15
Number of Observations Used for Validation	15
Number of Observations Used for Testing	0

Dimensions	
Number of Effects	19
Number of Parameters	19

Output 29.2.13 Details of the Model Selected by LASSO Selection for baseball_C Data

The REGSELECT Procedure

Selected Model

Analysis of Variance					
Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	5	8.70333	1.74067	3.95	0.0356
Error	9	3.96189	0.44021		
Corrected Total	14	12.66521			

Output 29.2.13 *continued*

Root MSE	0.66348
R-Square	0.68718
Adj R-Sq	0.51340
AIC	9.03005
AICC	25.03005
SBC	-3.72165
ASE (Train)	0.26413
ASE (Validate)	0.30621

Parameter Estimates

Parameter	DF	Estimate
Intercept	1	5.093821
nHits	1	0.001115
nRuns	1	0.006507
nBB	1	0.004172
CrAtBat	1	0.000237
nAssts	1	0.000372

The following statements perform elastic net selection for the baseball_C data:

```
proc regselect data=mylib.baseball(where=(Position='C'));
   partition roleVar = league(train='National' validate = 'American');
   class division;
   model logSalary = division nAtBat nHits nHome nRuns nRBI nBB
                     yrMajor crAtBat crHits crHome crRuns crRbi
                     crBB nOuts nAssts nError;
   selection method = elasticnet(choose=validate);
   ods output ElasticNetSummary=ElasticNetSummary;
run;
```

You can specify the L2= suboption of the METHOD option in the SELECTION statement if you have a good estimate of the ridge regression parameter. If not, you can leave the L2= suboption unspecified; in this case, PROC REGSELECT estimates this regression parameter according to the criterion that is specified in the CHOOSE= option, by trying a series of candidate values for the ridge regression parameter. The optimal regression parameter is set to the value that achieves the minimum validation ASE.

The "Elastic Net Selection Summary" table in Output 29.2.15 shows that the optimal L2 value is 0.14677993 and the minimal validation ASE is 0.17967. Output 29.2.14 shows the details of the model that is selected by elastic net method with optimal L2 value. If you compare the ASE values for the model in Output 29.2.14 and the model in Output 29.2.13, you see that the elastic net method selects a model that has a lower validation ASE than the LASSO method does in this case.

Output 29.2.14 Details of the Model Selected by Elastic Net Selection for baseball_C Data

The REGSELECT Procedure

Selected Model

Root MSE	0.60490
R-Square	0.82666
Adj R-Sq	0.59553
AIC	6.17483
AICC	61.17483
SBC	-4.45272
ASE (Train)	0.14636
ASE (Validate)	0.17967

Parameter Estimates

Parameter	DF	Estimate
Intercept	1	4.435995
nHits	1	0.001148
nRuns	1	0.007537
nBB	1	0.006697
YrMajor	1	0.028748
CrAtBat	1	0.000118
CrHits	1	0.000383
nAssts	1	0.003097
nError	1	0.017681

Output 29.2.15 Elastic Net Selection Summary for baseball_C Data

The REGSELECT Procedure

Selection Details

Elastic Net Summary			
Step	L2	Number Effects In	Validation ASE
1	0.00000000	6	0.3062
2	0.00000001	6	0.3062
3	0.00000001	6	0.3062
4	0.00000002	6	0.3062
5	0.00000003	6	0.3062
6	0.00000005	6	0.3062
7	0.00000007	6	0.3062
8	0.00000010	6	0.3062
9	0.00000015	6	0.3062
10	0.00000022	6	0.3062
11	0.00000032	6	0.3062
12	0.00000046	6	0.3062
13	0.00000068	6	0.3062
14	0.00000100	6	0.3062
15	0.00000147	6	0.3062
16	0.00000215	6	0.3062
17	0.00000316	6	0.3062
18	0.00000464	6	0.3062
19	0.00000681	6	0.3062
20	0.00001000	6	0.3062
21	0.00001468	6	0.3062
22	0.00002154	6	0.3062
23	0.00003162	6	0.3062
24	0.00004642	6	0.3062
25	0.00006813	6	0.3062
26	0.00010000	6	0.3062
27	0.00014678	6	0.3062
28	0.00021544	6	0.3063
29	0.00031623	6	0.2029
30	0.00046416	6	0.2030
31	0.00068129	6	0.2031
32	0.00100000	6	0.2033
33	0.00146780	6	0.2036
34	0.00215443	6	0.2040
35	0.00316228	6	0.2046
36	0.00464159	7	0.3064
37	0.00681292	7	0.3058
38	0.01000000	7	0.2094
39	0.01467799	7	0.2122
40	0.02154435	8	0.2147
41	0.03162278	9	0.1993
42	0.04641589	9	0.1803

*** Optimal Value Of Criterion**

Output 29.2.15 *continued*

The REGSELECT Procedure

Selection Details

Elastic Net Summary

Step	L2	Number Effects In	Validation ASE
43	0.06812921	9	0.1866
44	0.10000000	9	0.1854
45	0.14677993	9	0.1797*
46	0.21544347	9	0.2985
47	0.31622777	4	0.5786
48	0.46415888	4	0.6210
49	0.68129207	4	0.6675
50	1.00000000	4	0.7458

*** Optimal Value Of Criterion**

Similar to the previous analysis, which used the adaptive LASSO selection, you can also perform adaptive elastic net selection on the data when parameter estimates has different scales. The following statements perform adaptive elastic net selection for the baseball_C data:

```
proc regselect data=mylib.baseball(where=(Position='C'));
   partition roleVar = league(train='National' validate = 'American');
   class division;
   model logSalary = division nAtBat nHits nHome nRuns nRBI nBB
                     yrMajor crAtBat crHits crHome crRuns crRbi
                     crBB nOuts nAssts nError;
   selection method = elasticnet(adaptive choose=validate);
run;
```

The "Adaptive Elastic Net Selection Summary" table in Output 29.2.16 shows that the optimal L2 value is 0.0215 and the minimal validation ASE is 0.1661. Output 29.2.17 shows that adaptive elastic net selection can produce better results in the sense of a lower validation ASE and a sparser model. For more information about this method, see the section "Adaptive Elastic Net Selection" on page 86 in Chapter 2, "Shared Concepts."

Output 29.2.16 Adaptive Elastic Net Selection Summary for baseball_C Data

Elastic Net Summary

Step	L2	Number Effects In	Validation ASE
1	0.00000000	6	0.2780
2	0.00000001	5	0.1736
3	0.00000001	5	0.1736
	.	.	.
	.	.	.
	.	.	.
37	0.00681292	6	0.1952
38	0.01000000	5	0.1789
39	0.01467799	6	0.1667
40	0.02154435	7	0.1661*
41	0.03162278	7	0.1688
42	0.04641589	6	0.1986
43	0.06812921	6	0.2385
	.	.	.
	.	.	.
	.	.	.
48	0.46415888	4	0.5973
49	0.68129207	4	0.6637
50	1.00000000	4	0.7061

*** Optimal Value Of Criterion**

Output 29.2.17 Details of the Model Selected by Adaptive Elastic Net for baseball_C Data

Selected Model by Adaptive Elastic Net

Root MSE	0.46566
R-Square	0.86303
Adj R-Sq	0.76031
AIC	-1.35829
AICC	22.64171
SBC	-13.40194
ASE (Train)	0.11565
ASE (Validate)	0.16609

Parameter Estimates

Parameter	DF	Estimate
Intercept	1	4.159782
nRuns	1	0.015687
nBB	1	0.003749
YrMajor	1	0.027027
CrAtBat	1	0.000256
nAssts	1	0.002826
nError	1	0.029585

Example 29.3: SCAD and MCP Selection

(View the complete code for this example (regselectEx3.sas) in the example repository.)

This example continues the investigation of the baseball data set that was introduced in the section "Getting Started: REGSELECT Procedure" on page 1480 and in "Example 29.2: LASSO and Elastic Net Selection" on page 1528. That example uses convex penalized selection—the LASSO and elastic net methods—to select an optimal model. In comparison, this example explores model selection by using folded concave penalized selection—the smoothly clipped absolute deviation (SCAD) and minimax concave penalty (MCP)—to select an optimal model.

The following statements perform SCAD selection for the baseball data:

```
proc regselect data=mylib.baseball;
   partition roleVar = league(train='National' validate='American');
   class division;
   model logSalary = division nAtBat nHits nHome nRuns nRBI nBB
                     yrMajor crAtBat crHits crHome crRuns crRbi
                     crBB nOuts nAssts nError;
   selection method = scad(choose=validate) details=all;
run;
```

The PARTITION statement assigns observations to training and validation roles on the basis of the values of the input variable league. The METHOD=SCAD option requests SCAD selection, and the CHOOSE=VALIDATE suboption in the SELECTION statement selects the model that yields the lowest average square error (ASE) value for the validation data.

The "Selection Information" table in Output 29.3.1 shows the information that is relevant to SCAD selection, including the solver type (mixed integer linear programming, or MILP) and the ranges of the regularization parameters.

Output 29.3.1 Basic Information about SCAD Selection with MILP Solver

Basic Information about SCAD Selection

Number of Observations Read	322
Number of Observations Used	263
Number of Observations Used for Training	124
Number of Observations Used for Validation	139
Number of Observations Used for Testing	0

Class Level Information

Class	Levels	Values
Division	2	East West

Dimensions

Number of Effects	19
Number of Parameters	19

Output 29.3.1 *continued*

Selection Information	
Selection Method	SCAD
Solver	MILP
Choose Criterion	Validation ASE
Maximum Alpha	5.7
Minimum Alpha	2.7
Alpha Steps	4
Maximum Lambda	0.354142
Minimum Lambda	0.0104153
Lambda Steps	10
Lambda Grid	LOGSPACE

Recall that the SCAD estimator $\hat{\boldsymbol{\beta}}^{\text{scad}}$ solves the following minimization problem:

$$\hat{\boldsymbol{\beta}}^{\text{scad}} = \arg\min_{\boldsymbol{\beta}} \left\{ \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + n\sum_{j=1}^{m} P_{\lambda,\alpha}^{\text{scad}}(|\beta_j|) \right\}$$

where the $P_{\lambda,\alpha}(\cdot)$ takes the form

$$P_{\lambda,\alpha}^{\text{scad}}(\theta) = \begin{cases} \lambda\theta & \text{if} \quad 0 \le \theta \le \lambda \\ \frac{-1}{2(\alpha-1)}\left(\theta^2 - 2\alpha\lambda\theta + \lambda^2\right) & \text{if} \quad \lambda < \theta < \alpha\lambda \\ \frac{\alpha+1}{2}\lambda^2 & \text{if} \quad \theta > \alpha\lambda \end{cases}$$

This section explains in detail how PROC REGSELECT computes the reported regularization parameter ranges. The values of the grid points of α are $\{2.7, 3.7, 4.7, 5.7\}$ by default. In addition, the number of nonintercept parameters is $19-1 = 18$. Based on this information and the standard deviation of the response variable, the maximum and minimum values of λ are computed in the following way by using the formulas in the section "Folded Concave Penalized Selection" on page 89 in Chapter 2, "Shared Concepts":

$$\lambda_{\max} = y_{\text{std}} \times \sqrt{\frac{\log p}{p}} = 0.8837655 \times \sqrt{\frac{\log 18}{18}} = 0.354142$$

$$\lambda_{\min} = \frac{y_{\text{std}}}{20 \times \sqrt{\log p}} = \frac{0.8837655}{20 \times \sqrt{\log 18}} = 0.0104153$$

The default pattern of the λ grid is such that the values of the grid are evenly spaced on the logarithmic scale, based on the computed $\lambda_{\max}$ and $\lambda_{\min}$ values.

The "Selection Summary" table in Output 29.3.2 shows the values of α and λ together with the validation ASE and objective value at each step. The model that has the lowest validation ASE value is chosen as the final model.

Output 29.3.3 shows the details of this selected model, including fit statistics and parameter estimates. Notice that the model selected by the SCAD method has lower training and validation ASE values than the models selected by the LASSO method (Output 29.2.3) and the adaptive LASSO method (Output 29.2.6). This example demonstrates the advantages of nonconvex penalties over convex penalties.

Furthermore, you can find that for each fixed α, the number of effects increases as the value of λ decreases to zero.

Output 29.3.2 Summary of SCAD Selection with MILP Solver

Selection Summary

Step	Alpha	Lambda	Number of Effects	Validation ASE	Objective Value	Convergence Status
1	2.7	0.3541421718	3	0.6201	-519.8250989	Success
2	2.7	0.2393372071	6	0.4730	-242.594577	Success
3	2.7	0.161749442	5	0.3750	-120.7055356	Success
4	2.7	0.1093138936	5	0.3564	-68.35415937	Success
5	2.7	0.0738767763	6	0.3511	-44.77561769	Success
6	2.7	0.0499275791	7	0.3712	-34.64228789	Success
7	2.7	0.0337421756	8	0.3684	-30.37638497	Success
8	2.7	0.0228037176	10	0.3945	-28.63035462	Success
9	2.7	0.0154112628	12	**0.3387***	-28.05955057	Success
10	2.7	0.0104152764	14	0.3627	-27.90925291	Success
.	.	.	.	.	.	.
	.	.	.	.	.	.
	.	.	.	.	.	.
31	5.7	0.3541421718	3	0.6201	-939.7201363	Success
32	5.7	0.2393372071	6	0.4730	-434.3757135	Success
33	5.7	0.161749442	5	0.3913	-206.6854888	Success
34	5.7	0.1093138936	6	0.3592	-105.4374547	Success
35	5.7	0.0738767763	6	0.3521	-61.19314233	Success
36	5.7	0.0499275791	8	0.3493	-41.82085843	Success
37	5.7	0.0337421756	8	0.3564	-33.43914549	Success
38	5.7	0.0228037176	9	0.3634	-29.87208286	Success
39	5.7	0.0154112628	10	0.3909	-28.44965297	Success
40	5.7	0.0104152764	14	0.3644	-28.01750434	Success

*** Optimal Value of Criterion**

Reason
The model at Step 9, where Alpha = 2.7 and Lambda = 0.0154112628, is selected.
Selected model converged. All regularization parameters succeeded.

Label	Effects
Selected Effects:	Intercept Division_West nHits nHome nRuns nRBI nBB YrMajor CrAtBat CrBB nOuts nError

Output 29.3.3 Details of Model Selected by SCAD with MILP Solver

Details of the Model Selected by SCAD Method

Root MSE	0.60291
R-Square	0.57622
Adj R-Sq	0.53460
AIC	11.89325
AICC	15.20234
SBC	-80.26338
ASE (Train)	0.32832
ASE (Validate)	0.33873

Output 29.3.3 *continued*

Parameter Estimates		
Parameter	DF	Estimate
Intercept	1	4.252427
Division_West	1	-0.179169
nHits	1	0.007232
nHome	1	0.029747
nRuns	1	-0.007002
nRBI	1	-0.004489
nBB	1	0.014063
YrMajor	1	0.095316
CrAtBat	1	0.000109
CrBB	1	-0.000964
nOuts	1	0.000160
nError	1	-0.007590

You can also directly supply your own choice of regularization parameters to obtain a model fit. The following program produces an equivalent final model by taking the selected α and λ values from the previous analysis:

```
proc regselect data=mylib.baseball;
   partition roleVar = league(train='National' validate='American');
   class division;
   model logSalary = division nAtBat nHits nHome nRuns nRBI nBB
                     yrMajor crAtBat crHits crHome crRuns crRbi
                     crBB nOuts nAssts nError;
   selection method = scad(alpha=2.7  lambda=0.0154112628
                           solver=MILP  choose=validate);
run;
```

Suppose you have a good estimate of the regularization parameter α, and you want to use the nonlinear programming (NLP) solver instead of the default MILP solver to search for the best regularization parameter λ. You can perform SCAD selection in the following way:

```
proc regselect data=mylib.baseball;
       partition roleVar = league(train='National' validate='American');
       class division;
       model logSalary = division nAtBat nHits nHome nRuns nRBI nBB
                         yrMajor crAtBat crHits crHome crRuns crRbi
                         crBB nOuts nAssts nError;
       selection method = scad(alpha=2.7 solver=NLP choose=validate)
                 details=all;
run;
```

The "Selection Summary" table in Output 29.3.4 shows the optimal validation ASE as well as the value of the objective function for the SCAD selection with the NLP solver. Both values are larger than the ones that are produced by using the MILP solver, shown in Output 29.3.2. In general, the NLP solver is faster but more prone to return local minima because of possible nonconvexity of the reformulated objective function.

Output 29.3.4 Summary of SCAD Selection with NLP Solver

The REGSELECT Procedure

Selection Details

Selection Summary

Step	Alpha	Lambda	Number of Effects	Validation ASE	Objective Value	Convergence Status
1	2.7	0.3541421718	3	0.6201	-519.8250988	Success
2	2.7	0.2393372071	6	0.4730	-242.5945766	Success
3	2.7	0.161749442	5	0.3750	-120.7055354	Success
4	2.7	0.1093138936	8	0.3924	-67.45879634	Success
5	2.7	0.0738767763	8	**0.3503***	-44.63945938	Success
6	2.7	0.0499275791	6	0.3835	-34.43023651	Success
7	2.7	0.0337421756	10	0.3700	-30.11778983	Success
8	2.7	0.0228037176	14	0.3740	-28.04965108	Success
9	2.7	0.0154112628	13	0.3737	-27.90174756	Success
10	2.7	0.0104152764	16	0.3529	-27.83148385	Success

*** Optimal Value of Criterion**

The model at Step 5, where Alpha = 2.7 and Lambda = 0.0738767763, is selected.
Selected model converged. All regularization parameters succeeded.

Selected Effects: Intercept nAtBat nHome nBB YrMajor

Like SCAD selection, MCP selection also applies a nonconvex penalty to the penalized least squares objective function. The penalty function $P_{\lambda,\alpha}(\cdot)$ takes the following form:

$$P^{\text{mcp}}_{\lambda,\alpha}(\theta) = \begin{cases} \lambda\theta - \frac{1}{2\alpha}\theta^2 & \text{if} \quad 0 \le \theta \le \alpha\lambda \\ \frac{\alpha}{2}\lambda^2 & \text{if} \quad \theta > \alpha\lambda \end{cases}$$

The following statements perform MCP selection with a fixed α value of 1.7. The default solver is MILP.

```
proc regselect data=mylib.baseball;
       partition roleVar = league(train='National' validate='American');
       class division;
       model logSalary = division nAtBat nHits nHome nRuns nRBI nBB
                         yrMajor crAtBat crHits crHome crRuns crRbi
                         crBB nOuts nAssts nError;
       selection method = mcp(alpha=1.7 choose=validate) details=all;
run;
```

Output 29.3.6 shows that the model selected by MCP method also has lower training and validation ASE values than those produced by the LASSO method (Output 29.2.3) and the adaptive LASSO method (Output 29.2.6).

Output 29.3.5 Summary of MCP Selection with MILP Solver

The REGSELECT Procedure

Selection Details

Selection Summary

Step	Alpha	Lambda	Number of Effects	Validation ASE	Objective Value	Convergence Status
1	1.7	0.3541421718	2	0.4843	-4.780739689	Success
2	1.7	0.2393372071	3	0.3575	-12.4381079	Success
3	1.7	0.161749442	3	0.3566	-18.97905906	Success
4	1.7	0.1093138936	4	0.3847	-22.07263121	Success
5	1.7	0.0738767763	5	0.3775	-24.32243067	Success
6	1.7	0.0499275791	7	0.3711	-25.60457593	Success
7	1.7	0.0337421756	9	0.3980	-26.47039039	Success
8	1.7	0.0228037176	12	**0.3387***	-27.07526271	Success
9	1.7	0.0154112628	14	0.3627	-27.45939653	Success
10	1.7	0.0104152764	14	0.3627	-27.63619212	Success

*** Optimal Value of Criterion**

The model at Step 8, where Alpha = 1.7 and Lambda = 0.0228037176, is selected.
Selected model converged. All regularization parameters succeeded.

Selected Effects: Intercept Division_West nHits nHome nRuns nRBI nBB YrMajor CrAtBat CrBB nOuts nError

Output 29.3.6 Details of Model Selected by MCP with MILP Solver

The REGSELECT Procedure

Selected Model

Root MSE	0.60291
R-Square	0.57622
Adj R-Sq	0.53460
AIC	11.89325
AICC	15.20234
SBC	-80.26338
ASE (Train)	0.32832
ASE (Validate)	0.33873

Output 29.3.6 *continued*

Parameter Estimates		
Parameter	DF	Estimate
Intercept	1	4.252427
Division_West	1	-0.179169
nHits	1	0.007232
nHome	1	0.029747
nRuns	1	-0.007002
nRBI	1	-0.004489
nBB	1	0.014063
YrMajor	1	0.095316
CrAtBat	1	0.000109
CrBB	1	-0.000964
nOuts	1	0.000160
nError	1	-0.007590

For more information about the SCAD and MCP selection methods, see the section "Folded Concave Penalized Selection" on page 89 in Chapter 2, "Shared Concepts."

References

Akaike, H. (1969). "Fitting Autoregressive Models for Prediction." *Annals of the Institute of Statistical Mathematics* 21:243–247.

Burnham, K. P., and Anderson, D. R. (2002). *Model Selection and Multimodel Inference*. 2nd ed. New York: Springer-Verlag.

Collier Books (1987). *The 1987 Baseball Encyclopedia Update*. New York: Macmillan.

Darlington, R. B. (1968). "Multiple Regression in Psychological Research and Practice." *Psychological Bulletin* 69:161–182.

Draper, N. R., Guttman, I., and Kanemasu, H. (1971). "The Distribution of Certain Regression Statistics." *Biometrika* 58:295–298.

Fan, J., and Li, R. (2001). "Variable Selection via Nonconcave Penalized Likelihood and Its Oracle Properties." *Journal of the American Statistical Association* 96:1348–1360.

Furnival, G. M., and Wilson, R. W. (1974). "Regression by Leaps and Bounds." *Technometrics* 16:499–511.

Harrell, F. E., Jr. (2001). *Regression Modeling Strategies: With Applications to Linear Models, Logistic Regression, and Survival Analysis*. New York: Springer.

Hastie, T. J., Tibshirani, R. J., and Friedman, J. H. (2001). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York: Springer-Verlag.

Hocking, R. R. (1976). "The Analysis and Selection of Variables in a Linear Regression." *Biometrics* 32:1–50.

Hurvich, C. M., and Tsai, C.-L. (1989). "Regression and Time Series Model Selection in Small Samples." *Biometrika* 76:297–307.

Judge, G. G., Griffiths, W. E., Hill, R. C., Lütkepohl, H., and Lee, T.-C. (1985). *The Theory and Practice of Econometrics*. 2nd ed. New York: John Wiley & Sons.

Littell, R. C., Freund, R. J., and Spector, P. C. (1991). *SAS System for Linear Models*. 3rd ed. Cary, NC: SAS Institute Inc.

Mallows, C. L. (1973). "Some Comments on C_P." *Technometrics* 15:661–675.

Marquardt, D. W., and Snee, R. D. (1975). "Ridge Regression in Practice." *American Statistician* 29:3–20.

Muller, K. E., and Fetterman, B. A. (2002). *Regression and ANOVA: An Integrated Approach Using SAS Software*. Cary, NC: SAS Institute Inc.

Neter, J., Wasserman, W., and Kutner, M. H. (1990). *Applied Linear Statistical Models*. 3rd ed. Homewood, IL: Irwin.

Rawlings, J. O., Pantula, S. G., and Dickey, D. A. (1998). *Applied Regression Analysis: A Research Tool*. 2nd ed. New York: Springer-Verlag.

Schwarz, G. (1978). "Estimating the Dimension of a Model." *Annals of Statistics* 6:461–464.

Time Inc. (1987). "What They Make." *Sports Illustrated* (April 20): 54–81.

Zhang, C.-H. (2010). "Nearly Unbiased Variable Selection under Minimax Concave Penalty." *Annals of Statistics* 38:894–942.

Chapter 30
SANDWICH Procedure

Contents

Overview: SANDWICH Procedure **1554**
PROC SANDWICH Features 1554
PROC SANDWICH Compared with Other SAS Procedures 1555
Using CAS Sessions and CAS Engine Librefs 1556
Getting Started: SANDWICH Procedure **1557**
Syntax: SANDWICH Procedure **1561**
PROC SANDWICH Statement 1561
BY Statement 1562
CLASS Statement 1562
CLUSTER Statement 1563
DISPLAY Statement 1563
DISPLAYOUT Statement 1564
FREQ Statement 1565
MODEL Statement 1565
WEIGHT Statement 1566
Details: SANDWICH Procedure **1566**
Computational Details 1566
Notation 1566
Regression Coefficients 1567
Variance Estimation 1567
Computational Methods 1568
Distributed Computing and Multithreading 1568
Sparse Matrix Techniques 1569
Type III Tests 1570
Displayed Output 1570
Model Information 1570
Number of Observations 1570
Class Level Information 1570
Dimensions 1571
Fit Statistics 1571
ANOVA 1571
Parameter Estimates 1572
Type III Model ANOVA 1572
Timing 1572
OutputCasTables Table 1572
ODS Table Names 1573

Examples: SANDWICH Procedure **1573**
Example 30.1: Analysis of Microarray Data 1574
References **1576**

Overview: SANDWICH Procedure

The SANDWICH procedure analyzes linear models by using the classical robust sandwich variance estimator. This estimator is constructed by wrapping the inverse of the information matrix by the corrected sum of squares of the residuals. It is commonly used to protect against model misspecifications.

PROC SANDWICH uses algorithms that are specialized for distributed and multicore environments. It also takes advantage of sparse matrix techniques to reduce both memory and time requirements for problems that have classification effects with large numbers of levels. By taking advantage of distributed environments and sparse techniques, PROC SANDWICH can handle data that have large numbers of clusters, observations, and predictor variables.

PROC SANDWICH Features

The main features of the SANDWICH procedure are as follows:

- It computes the between-cluster variance-covariance matrix.
- It uses sparse matrix techniques in analytic computations.

Because the SANDWICH procedure runs on SAS Cloud Analytic Services (CAS; see the section "Using CAS Sessions and CAS Engine Librefs" on page 1556), it also does the following:

- enables you to run on a cluster of machines that distribute the data and the computations
- enables you to run in single-machine mode on CAS
- exploits all the available cores and concurrent threads. For information about how the SANDWICH procedure uses threads, see the section "Multithreading" on page 102 in Chapter 2, "Shared Concepts."

PROC SANDWICH Compared with Other SAS Procedures

The SANDWICH procedure in SAS Visual Statistics produces a robust covariance estimator for linear models, as do several procedures in SAS/STAT. These procedures are compared with PROC SANDWICH in Table 30.1.

Table 30.1 Robust Covariance Estimation Compared between Various SAS/STAT Procedures and PROC SANDWICH

Procedure	Procedure Overview	PROC SANDWICH Comparison
SURVEYREG	Performs regression analysis for sample survey data; provides a sandwich estimator in addition to several other estimators for the covariance matrix. Two important features of the complex survey sample designs—clustering and stratification—are accounted for in the sandwich estimator.	Addresses clustering in the computation of the sandwich estimator; is suitable for analyzing data that are clustered.
GENMOD	Fits generalized linear models and generalized estimating equations (GEE) models. The GEE approach extends independent estimating equations to correlated data. For correlated data, the GEE approach assumes a working correlation matrix from which the covariance matrix of the responses is constructed.	Assumes an independent correlation structure in estimating the weighted least squares regression coefficients, but accounts for correlation in estimating the covariance of these estimated coefficients.
GLIMMIX	Fits generalized linear mixed models; enables you to model the covariance structure of clustered data through random effects and residuals. Random effects, as well as the fixed effects and the parameters of the covariance structure, are estimated using the maximum likelihood method. PROC GLIMMIX computes the sandwich estimator after the mixed model is estimated. The estimation of a mixed model requires iterative optimization. It is a much more involved process than the estimation of a cluster-adjusted empirical covariance estimator.	Does not model the covariance structure of clustered data through random effects; accounts for the covariance structure while estimating the variance and covariance of the estimated parameters.

In addition, both the SANDWICH and LMIXED procedures in SAS Visual Statistics handle correlated data. These procedures are compared in Table 30.2.

Table 30.2 Correlated Data in the LMIXED and SANDWICH Procedures

Procedure	Procedure Overview	PROC SANDWICH Comparison
LMIXED	Fits linear mixed models to Gaussian data. The mean of the data and the variance and covariance of the data are modeled separately. The mean structure is a linear function of both fixed effects and random effects. The variance and covariance structure can arise from both marginal covariance and random effects. The inclusion of random effects and the variance and covariance parameters makes linear mixed models computationally more challenging. An iterative optimization algorithm is required in order to estimate the parameters.	Assumes an independent covariance structure and models the mean structure as a linear function of fixed effects; accounts for correlation in the computation of the covariance of the estimated fixed-effects coefficients. The estimation of fixed-effects coefficients and the robust variance and covariance structure does not require iterative optimization.

Using CAS Sessions and CAS Engine Librefs

SAS Cloud Analytic Services (CAS) is the analytic server and associated cloud services in SAS Viya. This section describes how to create a CAS session and set up a CAS engine libref that you can use to connect to the CAS session. It assumes that you have a CAS server already available; contact your system administrator if you need help starting and terminating a server. This CAS server is identified by specifying the host on which it runs and the port on which it listens for communications. To simplify your interactions with this CAS server, the host information and port information for the server are stored as SAS option values that are retrieved automatically whenever this CAS server needs to be accessed. You can examine the host and port values for the server at your site by using the following statements:

```
proc options option=(CASHOST CASPORT);
run;
```

In addition to starting a CAS server, your system administrator might also have created a CAS session and a CAS engine libref for your use. You can define your own sessions and CAS engine librefs that connect to the CAS server as shown in the following statements:

```
cas mysess;
libname mylib cas sessref=mysess;
```

The CAS statement creates the CAS session named mysess, and the LIBNAME statement creates the mylib CAS engine libref that you use to connect to this session. It is not necessary to explicitly name the CASHOST and CASPORT of the CAS server in the CAS statement, because these values are retrieved from the corresponding SAS option values.

If you have created the mysess session, you can terminate it by using the TERMINATE option in the CAS statement as follows:

```
cas mysess terminate;
```

For more information about the CAS and LIBNAME statements, see the section "Introduction to Shared Concepts" on page 6 in Chapter 2, "Shared Concepts."

Getting Started: SANDWICH Procedure

(View the complete code for this example at https://github.com/sascommunities/doc-supplement-casstat/tree/main/Examples/sandgs.sas.)

In health economics, large correlated data are common. These data usually contain information that is observed about large numbers of individuals, hospitals, or employers. The observations within individuals, hospitals, or employers are assumed to be correlated. There are two approaches to modeling such correlated data. One approach is to use random effects to introduce correlation in model estimation. Another approach is to use a robust variance estimator to adjust for correlation after the model is estimated. PROC SANDWICH provides a tool for analyzing large correlated data by using a robust variance estimator.

This example analyzes simulated health-care spending data by using a robust variance estimator. Suppose a number of health insurance choices are provided to 20,000 individuals each year over a period of four years. The goal of this analysis is to study the effect of choices of deductible and out-of-pocket payments on health care spending.

The following DATA step simulates 80,000 observations of annual health care spending from a linear model. The predictor variables are Individual, Year, Income, FamilySize, and choices of deductible and out-of-pocket payments for individual and family. DeductibleInd and DeductibleFam denote the five tiers of deductibles for an individual and a family, respectively. OutOfPocketInd and OutOfPocketFam denote the six tiers of out-of-pocket payments for an individual and a family, respectively. This DATA step assumes that your CAS engine libref is named mylib, but you can substitute any appropriately defined CAS engine libref.

NOTE: Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections "Using CAS Sessions and CAS Engine Librefs" on page 7 and "Loading a SAS Data Set onto a CAS Server" on page 7 in Chapter 2, "Shared Concepts."

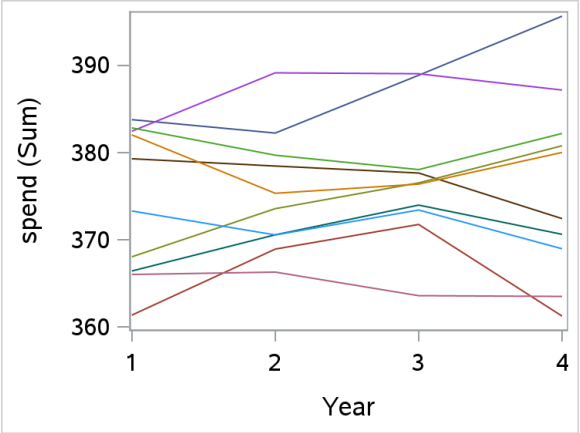
```
%let nDeductibleInd=5;
%let nDeductibleFam=5;
%let nOutOfPocketInd=6;
%let nOutOfPocketFam=6;
%let nFamSize=5;

data mylib.insure;
   call streaminit(1234);
   do Individual = 1 to 20000;
      rInd = 5 *rand("normal");
      Income = 3 + 2 *rand("normal");
      FamilySize = ceil(&nFamSize * rand("uniform"));
      do Year = 1 to 4;
         DeductibleInd = ceil(&nDeductibleInd * rand("uniform"));
```

```
            DeductibleFam = ceil(&nDeductibleFam * rand("uniform"));
            OutOfPocketInd= ceil(&nOutOfPocketInd * rand("uniform"));
            OutOfPocketFam= ceil(&nOutOfPocketFam * rand("uniform"));
            err   = 2*rand("normal");
            spend = 350 + Year + rInd + 3.2*Income + FamilySize +
                    DeductibleInd + DeductibleFam + OutOfPocketInd + OutOfPocketFam + err;
            output;
         end;
      end;
   run;
```

Figure 30.1 below shows the spaghetti plot of health-care spending of 10 individuals over the four-year period. You can see that the intercepts of the trajectories of health-care spending vary from individual to individual. The variability from individual to individual is an indication of correlation within individuals.

Figure 30.1 Spaghetti Plot of 10 Individuals



The following SANDWICH procedure statements fit a linear regression model to the annual health-care spending data:

```
ods select ModelInfo Dimensions ModelAnova;
proc sandwich data=mylib.insure sparse;
   class individual year familysize DeductibleInd DeductibleFam
         OutOfPocketInd OutOfPocketFam;
   clusters individual;
   model spend =  Year Income|DeductibleInd|OutOfPocketInd
                  Income|FamilySize|OutOfPocketFam|DeductibleFam/ss3;
run;
```

In the preceding statements, the cluster effect, Individual, accounts for correlations among observations from the same individual. The SPARSE option tells PROC SANDWICH to invoke sparse matrix methods. All two-way and the three-way interaction effects among Income, DeductibleInd, and OutOfPocketInd are included to study how the effects of individual deductible, out-of-pocket payment, and income change with each other. Also, all two-way, all three-way, and the four-way interaction effects among Income, FamilySize, OutOfPocketFam, DeductibleFam are included to account for the interaction between family deductible, family out-of-pocket payment, family size, and income.

These interaction effects in addition to the main effects add up to a total number of 590 parameters, leading to a sum of squares and crossproducts matrix $\mathbf{X}'\mathbf{X}$ that is of dimension 590×590. Analyzing these data in

PROC SURVEYREG requires the product $\mathbf{X}'\mathbf{X}$ for all 20,000 clusters. It takes about 2 GB of memory to store these matrices and a great deal of time to manipulate them. Such time-consuming and resource-draining problems can benefit from the parallel computing algorithms that are implemented in PROC SANDWICH.

Figure 30.2 shows the basic information of the model.

Figure 30.2 Model Information

The SANDWICH Procedure

Model Information	
Data Source	INSURE
Response Variable	spend
Design Matrix Method	Sparse

Figure 30.3 shows that there are 23 effects in this model and these effects have 590 parameters.

Figure 30.3 Model Dimensions

Dimensions	
Number of Effects	23
Number of Parameters	590
Number of Clusters	20000

Figure 30.4 shows estimates of some of the parameters.

Figure 30.4 Part of the Parameter Estimates Table

The SANDWICH Procedure

Parameter Estimates					
Parameter	**DF**	**Estimate**	**Standard Error**	**t Value**	**Pr > |t|**
Intercept	1	380.857116	0.435528	874.47	<.0001
Year 1	1	-2.992492	0.020482	-146.10	<.0001
Year 2	1	-2.040089	0.020680	-98.65	<.0001
Year 3	1	-0.994446	0.020661	-48.13	<.0001
Year 4	0	0	.	.	.
Income	1	3.257856	0.120709	26.99	<.0001
DeductibleInd 1	1	-4.121604	0.257734	-15.99	<.0001
DeductibleInd 2	1	-2.870825	0.261725	-10.97	<.0001
DeductibleInd 3	1	-1.794150	0.262846	-6.83	<.0001
DeductibleInd 4	1	-0.898897	0.261390	-3.44	0.0006
DeductibleInd 5	0	0	.	.	.
Income * DeductibleInd 1	1	0.028617	0.071237	0.40	0.6879
Income * DeductibleInd 2	1	-0.026624	0.070959	-0.38	0.7075
Income * DeductibleInd 3	1	-0.066286	0.072775	-0.91	0.3624
Income * DeductibleInd 4	1	-0.018352	0.071792	-0.26	0.7982
Income * DeductibleInd 5	0	0	.	.	.
	.	.	.	.	.
	.	.	.	.	.
	.	.	.	.	.

In the "Parameter Estimates" table, the *t* statistics of the parameters are computed using the sandwich variance estimator, which accounts for clustering. The degrees of freedom for the *t* test is the number of clusters minus one. The *t* test results of five individual deductibles categories indicate that they are significant. The estimates of these five individual categories show how annual health-care spending decreases with the increase of the individual deductible.

Part of the "Type III Model ANOVA" table is shown in Figure 30.5. By construction, the true model consists of the main effects Year, Income, FamilySize, DeductibleInd, DeductibleFam, OutOfPocketInd, and OutOfPocketFam. *F* test results in Figure 30.5 show that only the effects in the true model are significant. The *F* statistics of the parameters are computed using the sandwich variance estimator that accounts for clustering. The degrees of freedom for the *F* test is the number of clusters minus one.

Figure 30.5 Part of the Type III Model ANOVA Table

The SANDWICH Procedure

Type III Model ANOVA

Effect	DF	Sum of Squares	Mean Square	F Value	Pr > F
Year	3	23893	7964.17573	7964.18	<.0001
Income	1	32300	32300	32299.8	<.0001
DeductibleInd	4	1774.85622	443.71406	443.71	<.0001
Income*DeductibleInd	4	5.17970	1.29492	1.29	0.2694
OutOfPocketInd	5	2480.20898	496.04180	496.04	<.0001
Income*OutOfPocketInd	5	3.19300	0.63860	0.64	0.6703
DeductibleInd*OutOfPocketInd	20	17.04441	0.85222	0.85	0.6501
Income*DeductibleInd*OutOfPocketInd	20	27.23265	1.36163	1.36	0.1290
FamilySize	4	574.71260	143.67815	143.68	<.0001
	.	.	.	.	.
	.	.	.	.	.
	.	.	.	.	.

Syntax: SANDWICH Procedure

The following statements are available in the SANDWICH procedure:

PROC SANDWICH < *options* > **;**
BY *variables* **;**
CLASS *variable* < **(***options***)** >. . . < *variable* < **(***options***)** > > < / *global-options* > **;**
CLUSTER *variables* **;**
DISPLAY < *table-list* > < / *options* > **;**
DISPLAYOUT *table-spec-list* < / *options* > **;**
FREQ *variable* **;**
MODEL *dependent* **=** < *effects* > < / *model-options* > **;**
WEIGHT *variable* **;**

The PROC SANDWICH and MODEL statements are required.

If your model contains classification effects, the classification variables must be listed in the CLASS statement.

The following section describes the PROC SANDWICH statement and then describes the other statements in alphabetical order.

PROC SANDWICH Statement

PROC SANDWICH < *options* > **;**

The PROC SANDWICH statement invokes the SANDWICH procedure. Table 30.3 summarizes the options available in the PROC SANDWICH statement.

Table 30.3 PROC SANDWICH Statement Options

Option	Description
DATA=	Names the input data table that the SANDWICH procedure uses
SPARSE	Invokes sparse matrix techniques

You can specify the following *options* in the PROC SANDWICH statement.

DATA=*libref.data-table*
names the input data table for PROC SANDWICH to use. The default is the most recently created data table. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to the data, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 1556.

data-table specifies the name of the input data table.

SPARSE

invokes the sparse matrix techniques in the analytic computations. These techniques provide efficient ways to store sparse matrices and to solve sparse linear systems. For problems whose $\mathbf{X}'\mathbf{X}$ matrix is sparse, these techniques can reduce the time and memory requirements for solving the normal equation and computing the sandwich estimator.

BY Statement

BY *variables* **;**

You can specify a BY statement in PROC SANDWICH to obtain separate analyses of observations in groups that are defined by the values of the BY variables. If you specify more than one BY statement, only the last one specified is used. For more information, see the discussion of BY-group processing in the "Grouping Data" section of *SAS Programmers Guide: Essentials*.

CLASS Statement

CLASS *variable* <**(***options***)**>. . . <*variable* <**(***options***)**>> </ *global-options*> **;**

The CLASS statement names the classification variables to be used as explanatory variables in the analysis. These variables enter the analysis not through their values, but through levels to which the unique values are mapped.

Table 30.4 summarizes the values that you can use as either an *option* or a *global-option*. The options are fully documented in the section "CLASS Statement" on page 9 in Chapter 2, "Shared Concepts."

Table 30.4 CLASS Statement Options

Option	Description
DESCENDING	Reverses the sort order
MISSING	Treats missing values as valid levels
ORDER=	Specifies the sort order for the levels
PARAM=	Specifies the parameterization of the variable
REF=	Specifies the reference level of the variable

CLUSTER Statement

CLUSTER *variables* **;**

The CLUSTER statement specifies one or more *variables* in the DATA= input data table that identify the clusters. These variables must be specified as classification variables in the CLASS statement; they can be either character or numeric. The formatted values of the CLUSTER variables determine the CLUSTER variable levels. When you specify more than one variable in the CLUSTER statement, the combinations of categories of these variables define the clusters. You can use multiple CLUSTER statements to specify cluster variables. PROC SANDWICH uses variables from all CLUSTER statements to create clusters.

DISPLAY Statement

DISPLAY < *table-list* > < / *options* > **;**

The DISPLAY statement enables you to specify a list of display tables to display or exclude. This statement is similar to the ODS SELECT, ODS EXCLUDE, and ODS TRACE statements. However, the DISPLAY statement can improve performance when a large number of tables could be generated (such as in BY-group processing). The procedure processes the DISPLAY statement on a CAS server and thus sends only a subset of ODS tables to the SAS client. Because ODS statements are processed on a SAS client, first all the generated display tables are sent to the client, and then the client creates a subset.

If you use both DISPLAY and ODS statements together, the DISPLAY statement takes precedence over the ODS statements. Note that the ODS EXCLUDE statement processes tables that are sent to the client after they have been filtered by the DISPLAY statement. In some cases, it might appear that the ODS EXCLUDE statement is taking precedence because it can further filter the tables. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

You can specify the *table-list* as a list of table names, paths, partial pathnames, and regular expressions.

The table names that you can specify are listed in the section "ODS Table Names" on page 1573. A path is a table name that is prefixed with dot-separated grouping information. For example, a SelectionSummary table that a procedure produces during a selection routine might have the path *Bygroup1.Summary.SelectionSummary*. A partial pathname does not include all groups; for example, *SelectionSummary* and *Summary.SelectionSummary* are partial pathnames for *Bygroup1.Summary.SelectionSummary*.

When you specify a table name or partial pathname, all display tables whose paths end in the specified name are selected for display or exclusion. For example, both *SelectionSummary* and *Summary.SelectionSummary* select *Bygroup1.Summary.SelectionSummary*.

A regular expression is enclosed in forward slashes (/). For example, specifying "/tions/" selects all pathnames that contain the substring "tions"; in particular, the *Bygroup1.Summary.SelectionSummary* table is selected. Specifying "!/tions/" selects all pathnames that do not contain the substring "tions"; in particular, the *Bygroup1.Summary.SelectionSummary* table is not selected.

You can specify the following *options* after a slash (/):

CASESENSITIVE
performs a case-sensitive comparison of table names in the *table-list* to display table names when tables are subsetted for display. To preserve case, you must enclose table names in the *table-list* in quotation marks.

EXCLUDE
displays all display tables except those that you specify in the *table-list*.

EXCLUDEALL
suppresses display of all tables. This option takes precedence over the other options.

TRACE
displays the display table names, labels, and paths.

DISPLAYOUT Statement

DISPLAYOUT *table-spec-list* < / *options* > **;**

The DISPLAYOUT statement enables you to create output tables from your displayed output. This statement is similar to the ODS OUTPUT statement. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

The *table-spec-list* specifies a list of output tables to create. Each entry in the list has either a *key*=*value* format or a *key* format:

key=*value*	specifies *key* as the ODS table name, path, or partial pathname, and specifies *value* as the output table name.
key	specifies *key* as the ODS table name and also as the output table name.

The ODS table names that you can specify are listed in the section "ODS Table Names" on page 1573. You cannot specify the ODS table named OutputCasTables in the *table-spec-list*.

Table names and partial pathnames are discussed under the DISPLAY statement. The DISPLAYOUT statement does not support regular expressions.

You can specify the following *options* after a slash (/):

INCLUDEALL
creates output tables for all display tables. The name of the created output table is the same as the name of the corresponding display table. If you specify this option, the *table-spec-list* specification is ignored.

NOREPLACE
does not replace any existing output table of the same name.

REPEATED
replicates all output tables on all nodes.

FREQ Statement

FREQ *variable* **;**

The *variable* in the FREQ statement identifies a numeric variable in the input data table that contains the frequency of occurrence of each observation. PROC SANDWICH treats each observation as if it appears f times, where f is the value of the FREQ *variable* for the observation. If f is not an integer, it is truncated to an integer. If f is less than 1 or missing, the observation is not used in the analysis. When you omit the FREQ statement, each observation is assigned a frequency of 1. The FREQ statement is supported only when the CLUSTER statement is specified.

MODEL Statement

MODEL *dependent*=< *effects* > / < *model-options* > **;**

The MODEL statement names the dependent variable and the explanatory effects, including covariates, main effects, interactions, and nested effects.

You must specify the following argument:

dependent
specifies the dependent (response) variable.

You can also specify the following argument:

effects
specify the explanatory effects, including covariates, main effects, interactions, and nested effects. If you omit the explanatory effects, PROC SANDWICH fits an intercept-only model. For information about constructing the model effects, see the section "Specification and Parameterization of Model Effects" on page 64 in Chapter 2, "Shared Concepts."

You can also specify the following *model-options* after a slash (/); these *model-options* control other aspects of model formation and inference:

NOINT
suppresses the intercept term that is otherwise included in the model.

SS3
displays a "Model Anova" table that contains tests that are computed by using Type III sums of squares for each effect. For more information, see the section "Type III Tests" on page 1570.

WEIGHT Statement

WEIGHT *variable* **;**

The *variable* in the WEIGHT statement is used as a weight to perform a weighted analysis of the data. Observations that have nonpositive or missing weights are not included in the analysis. The values of the WEIGHT variable can be nonintegral and are not truncated. When you omit the WEIGHT statement, all observations that are used in the analysis are assigned a weight of 1.

Details: SANDWICH Procedure

Computational Details

PROC SANDWICH uses the Taylor series (linearization) method to estimate the variance-covariance matrix of estimators. This method is introduced in Huber (1967), White (1980), and MacKinnon and White (1985). It is also well discussed in the context of survey analysis. For more information see Fuller (1975), Särndal, Swensson, and Wretman (1992), Wolter (2007), and Rust (1985). The Taylor series method assumes an independent correlation structure in estimating the weighted least squares regression coefficients, but it accounts for correlations in estimating the covariance of the estimated coefficients. This section includes explicit formulas for estimating the regression coefficients and their covariance.

Notation

In PROC SANDWICH, the data are denoted as

$$(\mathbf{w}, \mathbf{y}, \mathbf{X}) = (w_{ij}, y_{ij}, \mathbf{x}_{ij})$$

where

- $\mathbf{w}$ denotes the weight vector
- $\mathbf{y}$ denotes the dependent variable
- $\mathbf{X}$ denotes the $N \times p$ design matrix
- $i = 1, 2, \ldots, n$ is the cluster index
- $j = 1, 2, \ldots, m_i$ is the unit index within cluster i
- $N = \sum_{i=1}^{n} m_i$

Regression Coefficients

PROC SANDWICH solves the normal equations $\mathbf{X}'\mathbf{W}\mathbf{X}\boldsymbol{\beta} = \mathbf{X}'\mathbf{W}\mathbf{y}$ by using a modified sweep routine that produces a generalized (g2) inverse $(\mathbf{X}'\mathbf{W}\mathbf{X})^{-}$ and a solution (Pringle and Rayner 1971),

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{W}\mathbf{X})^{-}\mathbf{X}'\mathbf{W}\mathbf{y}$$

where $\mathbf{W}$ is the diagonal matrix that is constructed from $\mathbf{w}$.

Because $\mathbf{X}'\mathbf{W}\mathbf{X}$ and $\mathbf{X}'\mathbf{W}\mathbf{y}$ can be decomposed into the corresponding matrices within each of the clusters

$$\mathbf{X}'\mathbf{W}\mathbf{X} = \sum_{i=1}^{n} \mathbf{X}_\mathbf{i}'\mathbf{W}_\mathbf{i}\mathbf{X}_\mathbf{i}$$

$$\mathbf{X}'\mathbf{W}\mathbf{y} = \sum_{i=1}^{n} \mathbf{X}_\mathbf{i}'\mathbf{W}_\mathbf{i}\mathbf{y}_\mathbf{i}$$

Because PROC SANDWICH shuffles the clusters to distinct threads, parallel processing can be leveraged to accumulate these matrices efficiently in parallel. Furthermore, for problems that have many classification effects, the components $\mathbf{X}_\mathbf{i}'\mathbf{W}_\mathbf{i}\mathbf{X}_\mathbf{i}$ and the overall $\mathbf{X}'\mathbf{W}\mathbf{X}$ could be handled as sparse matrices, making matrix storage and computation much more efficient.

Variance Estimation

Using the notation described in the section "Notation" on page 1566, the residuals from the linear regression can be written as

$$\mathbf{r} = \mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}$$

with r_{ij} as its elements. Let the $p \times p$ matrix $\mathbf{G}$ be defined as

$$\mathbf{G} = \frac{n}{n-1} \sum_{i=1}^{n} (\mathbf{e}_{i+} - \bar{\mathbf{e}}_{..})'(\mathbf{e}_{i+} - \bar{\mathbf{e}}_{..})$$

where

$$\begin{aligned} \mathbf{e}_{i+} &= \sum_{j=1}^{m_i} \mathbf{e}_{ij} \\ \mathbf{e}_{ij} &= w_{ij} r_{ij} \mathbf{x}_{ij} \\ \bar{\mathbf{e}}_{..} &= \frac{1}{n} \sum_{i=1}^{n} \mathbf{e}_{i+} \end{aligned}$$

If there is no CLUSTER statement, then the data can be denoted as

$$(\mathbf{w}, \mathbf{y}, \mathbf{X}) = (w_i, y_i, \mathbf{x}_i)$$

where $i = 1, 2, \ldots, N$ is the observation index. The $\mathbf{G}$ matrix becomes

$$\mathbf{G} = \frac{N}{N-1} \sum_{i=1}^{N} (\mathbf{e}_i - \bar{\mathbf{e}})'(\mathbf{e}_i - \bar{\mathbf{e}})$$

where

$$\begin{aligned} \mathbf{e}_i &= w_i r_i \mathbf{x}_i \\ \bar{\mathbf{e}} &= \frac{1}{N} \sum_{i=1}^{N} \mathbf{e}_i \end{aligned}$$

Given $\mathbf{G}$, the sandwich estimator of the covariance matrix of $\hat{\boldsymbol{\beta}}$ is

$$\widehat{\mathbf{V}}(\hat{\boldsymbol{\beta}}) = (\mathbf{X}'\mathbf{W}\mathbf{X})^{-}\mathbf{G}(\mathbf{X}'\mathbf{W}\mathbf{X})^{-}$$

In this formula, the first matrix factor, $(\mathbf{X}'\mathbf{W}\mathbf{X})^{-}$, would be an appropriate estimate for $\mathbf{V}(\hat{\boldsymbol{\beta}})$ if all observations were uncorrelated. Thus, the product of the last two factors effectively creates an adjustment that accounts for the within-cluster covariance.

Computational Methods

In order to perform linear regression with sandwich variance estimation efficiently, PROC SANDWICH uses distributed computing and sparse matrix techniques. The following two subsections discuss these techniques in detail.

Distributed Computing and Multithreading

Distributed computing refers to the use of multiple autonomous computers that communicate through a secure network. Distributed computing solves computational problems by dividing them into many tasks, each of which is solved by one or more computers. Multithreading divides the tasks on each computer so they can be completed in parallel by multiple threads.

PROC SANDWICH uses the following distributed computing and multithreading techniques:

- accumulating the components of weighted normal equations $(\mathbf{X}'\mathbf{W}\mathbf{X})$ and $(\mathbf{X}'\mathbf{W}\mathbf{Y})$
- accumulating the components of the within-cluster residual covariance $\mathbf{G}$
- solving the weighted normal equations
- computing the sandwich variance-covariance estimate

Regular weighted least squares regression depends on matrix quantities such as $\mathbf{X}'\mathbf{W}\mathbf{X}$ and $\mathbf{X}'\mathbf{W}\mathbf{y}$, which can be accumulated in any order and in parallel. Because clusters are independent of each other, these quantities can be processed in parallel by threads across the nodes. Hence, to efficiently compute the cluster components of the matrix $\mathbf{G}$ that forms the "meat" of the sandwich variance estimate, correlated observations

within the same cluster should be processed together. PROC SANDWICH shuffles input data to ensure that observations from the same cluster are on the same node so that they can be assigned to the same thread.

PROC SANDWICH divides the following computational tasks by clusters:

- Using the following decomposition, PROC SANDWICH assigns the task of computing $\mathbf{X_i'W_iX_i}$ and $\mathbf{X_i'W_iy_i}$ to the thread where the observations for cluster i are located. These pieces are then accumulated to give the overall $\mathbf{X'WX}$ and $\mathbf{X'Wy}$.

$$\mathbf{X'WX} = \sum_{i=1}^{n} \mathbf{X_i'W_iX_i}$$

$$\mathbf{X'Wy} = \sum_{i=1}^{n} \mathbf{X_i'W_iy_i}$$

- Similarly, the cluster components of matrix $\mathbf{G}$, $\mathbf{e}_{i\cdot}$, are computed in parallel and then accumulated.

In addition, operations such as sweeping and matrix solving are multithreaded to realize performance benefits for matrices of sufficient sizes.

Distributed computing enables PROC SANDWICH to handle large number of clusters that possibly have large number of observations.

Sparse Matrix Techniques

PROC SANDWICH uses the following sparse matrix techniques:

- storing and accumulating $(\mathbf{X'WX})$ and $(\mathbf{X'WY})$
- obtaining $\hat{\boldsymbol{\beta}}$ and the sandwich estimate $\widehat{\mathbf{V}}(\hat{\boldsymbol{\beta}})$ through sparse Cholesky solvers

A common component in the formula for $\hat{\boldsymbol{\beta}}$ and $\widehat{\mathbf{V}}(\hat{\boldsymbol{\beta}})$ is the matrix product $\mathbf{X'WX}$. The size of this product increases quadratically with the row dimension of $\mathbf{X}$. However, $(\mathbf{X'WX})$ would be relatively sparse if $\mathbf{X}$ is sparse. In such a case, PROC SANDWICH stores both of them in sparse format and computes $(\mathbf{X'WX})$ accordingly.

Furthermore, instead of computing the inversion of $\mathbf{X'WX}$ directly, PROC SANDWICH applies sparse Cholesky solvers to compute $(\mathbf{X'WX})^{-}\mathbf{X'Wy}$ and $(\mathbf{X'WX})^{-}\mathbf{G}(\mathbf{X'WX})^{-}$. In general, for a positive definite matrix $\mathbf{A}$, you can solve $\mathbf{Ax} = \mathbf{b}$ for $\mathbf{x}$ by first performing the Cholesky decomposition $\mathbf{A} = \mathbf{LL}'$, and then applying forward and backward substitution. For general matrices $\mathbf{A}$, the computational cost of this approach is still of order n^3, which is no better than solving $\mathbf{Ax} = \mathbf{b}$ directly. However, if $\mathbf{A}$ is sparse, $\mathbf{L}$ is often sparse too, and forward and backward substitution have much lower cost than of order n^3.

Type III Tests

Linear hypotheses for $\boldsymbol{\beta}$ are expressed in matrix form as

$$H_0: \mathbf{L}\boldsymbol{\beta} = \mathbf{c}$$

where $\mathbf{L}$ is a matrix of coefficients for the linear hypotheses and $\mathbf{c}$ is a vector of constants. The vector of regression coefficients $\boldsymbol{\beta}$ includes slope parameters and intercept parameters. The F statistic for testing H_0 is computed as

$$F = (\mathbf{L}\widehat{\boldsymbol{\beta}} - \mathbf{c})'[\mathbf{L}\widehat{\mathbf{V}}(\widehat{\boldsymbol{\beta}})\mathbf{L}']^{-1}(\mathbf{L}\widehat{\boldsymbol{\beta}} - \mathbf{c})/r$$

where $\widehat{\mathbf{V}}(\widehat{\boldsymbol{\beta}})$ is the robust covariance estimator of $\widehat{\boldsymbol{\beta}}$ and r is the rank of $\mathbf{L}\widehat{\mathbf{V}}(\widehat{\boldsymbol{\beta}})\mathbf{L}'$. Under H_0, it is common to assume that F follows an F distribution with the numerator degrees of freedom equal to r and the denominator degrees of freedom equal to the number of clusters minus one. If there is no CLUSTER statement, then the denominator degrees of freedom is equal to the number of observations minus the rank of the design matrix. For more information about Type III estimable functions, see the chapter "The GLM Procedure" and the section "The Four Types of Estimable Functions" in *SAS/STAT User's Guide*. Also see Littell, Freund, and Spector (1991).

Displayed Output

The following sections describe the output that the SANDWICH procedure produces. The output is organized into various tables, which are described in the order of their appearance.

Model Information

The "Model Information" table displays basic information about the model, including the data source, the response variable, and the design matrix method.

Number of Observations

The "Number of Observations" table displays the number of observations that are read from the input data table and the number of observations that are used in the analysis. If you specify a FREQ statement, this table also displays the sum of frequencies that are read and used.

Class Level Information

The "Class Level Information" table lists the levels of each variable that is specified in the CLASS statement. You can adjust the order of the CLASS variable levels by specifying the ORDER= option in the CLASS statement.

Dimensions

The "Dimensions" table displays the number of effects, the number of parameters, and the number of clusters in the model. This table can be useful in determining the requirements for CPU time and memory.

Fit Statistics

The "Fit Statistics" table displays fit statistics for the model. These statistics include the following:

- Root MSE, an estimate of the standard deviation of the error term. It is calculated as the square root of the mean square error.
- R-square, a measure between 0 and 1 that indicates the portion of the (corrected) total variation that is attributed to the fit rather than left to residual error. It is calculated as SS(Model) divided by SS(Total) and is also called the *coefficient of determination*. It is the square of the multiple correlation—in other words, the square of the correlation between the dependent variable and the predicted values.
- Adj R-Sq, the adjusted R-square, which is a version of R-square that has been adjusted for the number of parameters in the model. It is calculated as

$$\bar{R}^2 = 1 - \frac{(n-i)(1-R^2)}{n-p}$$

  where i is equal to 1 if there is an intercept and 0 otherwise, n is the number of observations that are used to fit the model, and p is the number of parameters in the model.

ANOVA

The "ANOVA" table displays the analysis of variance for the model. This table includes the following:

- the Source of the variation, Model for the fitted regression, Error for the residual error, and C Total for the total variation after correcting for the mean.
- the degrees of freedom (DF) that are associated with the source
- the Sum of Squares for the term
- the Mean Square, which is the sum of squares divided by the degrees of freedom
- the F Value for testing the hypothesis that all parameters are 0 except for the intercept. This is formed by dividing the mean square for Model by the mean square for Error.
- the Prob>F, which is the probability of getting a greater F statistic than that observed if the hypothesis is true. The degrees of freedom for the F test is the number of clusters minus one. If there is no CLUSTER statement, the degrees of freedom is the number of observations minus the rank of the design matrix.

Parameter Estimates

The "Parameter Estimates" table displays the parameters in the model and their estimates. The information displayed for each parameter in the model includes the following:

- the parameter label, which includes the effect name and level information for effects that contain classification variables
- the degrees of freedom (DF) for the parameter. There is usually one degree of freedom, except when the model is not of full rank.
- the parameter estimate
- the standard error of the parameter estimate
- t Value, which is computed as the parameter estimate divided by the standard error.
- the Pr > |t|, which the probability of getting a t statistic that has a greater absolute value than that observed if the true parameter is 0. The degrees of freedom for the t test is the number of clusters minus one. If there is no CLUSTER statement, the degrees of freedom is the number of observations minus the rank of the design matrix. This is the two-tailed significance probability.

Type III Model ANOVA

When you specify the SS3 option in the MODEL statement, the SANDWICH procedure produces the "Type III Model ANOVA" table. This table displays tests that all parameters for a particular effect are equal to 0. These tests are computed by using a Type III sum of squares, which yields a balanced test of each effect, adjusted for every other effect. For more information, see the section "Type III Tests" on page 1570.

Timing

The "Timing" table displays the amount of time (in seconds) and the percentage of the time that PROC SANDWICH required to perform different tasks in the analysis.

OutputCasTables Table

The OutputCasTables table is a special table that has information about each CAS table that is created during a CAS action execution. The information for each CAS table consists of the CAS table name, the caslib in which the table resides, and the number of columns and rows in the CAS table. Because this table is not a typical ODS table that contains analytical results, you cannot include it in the *table-spec-list* in the DISPLAYOUT statement.

ODS Table Names

Each table that the SANDWICH procedure creates has a name associated with it. You must use this name to refer to the table when you use the DISPLAY statement, the DISPLAYOUT statement, or ODS statements. These names are listed in Table 30.5.

Table 30.5 ODS Tables Produced by PROC SANDWICH

Table Name	Description	Required Statement / Option
ANOVA	Model ANOVA table	Default output
ClassInfo	Level information from the CLASS statement	CLASS
Dimensions	Model dimensions	Default output
FitStatistics	Fit statistics	Default output
ModelAnova	Type III Model ANOVA	MODEL / SS3
ModelInfo	Model information	Default output
NObs	Number of observations that are read and used	Default output
OutputCasTables	See the section "OutputCasTables Table" on page 1572	DISPLAYOUT
ParameterEstimates	Solutions for the parameter estimates that are associated with effects in the MODEL statement	Default output
Timing	Timing breakdown by task	Default output

Examples: SANDWICH Procedure

The examples in this chapter are available in the GitHub repository located at `https://github.com/sascommunities/doc-supplement-casstat`.

Example 30.1: Analysis of Microarray Data

(View the complete code for this example (sandex1.sas) in the example repository.)

Gene expression profiling is a technology that monitors differences in gene expression levels. It can be used to discover relationships between gene expression levels and a condition or treatment. Microarray experiments allow simultaneous gene expression profiling of hundreds to thousands of genes. In a microarray experiment, a collection of microscopic DNA spots are attached to a glass slide or silicon cassette. Each DNA spot contains a high density of a specific DNA sequence or gene. These genes are then bound with a mixture of cDNA samples from an experimental group and a reference group. Each group of cDNA samples are labeled with a fluorescent dye. You can measure the intensity of fluorescence to quantify the amount of binding of DNA molecules with cDNA samples from each group. The analysis of microarray data enables you to study genomic response to a particular condition or treatment.

PROC SANDWICH is a suitable tool for the analysis of microarray data. You can specify the microarray variable as the cluster effect to accommodate correlations among the observations from the same array. You can also take advantage of sparse techniques to handle the large number of genes on each array.

The following SAS statements simulate a experiment that includes six arrays:

```
%let nArray  = 6;
%let nDye    = 2;
%let nGene   = 500;
%let nTrt    = 2;

data mylib.microarray;
   keep  Microarray Array Gene Trt Dye Density;
   array GeneDist{&ngene};
   array ArrayEffect{&narray};
   do i = 1 to &nGene;
       GeneDist{i} = 1 + int(&nGene*ranuni(12345));
   end;
   do i = 1 to &nArray;
      ArrayEffect{i} = sqrt(0.014)*rannor(12345);
   end;

   do Microarray = 1 to &nArray;
      do Dye = 1 to &nDye;
         do k = 1 to &nGene;
            Trt   = 1 + int(&nTrt*ranuni(12345));
            Gene  = GeneDist{k};
            Array = ArrayEffect{Microarray};

            Density  = 1 +
                     + Dye
                     + Trt
                     + Gene/1000.0
                     + Gene*Dye/1000.0
                     + Gene*Trt/1000.0
                     + Array
                     + sqrt(0.02)*rannor(12345);
            output;
```

```
         end;
      end;
   end;
run;
```

In these simulated data, each of the six arrays is spotted with the same 500 genes. The cDNA samples are labeled with two dyes, one for the treatment group and one for the reference group. Littell et al. (2006) suggests using gene and dye interaction, Gene*Dye, to control for the fact that dyes incorporate into different target cDNA with different efficiency. Also, it is important to use the gene and treatment interaction, Gene*Trt, to account for the fact that the response to treatment might vary with genes. The cluster effect, Microarray, controls the fact that observations from the same microarray are correlated. With 500 genes, the gene effect and these two interaction effects account for 1,500 parameters of the total 1,600 parameters. A sparse problem of this size can benefit from the sparse techniques that PROC SANDWICH implements.

The following SANDWICH procedure fits a linear regression to the simulated data. The SPARSE option in the PROC statement requests that PROC SANDWICH use sparse matrix techniques.

```
ods select Dimensions ModelAnova ModelInfo;
proc sandwich data=mylib.microarray sparse;
   class Microarray Gene Trt Dye;
   model Density = Gene Trt Dye Gene*Trt Gene*Dye/ss3;
   cluster Microarray;
run;
```

The "Model Information" table (Output 30.1.1) shows that the sparse method was used in the model estimation.

Output 30.1.1 Model Information

The SANDWICH Procedure

Model Information	
Data Source	MICROARRAY
Response Variable	Density
Design Matrix Method	Sparse

The "Dimensions" table (Output 30.1.2) shows that this is a sparse problem with 6 effects, 6 clusters, and 1,600 parameters.

Output 30.1.2 Dimensions

Dimensions	
Number of Effects	6
Number of Parameters	1600
Number of Clusters	6

The "Type III Model ANOVA" table (Output 30.1.3) shows the Type III analysis of model effects. The degrees of freedom for the F test is the number of clusters minus one. The covariance of the estimates of model effects are adjusted for the correlation among the measurements from the same array.

Output 30.1.3 Type III Model ANOVA

Type III Model ANOVA

Effect	DF	Sum of Squares	Mean Square	F Value	Pr > F
Gene	5	3576.51442	715.30288	715.30	<.0001
Trt	1	248839	248839	248839	<.0001
Dye	1	166548	166548	166548	<.0001
Gene*Trt	5	256.35404	51.27081	51.27	0.0003
Gene*Dye	5	228.99388	45.79878	45.80	0.0004

The denominator degrees of freedom for the F tests is 5.

References

Fuller, W. A. (1975). "Regression Analysis for Sample Survey." *Sankhyā, Series C* 37:117–132.

Huber, P. J. (1967). "The Behavior of Maximum Likelihood Estimates under Nonstandard Conditions." *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability* 1:221–233.

Littell, R. C., Freund, R. J., and Spector, P. C. (1991). *SAS System for Linear Models*. 3rd ed. Cary, NC: SAS Institute Inc.

Littell, R. C., Milliken, G. A., Stroup, W. W., Wolfinger, R. D., and Schabenberger, O. (2006). *SAS for Mixed Models*. 2nd ed. Cary, NC: SAS Institute Inc.

MacKinnon, J. G., and White, H. (1985). "Some Heteroskedasticity-Consistent Covariance Matrix Estimators with Improved Finite Sample Properties." *Journal of Econometrics* 29:305–325.

Pringle, R. M., and Rayner, A. A. (1971). *Generalized Inverse Matrices with Applications to Statistics*. New York: Hafner Publishing.

Rust, K. (1985). "Variance Estimation for Complex Estimators in Sample Surveys." *Journal of Official Statistics* 1:381–397.

Särndal, C.-E., Swensson, B., and Wretman, J. (1992). *Model Assisted Survey Sampling*. New York: Springer-Verlag.

White, H. (1980). "A Heteroskedasticity-Consistent Covariance Matrix Estimator and a Direct Test for Heteroskedasticity." *Econometrica* 48:817–838.

Wolter, K. M. (2007). *Introduction to Variance Estimation*. 2nd ed. New York: Springer.

Chapter 31
SIMSYSTEM Procedure

Contents

Overview: SIMSYSTEM Procedure **1578**
- Simulation from Probability Distributions 1578
- Simulation from Families of Distributions 1579
- Simulation from the Pearson and Johnson Systems of Distributions 1580
- Benefits of the SIMSYSTEM Procedure 1580
- Features of the SIMSYSTEM Procedure 1581
- Using CAS Sessions and CAS Engine Librefs 1581

Getting Started: SIMSYSTEM Procedure **1582**
- ANOVA for Nonnormal Data 1582

Syntax: SIMSYSTEM Procedure **1590**
- PROC SIMSYSTEM Statement 1590
- MOMENTGRID Statement 1594
- MOMENTS Statement 1595
- OUTPUT Statement 1596

Details: SIMSYSTEM Procedure **1597**
- Moments and Moment Ratios 1597
- Definitions and Interpretations of Skewness and Kurtosis 1598
- Moment Ratio Coverage 1599
- Pearson System of Distributions 1600
- Johnson System of Distributions 1607
- Specifying Boundary Widths for Transition Families 1610
- Shapes of Pearson and Johnson Density Functions 1611
- Output Data Table 1614
- Displayed Output 1615
- ODS Table Names 1615
- ODS Graphics 1615

Examples: SIMSYSTEM Procedure **1616**
- Example 31.1: Estimating the Parameters of Johnson Distributions 1616
- Example 31.2: Exploring the Coverage of Three-Sigma Limits 1619
- Example 31.3: Simulating Portfolio Values 1623
- Example 31.4: Simulating Estimation of Value at Risk 1629
- Example 31.5: Interpreting Distribution Parameters 1633

References **1639**

Overview: SIMSYSTEM Procedure

The SIMSYSTEM procedure simulates continuous univariate data from distributions in the Pearson and Johnson systems. You request these distributions by directly specifying their measures of skewness and kurtosis rather than their distribution names and mathematical parameters. This makes it convenient to simulate data from a series of distributions whose degrees of asymmetry and tail weight vary over a grid of skewness-kurtosis combinations.

A property of the Pearson and Johnson systems is that there is a unique distribution in each system for any feasible skewness-kurtosis combination. This generality, together with the convenience of specifying skewness and kurtosis, enables you to design simulation studies that have a greater variety of distributional shapes and to reach conclusions that are more reliable than is possible with smaller families of distributions—such as the lognormal and Weibull—that are traditionally used in simulation.

Simulation from Probability Distributions

Simulation is a powerful method for understanding and predicting the behavior of a complex system under different conditions when it is not feasible to experiment with the system and measure how the output changes in response to variation in the inputs. See Ross (2006) for an introduction to the design and analysis of simulation studies.

Probabilistic simulation—often referred to as Monte Carlo simulation—is based on a probability model that assumes distributions for the input variables and assumes a functional relationship between the input variables and the output. By sampling a large number of values from the input distributions, you can simulate the distribution of the output and compute probabilities that quantify uncertainty in the performance of the system.

Simulation based on probability distributions is essential for assessing and managing risk in many fields of business, government, and research where actual data for unusual events are limited or unavailable. These fields include banking (market and credit risk), insurance (actuarial risk), fraud prevention, cybersecurity risk assessment, reliability engineering, national defense, climate modeling, environmental analysis, and epidemiology. In the field of statistics, simulation is a standard approach for evaluating the performance and robustness of statistical methods when applied to data that cannot be represented by standard modeling distributions such as the normal distribution.

Simulation from Families of Distributions

Because the actual distribution of an input variable is generally unknown, the reliability of a simulation study rests on sampling from distributions that represent a broad set of realistic scenarios. Well-designed simulation studies sample from a range of parametric distributions that have different means, standard deviations, and shapes. In studies of this type, you typically select several families of distributions to sample from, and then you specify the mathematical parameters for particular distributions within each family.

For example, if your goal is to model the effects of nonnormality on a statistical method, you might simulate data from various lognormal distributions to allow for different degrees of skewness, and you might simulate data from various *t* distributions to allow for distribution tails that are heavier than those of the normal distribution. The following statements illustrate how you carry this out with the RAND function in the DATA step:

```
data SimNonNormal(drop=i);
   call streaminit(15531);
   do i=1 to 10000;
      normal        = rand('NORMAL');
      t_4           = rand('T', 4);
      t_10          = rand('T', 10);
      lognormal_0_2 = rand('LOGNORMAL', 0, 0.2);
      lognormal_0_4 = rand('LOGNORMAL', 0, 0.4);
      output;
   end;
run;
```

Note that for each distribution in the *t* family you must specify its degrees of freedom (in this case, the values 4 and 10). Likewise, for each distribution in the lognormal family you must specify a shape parameter σ (in this case, the values 0.2 and 0.4).

When using the RAND function you must choose both a distribution family and the mathematical parameters for the distributions in that family. This approach has three major disadvantages:

- The choice of a family is generally not obvious. In some applications the choice is well established; for example, the lognormal and Weibull families are commonly chosen for simulating lifetime distributions in reliability analysis. However, in most applications the family is chosen in an ad hoc manner.
- The parameter choices are nonintuitive. Generally, the most useful descriptors of a distribution—in particular, its mean, standard deviation, skewness, and kurtosis—are complicated mathematical functions of the parameters. For example, the skewness of a lognormal distribution with parameter σ is $\sqrt{\beta_1} = \sqrt{\exp(\sigma^2) - 1}(\exp(\sigma^2) + 2)$. This makes it difficult to simulate data by specifying the descriptors of this distribution.
- It is computationally challenging to simulate data from distributions with a broad combinations of skewness and kurtosis because the families available in the RAND function—and distribution families in general—cover only a subset of the combinations that are feasible.

You can avoid these disadvantages by using the SIMSYSTEM procedure.

Simulation from the Pearson and Johnson Systems of Distributions

In contrast to the traditional approach of simulating from families of distributions, the SIMSYSTEM procedure simulates values from the Pearson system of distributions (Pearson 1895, 1901, 1916) and the Johnson system of distributions (Johnson 1949). You request a range of distributions simply by specifying combinations of skewness and kurtosis, also referred to as moment ratios, rather than names and parameters of distributions, as illustrated in the following statements:

```
proc simsystem system=pearson n=10000 seed=15531;
   moments skewness = 0  0  0  0.5  1.0
           kurtosis = 3  2  4  3.5  4.5;
   output out=mylib.MySimulation;
run;
```

In this example, the procedure simulates 10,000 values from each of five Pearson distributions. The first distribution, with a skewness of 0 and a kurtosis of 3, is the normal distribution. The next two distributions are also symmetric (with a skewness of 0) but differ from the normal distribution in their values of kurtosis. The final two distributions are skewed (with skewness values of 0.5 and 1.0) and have different values of kurtosis (3.5 and 4.5).

Benefits of the SIMSYSTEM Procedure

The SIMSYSTEM procedure provides three major benefits in practice:

- You can request a range of distributions just by specifying their combinations of mean, standard deviation, skewness, and kurtosis. These descriptors control the location, scale, and shapes of the distributions—aspects that are intuitive to understand and can be visualized in graphical displays created by the procedure.
- You can specify any feasible combination of skewness and kurtosis, because there is a unique distribution in each system whose theoretical skewness and kurtosis match that combination. By comparison, families of better-known distributions, which are smaller than these systems, cover only a limited set of combinations.
- By specifying a grid of evenly spaced combinations of skewness and kurtosis, you can explore their effects more thoroughly and form conclusions that are more general than those you would obtain from an ad hoc selection of distributions.

Features of the SIMSYSTEM Procedure

Key features of the SIMSYSTEM procedure are as follows:

- The procedure provides three convenient ways to specify combinations of mean, standard deviation, skewness, and kurtosis for the distributions to be sampled: the MOMENTS statement, the MOMENTGRID statement, and the DATA= option in the PROC SIMSYSTEM statement.
- The procedure produces summary tables and graphical displays for the distributions that you specify. When a specified Pearson distribution also happens to be a well-known parametric distribution, such as a beta distribution or an exponential distribution, the summary table identifies the well-known distribution in addition to the Pearson type.
- The procedure produces moment-ratio diagrams, which map the specified skewness and kurtosis values in the skewness-kurtosis plane.
- The procedure produces summary tables and histograms for the values that are sampled from each specified distribution.
- The procedure provides the ability to create replicate samples.
- The procedure creates a data table in which to save the values that are sampled from specified distributions.

Using CAS Sessions and CAS Engine Librefs

SAS Cloud Analytic Services (CAS) is the analytic server and associated cloud services in SAS Viya. This section describes how to create a CAS session and set up a CAS engine libref that you can use to connect to the CAS session. It assumes that you have a CAS server already available; contact your system administrator if you need help starting and terminating a server. This CAS server is identified by specifying the host on which it runs and the port on which it listens for communications. To simplify your interactions with this CAS server, the host information and port information for the server are stored as SAS option values that are retrieved automatically whenever this CAS server needs to be accessed. You can examine the host and port values for the server at your site by using the following statements:

```
proc options option=(CASHOST CASPORT);
run;
```

In addition to starting a CAS server, your system administrator might also have created a CAS session and a CAS engine libref for your use. You can define your own sessions and CAS engine librefs that connect to the CAS server as shown in the following statements:

```
cas mysess;
libname mylib cas sessref=mysess;
```

The CAS statement creates the CAS session named mysess, and the LIBNAME statement creates the mylib CAS engine libref that you use to connect to this session. It is not necessary to explicitly name the

CASHOST and CASPORT of the CAS server in the CAS statement, because these values are retrieved from the corresponding SAS option values.

If you have created the mysess session, you can terminate it by using the TERMINATE option in the CAS statement as follows:

```
cas mysess terminate;
```

For more information about the CAS and LIBNAME statements, see the section "Introduction to Shared Concepts" on page 6 in Chapter 2, "Shared Concepts."

Getting Started: SIMSYSTEM Procedure

ANOVA for Nonnormal Data

(View the complete code for this example at https://github.com/sascommunities/doc-supplement-casstat/tree/main/Examples/ssy_gs.sas.)

This example demonstrates a basic application of the SIMSYSTEM procedure in which the MOMENTS statement is used to simulate samples from a small number of nonnormal distributions with specified skewness (denoted by $\sqrt{\beta_1}$) and kurtosis (denoted by β_2); see "Moments and Moment Ratios" on page 1597. The example concludes by explaining the use of the MOMENTGRID statement to simulate samples from a large number of nonnormal distributions whose skewness and kurtosis, also referred to as moment ratios, vary evenly over a grid.

An industrial engineer in a manufacturing facility is investigating the effect of five operators with different levels of training on a critical functional parameter of a product. The engineer obtains 100 measurements of the parameter, 20 for each operator. A one-way analysis of variance (ANOVA) seems like an appropriate method for testing the hypothesis of no training effect, because if there is an effect the level of training would shift the expected (mean) value of the parameter.

However, it cannot be assumed that the measurements are normally distributed. The distribution of measurements taken previously was found to be skewed and heavy-tailed, with a skewness of 1.5 and a kurtosis of 7. There is concern that ANOVA F tests might not be valid for this type of data. You can use PROC SIMSYSTEM to simulate data with this particular combination of skewness and kurtosis, and with combinations that depart from the observed combination (including a skewness of 0 and a kurtosis of 3, which correspond to the normal distribution). You can then use the simulated data to study the behavior of ANOVA tests.

To start, consider how to simulate data from a single distribution with the observed combination of skewness and kurtosis. The following statements use PROC SIMSYSTEM to simulate 100 values from the Pearson distribution that has mean 0, standard deviation 1, skewness 1.5, and kurtosis 7:

```
ods graphics on;
proc simsystem system=pearson n=100 seed=12345;
   moments skewness=1.5 kurtosis=7;
   output out=mylib.sim;
run;
```

The SYSTEM= option specifies the system of distributions to sample from. The N= option specifies the number of simulated values, and the SEED= option specifies the seed for the random number generator in order to make the results reproducible. The MOMENTS statement specifies the values of skewness and kurtosis. Note that skewness and kurtosis are defined somewhat differently for theoretical distribution in the statistics literature and in SAS; see "Definitions and Interpretations of Skewness and Kurtosis" on page 1598 for the definitions used by the SIMSYSTEM procedure. Because no values are specified for the mean and standard deviation, default values of 0 and 1 are used, respectively. The OUTPUT statement requests a CAS table named SIM that contains the simulated values.

Figure 31.1 Pearson Distribution with Specified Skewness and Kurtosis

The SIMSYSTEM Procedure

Simulation Information	
System of Distributions	Pearson
Moment Combinations Specified	1
Valid Moment Combinations	1
Sample Size	100
Random Seed	12345

Parameters for Pearson Distributions

	Skewness	Kurtosis	Family	Distribution Notes	Shift	Scale
1	1.500	7	Type VI	F(DF1=6.777,DF2=38)	-1.641	1.555

Distribution Moments

	Specified Moments				Sample Moments			
	Mean	Std Dev	Skewness	Kurtosis	Mean	Std Dev	Skewness	Kurtosis
1	0	1	1.500	7	0.0029	0.980	0.977	3.515

The SIMSYSTEM procedure determines the unique distribution in the Pearson system that has the specified skewness, kurtosis, mean, and standard deviation. Distributions in the Pearson system are classified into families, also referred to as Pearson types, which are designated by Roman numerals; see "Pearson System of Distributions" on page 1600. The procedure determines both the Pearson type and the mathematical parameters for the distribution, as displayed by the Parameters for Pearson Distributions table in Figure 31.1. The specified moments and the sample moments for the simulated values are displayed in the Distribution Moments table.

As shown in Figure 31.1, the Pearson distribution determined by the procedure belongs to the Type VI family, which corresponds to a shifted and scaled F distribution with 6.777 and 38 degrees of freedom; see "Pearson Type VI Family" on page 1604. Because the sample size is relatively small, the sample skewness and especially the sample kurtosis in the Distribution Moments table are not all that close to the specified values.

In addition to these tables, if ODS Graphics is enabled, PROC SIMSYSTEM displays a graphical map of the Pearson system of distributions in which the specified distribution is marked. The procedure also displays the theoretical density for the distribution and a histogram for the 100 sampled values. These plots are shown in Figure 31.2 and Figure 31.3.

Figure 31.2 Pearson Distributions

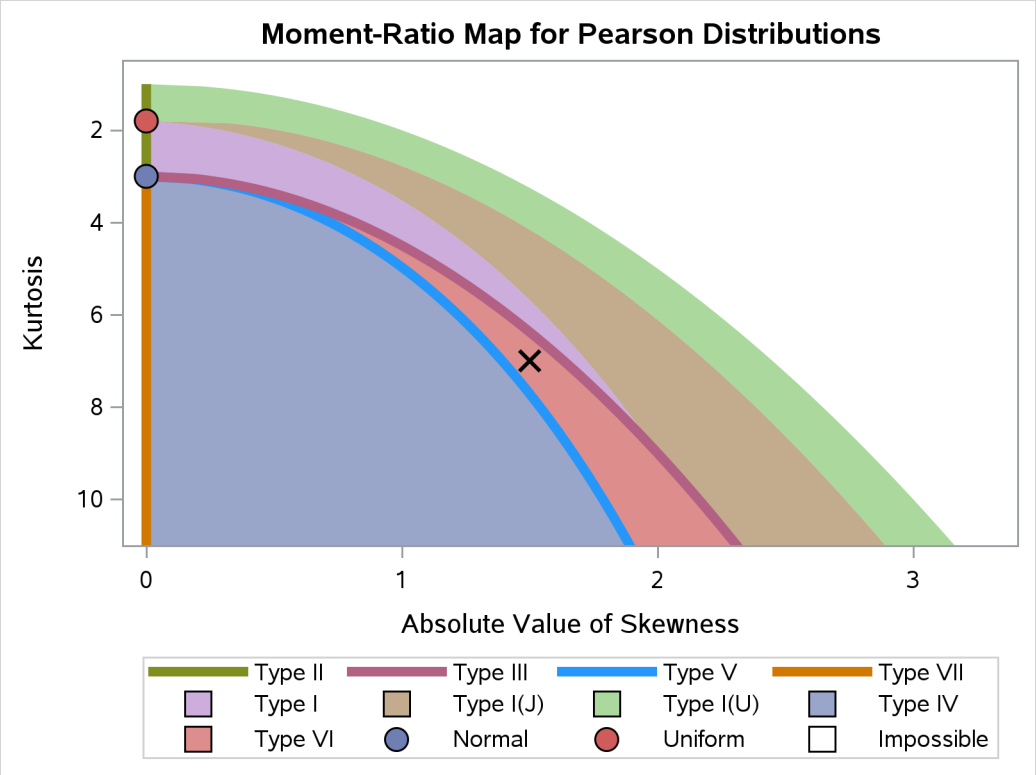
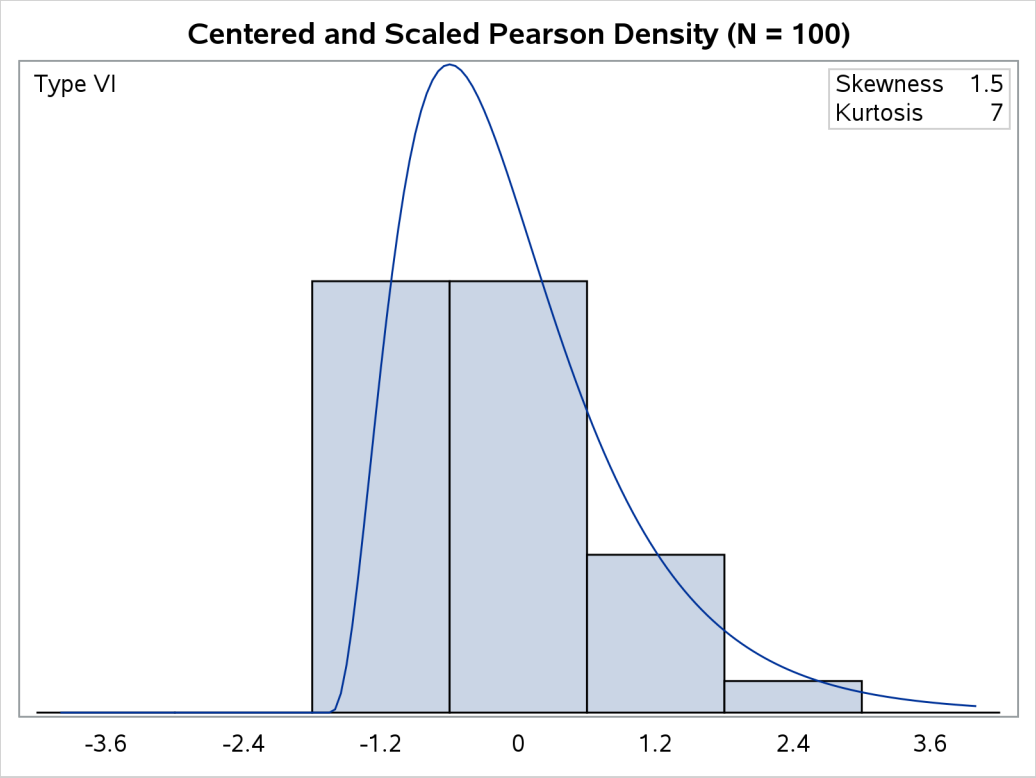


Figure 31.3 Density of Selected Distribution and Histogram of Simulated Values



To carry out an F test with the simulated data, you can create a categorical variable (named A) that randomly groups the measurements by operator and run PROC REGSELECT, as follows:

```
data mylib.sim; set mylib.sim;
   a = mod(iObs-1,5)+1;
   y = variate;
run;

proc regselect data=mylib.sim;
   class a;
   model y = a;
run;
```

Because all the measurements come from a distribution with the same mean, you expect that the F test will fail to reject the hypothesis of no operator effect. Figure 31.4 shows that this is what you get.

Figure 31.4 ANOVA F Test on Data from Distribution with Skewness=1.5 and Kurtosis=7

The REGSELECT Procedure

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	4	3.18986	0.79746	0.82	0.5132
Error	95	91.96079	0.96801		
Corrected Total	99	95.15065			

However, one F test does not say anything about the general characteristics of ANOVA on such data. One way to explore this is to generate many replicates of such null data and see how often the F test is rejected. In addition, because you cannot be certain that the distribution of the functional parameter has a precise skewness of 1.5 and a kurtosis of 7, you need to study how well ANOVA works for a variety of skewness-kurtosis combinations.

For a valid analysis of variance, the F test on null data should be rejected about 5% of the time at the 0.05 level, and likewise 1% and 10% of the time at the 0.01 and 0.10 levels, respectively. The following code checks this by using PROC SIMSYSTEM with the NREP= option to generate 1,000 samples of size 100, and by using multiple skewness and kurtosis pairs in the MOMENTS statement to draw these samples from a selection of six Pearson distributions whose skewness ranges from 0 to 1.5 and whose kurtosis ranges from 3 to 7.

```
proc simsystem system=pearson n=100 seed=12345 nrep=1000;
   moments skewness = 0 0 0.75 0.75 0.75 1.5
           kurtosis = 3 5 3    5    7    7  ;
   output out=mylib.sim;
run;
```

The Parameters table in Figure 31.5 displays the Pearson family types and parameters determined by the procedure for these six distributions. The first distribution is the standard normal distribution, which is the Pearson distribution with a skewness of 0 and a kurtosis of 3. The second distribution is a Pearson Type VII distribution, also known as a t distribution, with 7 degrees of freedom; see "Pearson Type VII Family" on page 1605. The third distribution is a Pearson Type I distribution, also known as a beta distribution, with shape parameters 1.342 and 3.769; see "Pearson Type I Family" on page 1601. The fourth and fifth distributions are Pearson Type IV distributions, which do not correspond to any well-known family of distributions; see

"Pearson Type IV Family" on page 1603. The sixth distribution is the Pearson Type VI distribution, also known as an *F* distribution, that was determined at the beginning of this example; see "Pearson Type VI Family" on page 1604.

For each of these distributions, the Distribution Moments table in Figure 31.5 displays the specified moments and the sample moments averaged over the 1,000 replicate samples.

Figure 31.6 locates the specified skewness-kurtosis combinations for the six distributions in a moment-ratio map of the Pearson system. Figure 31.7 shows the shapes of the six corresponding density functions.

Figure 31.5 Parameters and Average Moments for 1,000 Samples

The SIMSYSTEM Procedure

Parameters for Pearson Distributions

	Skewness	Kurtosis	Family	Distribution Notes	Shift	Scale
1	0	3	Normal		0	1
2	0	5	Type VII	Students t, DF = 7	0	0.845
3	0.750	3	Type I	Beta(1.342,3.769)	-1.475	5.617
4	0.750	5	Type IV	Alpha = -4.633, m = 5.4595	-1.297	2.497
5	0.750	7	Type IV	Alpha = -1.572, m = 3.5842	-0.594	1.953
6	1.500	7	Type VI	F(DF1=6.777,DF2=38)	-1.641	1.555

Distribution Moments

	Specified Moments				Avg Sample Moments			
	Mean	Std Dev	Skewness	Kurtosis	Mean	Std Dev	Skewness	Kurtosis
1	0	1	0	3	-69E-5	0.997	-0.009	3.007
2	0	1	0	5	0.0040	0.994	0.0082	4.424
3	0	1	0.750	3	-0.005	0.993	0.743	3.039
4	0	1	0.750	5	-0.001	0.996	0.667	4.475
5	0	1	0.750	7	-0.006	0.992	0.568	5.252
6	0	1	1.500	7	-0.001	0.989	1.330	5.722

Figure 31.6 Selected Pearson Distributions

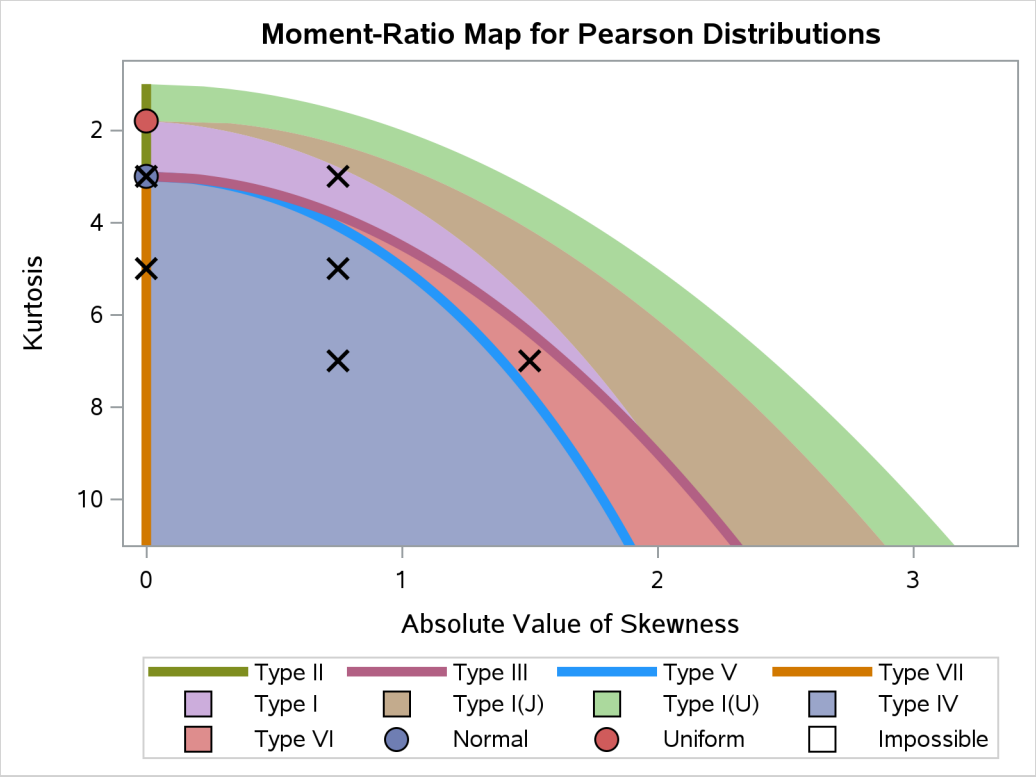
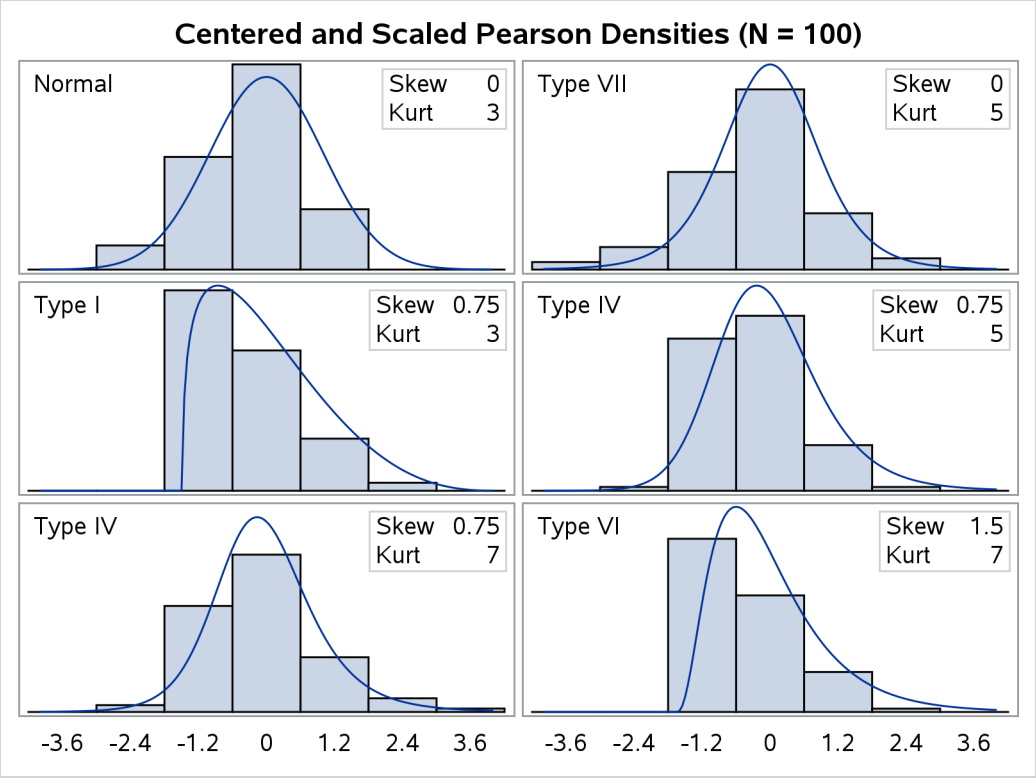


Figure 31.7 Densities and Histograms of Selected Pearson Distributions



Notice how the Pearson system provides all of the distribution shapes necessary to match the entire set of skewness-kurtosis combinations specified in the MOMENTS statement. This could not be accomplished with a single family of distributions.

The average sample moments in Figure 31.5 are all close to the specified values, but there are noticeable discrepancies for the standard deviation, for the skewness, and especially for the kurtosis. This is because these sample moments are biased estimates of the specified population moments, with a bias that can be rather high even for moderately large samples if the kurtosis is large.

In the following statements, the PROC REGSELECT step shown earlier is performed on each of the 1,000 samples of size 100, for each combination of skewness and kurtosis values, and the DATA step and PROC MDSUMMARY are used to compute how often the F test is rejected at the 10%, 5%, and 1% levels. The ODS SELECT NONE statement suppresses printed output for all the ANOVA analyses.

```
data mylib.sim; set mylib.sim;
   a = mod(iObs-1,5)+1;
   y = variate;
run;

ods select none;
proc regselect data=mylib.sim;
   by Skewness Kurtosis Rep;
   class a;
   model y = a;
   displayout ANOVA=AOV;
run;
ods select all;

data mylib.AOV; set mylib.AOV;
   where (Source = "Model");
   Rej10 = (ProbF < 0.10);
   Rej05 = (ProbF < 0.05);
   Rej01 = (ProbF < 0.01);
proc mdsummary data=mylib.AOV;
   groupby Skewness Kurtosis;
   var Rej:;
   output out=mylib.AOV_summary;
proc transpose data=mylib.AOV_summary(rename=(_Column_=_Name_)) out=summary;
   by notsorted Skewness Kurtosis;
   var _Mean_;
proc sort data=summary;
   by Skewness Kurtosis;
proc print data=summary noobs;
   var Skewness Kurtosis Rej01 Rej05 Rej10;
   format Rej01 Rej05 Rej10 5.3;
run;
```

The results, shown in Figure 31.8, indicate that the simulated rejection rates are reasonably close to the nominal levels over the entire set of skewness-kurtosis combinations. The conclusion is that you can be confident in using ANOVA with these nonnormal data.

Figure 31.8 Simulated Rejection Rates for ANOVA F Test on Nonnormal Data

Skewness	Kurtosis	Rej01	Rej05	Rej10
0.00	3	0.013	0.066	0.124
0.00	5	0.005	0.050	0.101
0.75	3	0.011	0.050	0.100
0.75	5	0.009	0.051	0.105
0.75	7	0.012	0.047	0.090
1.50	7	0.009	0.039	0.098

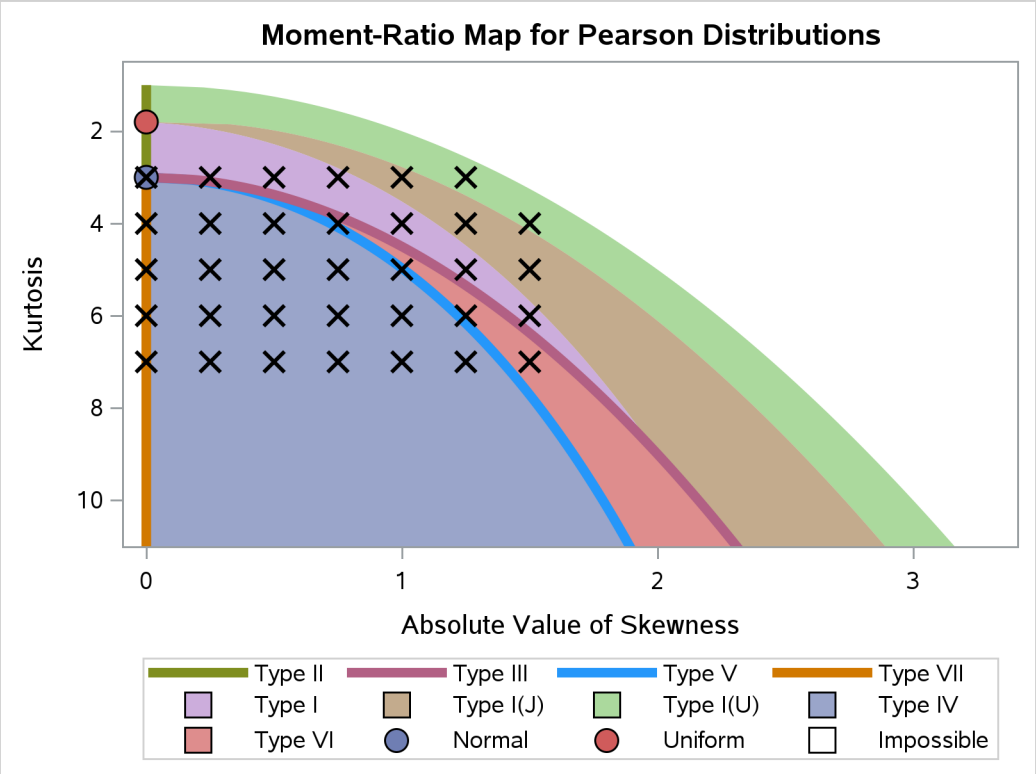
This example demonstrates how easy it is to simulate data from a variety of distributions with a single invocation of the SIMSYSTEM procedure. In practice, you will often want to simulate data from a larger number of distributions whose moment ratios lie on an evenly spaced grid of skewness-kurtosis combinations. The MOMENTGRID statement is useful for this purpose, as shown in the following statements.

```
proc simsystem system=pearson n=100 seed=12345 nrep=1000 plot(only)=mrmap;
   momentgrid skewness = 0 to 1.5 by 0.25
              kurtosis = 3 to 7   by 1;
   output out=mylib.sim;
run;
```

When you use the MOMENTGRID statement, each value of $\sqrt{\beta_1}$ that you specify in the SKEWNESS= option is paired with all the values of β_2 that you specify in the KURTOSIS= option to form a grid of combinations of $\sqrt{\beta_1}$ and β_2.

The Pearson distributions determined by these combinations are mapped in Figure 31.9.

Figure 31.9 Selected Pearson Distributions



Although the preceding MOMENTGRID statement requests Pearson distributions for a grid of $6 \times 7 = 35$ moment ratio combinations, the combination with skewness 1.5 and kurtosis 3 is not feasible because it does not satisfy the condition $\beta_2 > 1 + \beta_1$.

In general, you can use the MOMENTGRID statement to simulate data from distributions specified by a comprehensive grid of evenly spaced combinations of skewness and kurtosis. This enables you to explore the effects of a wide variety of distributional shapes and to form conclusions that are framed in terms of skewness and kurtosis.

Syntax: SIMSYSTEM Procedure

The following statements are available in the SIMSYSTEM procedure:

PROC SIMSYSTEM < *options* > **;**
  MOMENTGRID SKEWNESS=*values* **KURTOSIS=***values* < **MEAN=***values* **STDDEV=***values* > **;**
  MOMENTS SKEWNESS=*values* **KURTOSIS=***values* < **MEAN=***values* **STDDEV=***values* > **;**
  OUTPUT OUT=*libref.data-table* < *keyword*=*name* . . . *keyword*=*name* > **;**

The following sections describe the PROC SIMSYSTEM statement and then describe the other statements in alphabetical order. You must specify the moments of the distributions to be sampled by specifying only one of the following: the DATA= option in the PROC SIMSYSTEM statement; the MOMENTGRID statement; or the MOMENTS statement. The OUTPUT statement is required if you want to save the sample in a data table.

PROC SIMSYSTEM Statement

PROC SIMSYSTEM < *options* > **;**

The PROC SIMSYSTEM statement invokes the SIMSYSTEM procedure. Table 31.1 summarizes the *options* available in the PROC SIMSYSTEM statement.

Table 31.1 PROC SIMSYSTEM Statement Options

Option	Description
CDF=	Specifies values at which to compute the cumulative probability
DATA=	Specifies an input CAS data table
MOMENTREPS	Displays moments of individual sample replicates
N=	Specifies the number of values to sample
NOPRINT	Suppresses the normal display of results
NREP=	Specifies the number of sample replicates
PDF=	Specifies values at which to compute the probability density
PLOTS	Controls the plots that are produced through ODS Graphics
QUANTILE=	Specifies quantiles of the distribution to be computed
SEED=	Specifies the seed for pseudorandom number generation
SIMPLETOL=	Specifies the tolerance value for choosing a simpler distribution
SYSTEM=	Specifies the system of distributions to sample from

You can specify the following *options*:

CDF=*values*
specifies values at which to compute the exact cumulative probability for each distribution. The computed cumulative probabilities are included as nonprinting columns in the table of distribution parameters.

DATA=*libref.data-table*
names the input data table for PROC SIMSYSTEM to use. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to the data, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 1581.

data-table specifies the name of the input data table.

This data table must contain a numeric variable named Skewness and a numeric variable named Kurtosis. The data table can also contain the following numeric variables:

- Mean, which contains the mean values of the distributions to be simulated
- StdDev, which contains the standard deviation values of the distributions to be simulated
- SimIndex, which contains simulation indexes that identify the distributions in output tables and data tables

If you specify this option, then you cannot use the MOMENTS statement or the MOMENTGRID statement.

MOMENTREPS
MREPS
displays sample moments for individual samples when multiple replicates are computed by using the NREP= option.

N=*value*
specifies the number of simulated values to compute for each distribution. By default, N=0; that is, if you omit this option, then no samples are computed.

NOPRINT
suppresses the normal display of results. This option temporarily disables the Output Delivery System (ODS).

NREP=*value*
specifies the number of sample replicates to compute for each distribution. If you specify a value larger than 1, then PROC SIMSYSTEM displays the sample moments averaged over the replicates instead of displaying the sample moments. If you still want to see the moments for individual samples, use the MOMENTREPS option.

By default, NREP=1.

PDF=*values*
specifies values at which to compute the exact probability density function for each distribution. The computed PDF values are included as nonprinting columns in the table of distribution parameters.

QUANTILE=*values*
specifies probability values at which to compute the exact quantiles for each distribution, where each of the specified *values* must be between 0 and 1, inclusive. The computed quantiles are included as nonprinting columns in the table of distribution parameters.

PLOTS <(*global-plot-options***)> <=***plot-request* **<(***options***)>>**

PLOTS <(*global-plot-options***)> <=(***plot-request* **<(***options***)> <...** *plot-request* **<(***options***)>>)>**
controls the plots produced through ODS Graphics. When you specify only one *plot-request*, you can omit the parentheses around the *plot-request*. The following examples show a variety of ways that you can specify this option:

```
PLOTS=NONE
PLOTS=(MRMAP HISTOGRAM)
PLOTS(UNPACK)=HISTOGRAM
PLOT=MRMAP(SKEWSCALE=SQUARE)
```

ODS Graphics must be enabled before plots can be requested, as shown in the following example:

```
ods graphics on;
proc simsystem n=100 seed=12345;
   momentgrid skew=1.5 kurt=7;
   output out=mylib.sim;
run;
ods graphics off;
```

If ODS Graphics is enabled but you do not specify the PLOTS= option, then PROC SIMSYSTEM produces a moment-ratio map (PLOT=MRMAP) by default and also a histogram of the centered and scaled sample values. If multiple distributions are being sampled, then the histogram for each one is displayed by default in a lattice.

You can specify the following *global-plot-options*:

ONLY
suppresses the default plots. Only plots that you specifically request are displayed.

UNPACKPANEL

UNPACK
suppresses paneling. By default, multiple histograms can appear in an output panel. Specify this option to display each plot in a separate panel.

You can specify the following individual *plot-requests* and their *plot-options*. If you specify only one *plot-request*, then you can omit the parentheses.

ALL
produces all appropriate plots. You can specify other options by using ALL; for example, to request all plots and unpack only the histogram lattices, specify PLOTS=(ALL HISTOGRAM(UNPACK)).

HISTOGRAM< (UNPACK) >
modifies the histograms of sampled values. By default, histograms of multiple distributions are displayed together in a panel. The UNPACK suboption suppresses their display in a panel and instead produces the series of individual plots that form the paneled display.

MRMAP< (INDEXLABELS) >
modifies the moment-ratio map. The specified distributions are marked on the map by using their simulation index values instead of the default marker.

MRMAP< (KURTSCALE=DIRECT | REVERSE) >
modifies the moment-ratio map. You can specify the following values for the KURTSCALE= suboption:

DIRECT displays the kurtosis values in increasing order from bottom to top.

REVERSE displays the kurtosis values in reverse order, increasing from top to bottom.

By default, KURTSCALE=REVERSE.

MRMAP< (SKEWSCALE=ABS | SKEWNESS | SQUARE) >
modifies the moment-ratio map. You can specify the following values for the SKEWSCALE= suboption:

ABS displays the absolute value of skewness.

SKEWNESS displays the skewness.

SQUARE displays the square of the skewness.

By default, SKEWSCALE=ABS.

NONE
suppresses all graphics.

SEED=*n*
specifies an integer to use to start the pseudorandom number generator for drawing samples. If you do not specify a seed or if you specify a value less than or equal to 0, the seed is generated by reading the time of day from the computer's clock.

SIMPLETOL=*n*
specifies a tolerance value to use in choosing to map input skewness and kurtosis values to those of a simpler distribution. For more information about how this value is used, see "Specifying Boundary Widths for Transition Families" on page 1610. The default value for n is 10^{-4}.

SYSTEM=PEARSON | JOHNSON
specifies the system to use to simulate data. You can specify the following values:

JOHNSON uses the Johnson system (Johnson 1949).

PEARSON uses the Pearson system (Pearson 1895, 1901, 1916).

By default, SYSTEM=PEARSON. For more information about the Johnson and Pearson systems, see "Details: SIMSYSTEM Procedure" on page 1597.

MOMENTGRID Statement

MOMENTGRID SKEWNESS=*values* **KURTOSIS=***values* < **MEAN=***values* **STDDEV=***values* > **;**

The MOMENTGRID statement specifies a grid of combinations of skewness ($\sqrt{\beta_1}$) and kurtosis (β_2) for the distributions that you want to sample. Each value of $\sqrt{\beta_1}$ that you specify in the SKEWNESS= option is paired with all the values of β_2 that you specify in the KURTOSIS= option to form a series of combinations $(\sqrt{\beta_1}, \beta_2)$. Distributions are produced and sampled for all the combinations.

The moment values that you specify should be the values that you believe might be relevant to the process that you intend to study with the simulation. They might be suggested by sample moments computed from historical data or by the shape of the corresponding distributions, as indicated by surveys such as those in "Shapes of Pearson and Johnson Density Functions" on page 1611.

The MOMENTGRID statement is recommended for situations in which you want to conduct a simulation study for a rectangular grid of moment ratio combinations, so that you can draw conclusions about the systematic effects of skewness and kurtosis. If you use the MOMENTGRID statement, then you cannot also use the MOMENTS statement or the DATA= option in the PROC SIMSYSTEM statement.

You must specify the following options:

KURTOSIS=*values*
specifies values of kurtosis, denoted by β_2. Kurtosis values must be greater than $1 + \beta_1$, where $\sqrt{\beta_1}$ is the associated value of skewness that you specify in the SKEWNESS= option.

SKEWNESS=*values*
specifies the skewness values. Any numeric value is valid in principle, although extremely heavily skewed distributions might be numerically unstable.

You can also specify the following options:

MEAN=*values*
specifies the mean values. By default, MEAN=0 for all distributions.

STDDEV=*values*
specifies the standard deviation values. By default, STDDEV=1 for all distributions.

You can specify any number of values in any of these options. The number of combinations that are defined will be the product of the number of values in each specification.

MOMENTS Statement

MOMENTS SKEWNESS=*values* **KURTOSIS=***values* < **MEAN=***values* **STDDEV=***values* > **;**

The MOMENTS statement specifies parallel lists of skewness and kurtosis for the distributions that you want to sample. Each value of $\sqrt{\beta_1}$ that you specify in the SKEWNESS= option is paired with the corresponding value of β_2 that you specify in the KURTOSIS= option to form a list of combinations $(\sqrt{\beta_1}, \beta_2)$. A distribution is produced and sampled for each combination in the list.

The moment values that you specify should be the values that you believe might be relevant to the process that you intend to study with the simulation. They might be suggested by sample moments computed from historical data or by the shape of the corresponding distributions, as indicated by surveys such as those in "Shapes of Pearson and Johnson Density Functions" on page 1611.

If you use the MOMENTS statement, then you cannot also use the MOMENTGRID statement or the DATA= option in the PROC SIMSYSTEM statement.

You must specify the following options:

KURTOSIS=*values*
specifies values of kurtosis, denoted by β_2. Kurtosis values must be greater than $1 + \beta_1$, where $\sqrt{\beta_1}$ is the associated value of skewness that you specify in the SKEWNESS= option.

SKEWNESS=*values*
specifies values of skewness, denoted by $\sqrt{\beta_1}$. Any numeric value is valid in principle, although extremely heavily skewed distributions might be numerically unstable.

You can also specify the following options:

MEAN=*values*
specifies the mean values. By default, MEAN=0 for all distributions.

STDDEV=*values*
specifies values of the standard deviation, which must be positive. By default, STDDEV=1 for all distributions.

If you specify $n > 1$ values in any of the four options (SKEWNESS=, KURTOSIS=, MEAN=, and STDDEV=), then in each of the remaining options you must specify either exactly n values or one value. The procedure will create exactly n combinations of moments by using values from each option in the order specified and by reusing single values.

OUTPUT Statement

OUTPUT OUT=*libref.data-table* < *keyword*=*name* . . . *keyword*=*name* > **;**

The OUTPUT statement creates a new data table in which to save the values that are sampled from all specified distributions.

You must specify the following option:

OUT=*libref.data-table*
names the output data table for PROC SIMSYSTEM to use. You must specify this option before any other options. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to where the data table is to be stored, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 1581.

data-table specifies the name of the output data table.

You can also specify one or more of the following *keyword=name* pairs:

IOBS=*name*
assigns a *name* to the variable that indexes the observation within a replicate of the distribution being sampled.

KURT=*name*
assigns a *name* to the variable that contains the kurtosis values.

MEAN< =*name* **>**
adds the mean of the distribution being sampled to the output data table and optionally assigns a *name* to it. If you do not specify a *name*, the name Mean is used by default.

REPIDX=*name*
assigns a *name* to the variable that indexes the replicate of the distribution being sampled.

SIMIDX=*name*
assigns a *name* to the variable that indexes the distribution being sampled.

SIMVALUE=*name*
assigns a *name* to the variable that contains the sampled values.

SKEW=*name*
assigns a *name* to the variable that contains the skewness values.

STDDEV< =*name* **>**
adds the standard deviation of the distribution being sampled to the output data table and optionally assigns a *name* to it. If you do not specify a *name*, the name StdDev is used by default.

By default, the output data table contains the variables Skewness and Kurtosis, which define the distribution that is being sampled; the variable Variate, which contains the sampled values; and the variable SimIndex, which indexes the distribution being sampled. If you specify the MEAN or STDDEV keyword, the output data table also contains the mean (Mean) or standard deviation (StdDev), respectively, of the distribution being sampled.

Details: SIMSYSTEM Procedure

Moments and Moment Ratios

For a random variable X, the first moment (expected value) is denoted by

$$\mu = E[X]$$

The rth uncorrected moment (rth moment about zero) is denoted by

$$\mu'_r(X) = \mu'_r = E[X^r], \quad r = 2, 3, \ldots$$

The rth central moment (rth moment about the mean) is denoted by

$$\mu_r(X) = \mu_r = E[(X - E[X])^r], \quad r = 2, 3, \ldots$$

The central moments can be expressed in terms of the uncorrected moments. In particular,

$$\begin{aligned} \mu_2 &= \mu'_2 - \mu^2 \\ \mu_3 &= \mu'_3 - 3\mu'_2\mu + 2\mu^3 \\ \mu_4 &= \mu'_4 - 4\mu'_3\mu + 6\mu'_2\mu^2 - 3\mu^4 \end{aligned}$$

For $r = 3, 4, \ldots$, the ratio $\mu_r/\mu_2^{r/2}$ is referred to as the rth moment ratio (rth standardized moment). Moment ratios are denoted as

$$\beta_{r-2}^{(r-2)/2} = \mu_r/\mu_2^{r/2}, \quad r = 3, 4, \ldots$$

For any two constants a and b, the moment ratios for $aX + b$ are identical to the moment ratios for X. In other words, moment ratios are invariant with respect to the location and scale of a distribution. Moment ratios vary with the shape of a distribution, and for this reason they are often called shape factors.

Moment ratios do not uniquely determine the shape of a distribution (or more precisely, its density function). In fact, they do not even exist for some distributions, such as the Cauchy distribution. However, for continuous distributions whose first four moments exist, the first two moment ratios, β_1 (or $\sqrt{\beta_1}$) and β_2, are useful measures of the skewness and kurtosis.

Definitions and Interpretations of Skewness and Kurtosis

When you run the SIMSYSTEM procedure, you specify the Pearson and Johnson distributions to use for simulation by their shapes—as quantified by their measures of skewness and kurtosis—rather than by their mathematical parameters. Unfortunately, the definitions and interpretations of skewness and kurtosis are slightly inconsistent in the statistics literature and in SAS. This section explains subtle differences that you should be aware of when running the procedure and reporting your results.

In the context of the SIMSYSTEM procedure, the terms skewness and kurtosis refer to the shape characteristics of probability distributions that are used to simulate data. Elsewhere in SAS, these terms typically apply to shape characteristics of data distributions. For example, several summarization procedures, such as PROC UNIVARIATE, compute sample skewness and sample kurtosis among the descriptive statistics that they provide for univariate data.

Skewness measures the asymmetry of a distribution about its median or mean. It is important to consider various degrees of skewness when you select the distributions for a simulation study. Typically, a distribution is positively skewed (right skewed) if the tail of high values is longer than the tail of low values, and it is negatively skewed (left skewed) if the tail of low values is longer than the tail of high values.

In the statistics literature, different authors use $\sqrt{\beta_1}$ and β_1 to measure the skewness of a probability distribution. The sign of $\sqrt{\beta_1}$ can be either positive or negative, and the sign is often included with $\sqrt{\beta_1}$. This can lead to confusion, so it is important to understand and indicate which measure is being used when you are specifying or reporting values of skewness.

The SIMSYSTEM procedure consistently uses $\sqrt{\beta_1}$ as the measure for skewness in its syntax and output. Note the following points:

- You must specify values of $\sqrt{\beta_1}$ in the SKEWNESS= option in the MOMENTS and MOMENTGRID statements.
- Values of $\sqrt{\beta_1}$ are displayed in the Specified Skewness column in the parameters and moments tables.
- In moment-ratio diagrams produced by the procedure, the axis labeled "Skewness" refers to $\sqrt{\beta_1}$.

Also note that values of sample skewness in moments tables are computed using the square root version of the formula for sample skewness. The UNIVARIATE and MEANS procedures also use this version.

Kurtosis measures the heaviness of the tails of a distribution relative to that of the normal distribution. The degree of kurtosis—like the degree of skewness—is a consideration in selecting distributions for simulation.

Kurtosis is sometimes said to measure the peakedness of a distribution as opposed to its flatness. However, Westfall (2014) and other authors have pointed out that this interpretation is incorrect. Westfall makes the following distinction:

> Kurtosis tells you virtually nothing about the shape of the peak—its only unambiguous interpretation is in terms of tail extremity, that is, either existing outliers (for the sample kurtosis) or propensity to produce outliers (for the kurtosis of a probability distribution).

For additional discussion of the interpretation of kurtosis, see Darlington (1970), Moors (1986), Ruppert (1987), and Balanda and MacGillivray (1988).

In the statistics literature, different authors use β_2 and $\beta_2 - 3$ (referred to as *excess kurtosis*) to measure the kurtosis of a probability distribution. The reason for adjusting β_2 by 3 is that the kurtosis for the normal distribution is then 0, which seems like a more natural baseline for comparisons with the kurtosis of other distributions. This can lead to confusion, so it is important to understand and indicate which measure is being used when you are specifying or reporting values of kurtosis.

The SIMSYSTEM procedure consistently uses β_2 as the measure for kurtosis in its syntax and output. Note the following points:

- You must specify values of β_2 in the KURTOSIS= option in the MOMENTS and MOMENTGRID statements.
- Values of β_2 are displayed in the Specified Kurtosis column in the parameters and moments tables.
- In moment-ratio diagrams produced by the procedure, the axis labeled "Kurtosis" refers to β_2.

Also note that values of sample kurtosis in moments tables are computed using a sample estimate of β_2. In contrast, the UNIVARIATE and MEANS procedures report a sample estimate of $\beta_2 - 3$.

Moment Ratio Coverage

For any distribution whose first four moments exist, $\sqrt{\beta_1}$ and β_2 satisfy the feasibility condition

$$\beta_2 - \beta_1 - 1 \geq 0$$

Moment-ratio diagrams such as those in Figure 31.10 and Figure 31.11 represent the skewness and kurtosis values of all feasible distributions by the region of the $(\sqrt{\beta_1}, \beta_2)$ plane that lies above the curve $\beta_2 = 1 + \beta_1$. Within this region, the skewness and kurtosis values of familiar parametric families of distributions are represented by points, curves, or subregions, depending on whether the number of shape parameters for the distribution is 0, 1, or 2, respectively.

For example, consider the subregions that correspond to the normal, lognormal, and beta families. The normal distribution, which lacks a shape parameter, is represented by the point at $\sqrt{\beta_1} = 0$ and $\beta_2 = 3$. The lognormal family of distributions is represented by the parametric curve whose coordinates are

$$\begin{aligned} \sqrt{\beta_1} &= (\omega - 1)^{\frac{1}{2}}(\omega + 2) \\ \beta_2 &= \omega^4 + 2\omega^3 + 3\omega^2 - 3 \end{aligned}$$

where $\omega = \exp(\sigma^2)$ and σ is the lognormal shape parameter ($\sigma > 0$). The beta family of distributions is represented by the subregion whose coordinates are

$$\begin{aligned} \sqrt{\beta_1} &= 2(\beta - \alpha)\frac{\sqrt{\alpha^{-1} + \beta^{-1} + (\alpha\beta)^{-1}}}{\alpha + \beta + 2} \\ \beta_2 &= 3(\alpha + \beta + 1)\frac{2(\alpha + \beta)^2 + \alpha\beta(\alpha + \beta - 6)}{\alpha\beta(\alpha + \beta + 2)(\alpha + \beta + 3)} \end{aligned}$$

where α and β are the shape parameters ($\alpha > 0$ and $\beta > 0$).

Unlike families of distributions that are commonly used for simulation, the Pearson and Johnson systems of distributions completely span the feasible region of the $(\sqrt{\beta_1}, \beta_2)$ plane. For any combination of skewness and kurtosis within this region, there is exactly one distribution in each of the Pearson and Johnson systems whose values of $\sqrt{\beta_1}$ and β_2 match that combination. Furthermore, a number of familiar distributions that are used for simulation (such as the normal, lognormal, and beta) are special cases of the Pearson and Johnson systems. Consequently, simulating data from these two systems makes it possible to synthesize data that have a much broader range of distributional shapes that vary evenly and systematically. Using the SIMSYSTEM procedure, you can request a comprehensive set of shapes simply by specifying a grid of values of skewness and kurtosis, as opposed to specifying particular families and parameters.

Pearson System of Distributions

The Pearson system of continuous distributions was introduced by Karl Pearson beginning in the 1890s (Pearson 1895, 1901, 1916). For an explanation of Pearson's derivation, see Chapter 4 of Elderton and Johnson (1969) and Chapter 12 of Johnson, Kotz, and Balakrishnan (1994). Rodriguez (2006) and Ord (2006) provide overviews of the Pearson system.

Each probability density function $p(y)$ in the Pearson system is a solution of the differential equation

$$\frac{d \log p(y)}{dy} = \frac{a + y}{c_0 + c_1 y + c_2 y^2}$$

The location, scale, and shape parameters of the density function depend on the constants a, c_0, c_1, and c_2. Solving the equation for various combinations of these constants gives rise to different families of distributions, which Pearson designated by using roman numerals. As pointed out by Johnson, Kotz, and Balakrishnan (1994), this numbering scheme does not have a clear basis, but it is well established in the statistical literature.

Subsequent to Pearson's work, most families in the Pearson system were re-derived in other contexts, where they received other names that are familiar in modern statistical work. For instance, the distributions in the Pearson Type III family are now commonly known as the gamma distributions. The parameters table that is produced by the SIMSYSTEM procedure identifies each Pearson distribution used for simulation by its Pearson type and its common name (if there is one). This table also provides the location, scale, and shape parameter(s) for each distribution.

There are three main types of distributions in the Pearson system:

- the Type I family, known as beta distributions
- the Type IV family, which has no familiar counterpart
- the Type VI family, known as F distributions

Each of these three families has two shape parameters. Their regions in the $(\sqrt{\beta_1}, \beta_2)$ plane complement each other and completely cover the feasible region.

The Pearson system also includes families of distributions that are limiting forms of the main types and are referred to as transition types:

- the normal distribution (no associated type number)

- the Type II family, known as symmetric beta distributions
- the Type III family, known as gamma distributions
- the Type V family, known as inverse gamma distributions
- the Type VII family, known as Student's *t* distributions

Finally, the Pearson system also includes families of distributions that are special cases of the more important types:

- Type VIII, known as power distributions
- Type IX, known as power distributions
- Type X, known as exponential distributions
- Type XI, known as power distributions
- Type XII, a special case of Type I

These additional types are not required for defining distributions that cover the full skewness/kurtosis range, so they are not included in the PROC SIMSYSTEM results.

The following subsections provide the probability density function for each Pearson family. The parameter notation matches that of the parameters table produced by the procedure.

Pearson Normal Family

The family of normal distributions belongs to the Pearson system in the sense that it is a limiting form of each of the three main types of Pearson distributions. Pearson did not assign a roman numeral to the normal family, but it could be referred to as the "Type 0 family."

The probability density function for the normal family is

$$p(y) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2}\left(\frac{y-\mu}{\sigma}\right)^2\right], \quad \sigma > 0$$

For normal distributions, $\sqrt{\beta_1} = 0$ and $\beta_2 = 3$.

Pearson Type I Family

Pearson Type I probability density functions are obtained as solutions to Pearson's differential equation when the roots of $c_0 + c_1 y + c_2 y^2 = 0$ are real and have opposite signs.

The Type I family is composed of the beta distributions, whose probability density functions are of the form

$$p(y) = \frac{1}{B(\alpha, \beta)} \frac{(y-a)^{\alpha-1}(b-y)^{\beta-1}}{(b-a)^{\alpha+\beta-1}}, \quad a \le y \le b$$

where a is the location parameter, $b - a$ is the scale parameter, and α and β are shape parameters ($\alpha > 0$ and $\beta > 0$). The range of the distribution is bounded below by a and above by b.

The standard form of the beta distribution is obtained by setting $a = 0$ and $b = 1$, and the standard beta density function is

$$p(y) = \frac{1}{B(\alpha, \beta)} y^{\alpha-1}(1-y)^{\beta-1}, \quad 0 \le y \le 1$$

Beta density functions come in a variety of shapes:

- unimodal (bell-shaped) when $\alpha > 1$ and $\beta > 1$
- unimodal and symmetric when $\alpha > 1$, $\beta > 1$, and $\alpha = \beta$
- rectangular when $\alpha = \beta = 1$ (uniform distributions)
- bimodal (U-shaped) when $\alpha < 1$ and $\beta < 1$
- U-shaped and symmetric when $\alpha < 1$, $\beta < 1$, and $\alpha = \beta$
- J-shaped or reverse J-shaped when $(\alpha - 1)(\beta - 1)$ is not positive

For more information about beta distributions, see Chapter 25 of Johnson, Kotz, and Balakrishnan (1995).

Pearson Type II Family

The Pearson Type II family is a limiting form of the Type I family. It is composed of the symmetric beta distributions, which have identical shape parameters $\alpha = \beta$, and its probability density functions are of the general form

$$p(y) = \frac{1}{B(\alpha, \alpha)} \frac{(y-a)^{\alpha-1}(b-y)^{\alpha-1}}{(b-a)^{2\alpha-1}}, \quad a \le y \le b$$

where a is the location parameter, $b - a$ is the scale parameter, and $\alpha > 0$. The range of the distribution is bounded below by a and above by b.

Setting $a = -1$ and $b = 1$ gives the particular form

$$p(y) = \frac{1}{B(\alpha, \alpha)} \frac{(1+y)^{\alpha-1}(1-y)^{\alpha-1}}{2^{2\alpha-1}}, \quad -1 \le y \le 1$$

which can be rewritten as

$$p(y) = \frac{1}{B(\alpha, \alpha)} \frac{\left(1-y^2\right)^{\alpha-1}}{2^{2\alpha-1}}, \quad -1 \le y \le 1$$

Some references present this as the standard form of the Type II density function.

For distributions in the Type II family, $\sqrt{\beta_1} = 0$ and $\beta_2 < 3$. The uniform distribution is a special case of the Type II family.

Pearson Type III Family

The Pearson Type III family is composed of the gamma distributions, whose probability density functions are of the form

$$p(y) = \frac{[(y-\xi)/\lambda]^{\alpha-1} \exp[-(y-\xi)/\lambda]}{\lambda\Gamma(\alpha)}, \quad y > \xi$$

where ξ is the location parameter, λ is the scale parameter, and α is the shape parameter ($\alpha > 0$). The range of the distribution is bounded below by ξ.

For distributions in the Type III family,

$$2\beta_2 - 3\beta_1 - 6 = 0$$

The standard form of the gamma distribution is obtained by setting $\xi = 0$ and $\lambda = 1$:

$$p(y) = \frac{y^{\alpha-1} \exp(-y)}{\Gamma(\alpha)}, \quad y > 0$$

Gamma density functions come in a variety of shapes:

- unimodal (bell-shaped) when $\alpha > 1$
- reverse J-shaped and finite at $y = 0$ when $\alpha = 1$ (exponential distributions)
- reverse J-shaped and tending to infinity as y tends to 0

Within the gamma family, three subfamilies are particularly well-known by other names:

- Gamma distributions where $\alpha = 1$ are referred to as exponential distributions and form the Pearson Type X family.
- Gamma distributions for which α is an integer are referred to as Erlang distributions.
- Standard gamma distributions where $\lambda = 2$ and $\alpha = \nu/2$ for $\nu > 0$ are referred to as chi-square distributions with ν degrees of freedom.

For more information about gamma distributions, see Chapter 17 of Johnson, Kotz, and Balakrishnan (1994).

Pearson Type IV Family

Pearson Type IV probability density functions are obtained as solutions to Pearson's differential equation when the roots of $c_0 + c_1 y + c_2 y^2 = 0$ are complex.

The Pearson Type IV family is composed of distributions whose probability density functions are of the form

$$p(y) = K\left(1 + \left(\frac{y-\xi}{\lambda}\right)^2\right)^{-m} \exp\left(-\nu \tan^{-1}\left(\frac{y-\xi}{\lambda}\right)\right), \quad -\infty < y < \infty$$

where ξ is the location parameter, λ is the scale parameter, m and ν are shape parameters ($m > 5/2$), and K is the constant of integration.

Elderton and Johnson (1969, pp. 58–66) explain how to compute K and the parameters given μ, μ_2, $\sqrt{\beta_1}$, and β_2. Johnson, Kotz, and Balakrishnan (1994, pp. 18–19) note that integrating the expression for $p(y)$ to obtain the cumulative distribution function leads to intractable mathematics. The SIMSYSTEM procedure uses specialized numerical techniques to compute K, the distribution function, and the corresponding quantile function.

Type IV distributions are unimodal and bell-shaped. No common statistical distributions are of the Type IV form.

Pearson Type V Family

Pearson Type V probability density functions are obtained as solutions to Pearson's differential equation when $c_1^2 = 4c_0c_2$.

The Pearson Type V family includes the inverse gamma distributions, whose probability density functions are of the form

$$p(y) = \frac{\lambda^\alpha}{\Gamma(\alpha)} \left(\frac{y-\xi}{\lambda}\right)^{-\alpha-1} \exp\left(-\left(\frac{y-\xi}{\lambda}\right)^{-1}\right), \quad y > \xi$$

where ξ is the location parameter, λ is the scale parameter ($\lambda > 0$), and α is the shape parameter ($\alpha > 0$). Inverse gamma densities are unimodal and either bell-shaped or skewed.

The Type V family is a transition family; its values of skewness and kurtosis lie on a curve in the $(\sqrt{\beta_1}, \beta_2)$ plane that separates the regions of the Type IV and Type VI families. The equation for this curve is

$$\beta_1 (\beta_2 + 3)^2 = 4 (4\beta_2 - 3\beta_1) (2\beta_2 - 3\beta_1 - 6)$$

Pearson Type VI Family

Pearson Type VI probability density functions are obtained as solutions to Pearson's differential equation when the roots of $c_0 + c_1 y + c_2 y^2 = 0$ are real and have the same sign. If the roots a_1 and a_2 are both negative and $a_1 < a_2 < 0$, then the densities have the general form

$$p(y) = K(y - a_1)^{m_1} (y - a_2)^{m_2}$$

where K is a constant of integration, $m_2 < -1$, and $m_1 + m_2 < 0$. For more information, see Johnson, Kotz, and Balakrishnan (1994, chap. 12).

Pearson Type VI probability density functions can be expressed as

$$p(y) = \frac{1}{B(\nu_1/2, \nu_2/2)} \frac{\left(\frac{y-\xi}{\lambda}\right)^{(\nu_1/2)-1}}{\left(1 + \frac{y-\xi}{\lambda}\right)^{(\nu_1+\nu_2)/2}}, \quad y > \xi$$

where ξ is the location parameter, λ is the scale parameter ($\lambda > 0$), and ν_1 and ν_2 are shape parameters ($\nu_1 > 0$, $\nu_2 > 0$). For $\xi = 0$ and $\lambda = 1$, this reduces to the standard form

$$p_G(y) = \frac{1}{B(\nu_1/2, \nu_2/2)} \frac{y^{(\nu_1/2)-1}}{1 + y^{(\nu_1+\nu_2)/2}}, \quad y > 0$$

which is the density function of $G = X_1/X_2$, the ratio of two independent variables X_1 and X_2 with chi-square distributions whose degrees of freedom are ν_1 and ν_2, respectively (Johnson, Kotz, and Balakrishnan 1995, chap. 27). Pearson Type VI distributions are also known as beta distributions of the second kind (Johnson, Kotz, and Balakrishnan 1995, chap. 25).

Pearson Type VI distributions are closely related to F distributions (Johnson, Kotz, and Balakrishnan 1995, chap. 27). The ratio

$$F = \frac{X_1/\nu_1}{X_2/\nu_2}$$

has the F distribution with degrees of freedom ν_1 and ν_2. The density function of F is

$$p_F(y) = \frac{(\nu_1/\nu_2)^{\nu_1/2}}{B(\nu_1/2, \nu_2/2)} \frac{y^{(\nu_1/2)-1}}{(1+\nu_1 y/\nu_2)^{(\nu_1+\nu_2)/2}}, \quad y > 0$$

Pearson Type VII Family

Pearson Type VII probability density functions are obtained as solutions to Pearson's differential equation when $c_1 = a = 0$, $c_0 > 0$, and $c_2 > 0$.

Pearson Type VII densities have the general form

$$p(y) = K(c_0 + c_2 y^2)^{-1/(2c_2)}$$

where K is a constant of integration.

For distributions in the Type VII family, $\sqrt{\beta_1} = 0$ and $\beta_2 > 3$.

Pearson Type VII distributions are closely related to Student's t distributions, which have the form

$$p_{t_\nu}(y) = \frac{\Gamma\left((\nu+1)/2\right)}{\Gamma(\nu/2)} \left(1 + \frac{y^2)}{2}\right)^{-(\nu+1)/2} \qquad y > 0$$

Note that ν must be positive but does not need to be an integer. The t distributions have skewness $\sqrt{\beta_1} = 0$ and kurtosis $\beta_2 = 3(\nu-2)/(\nu-4)$, which exist if $\nu > 4$. For more information about the t family, see Johnson, Kotz, and Balakrishnan (1995, chap. 28) .

Parameter Values for Pearson Distributions

The purpose of this subsection is to explain how parameter values that the SIMSYSTEM procedure computes for the well-known distributions that underlie most of the Pearson system are related to the parameters used by the other functions in SAS to work with these distributions. These distributions are special cases of the Pearson system, and they are identified in the parameters table when the combination of skewness and kurtosis specified matches the theoretical skewness and kurtosis for these cases. The tables that are provided are helpful for understanding what the procedure is computing in each particular case, and they are also helpful if you want to reuse parameter values from the procedure to compute your own simulations in DATA step code by using the RAND function.

Table 31.2 summarizes the Pearson types, their corresponding standard distributions, and the mathematical form of the density for these standard distributions in terms of the parameters displayed in the "Parameters for Pearson Distributions" table. In Table 31.2, $\Gamma(\alpha)$ denotes the gamma function, and $B(\alpha, \beta) = \Gamma(\alpha)\Gamma(\beta)/\Gamma(\alpha+\beta)$ denotes the beta function. The one exception is Type IV, which is essentially a skewed version of Student's t distribution. There is no closed form for the constant C in the Type IV density; internally, it is computed by using specialized methods.

Table 31.2 Pearson Distributions for $y = (x - \xi)/\lambda$

Type	Standard Distribution	Probability Density Function	Constant C
Type I	Beta(α,β)	$Cy^{\alpha-1}(1-y)^{\beta-1}$	$1/B(\alpha,\beta)$
Type II	Beta(α,α)	$Cy^{\alpha-1}(1-y)^{\alpha-1}$	$1/B(\alpha,\alpha)$
Type III	Gamma(α)	$Cy^{\alpha-1}\exp(-y)$	$1/\Gamma(\alpha)$
Type IV		$C(1+y^2)^{-m}\exp(-\alpha\tan^{-1}(y))$	No closed form
Type V	Inverse gamma(α)	$C(1/y)^{\alpha+1}\exp(-1/y)$	$1/\Gamma(\alpha)$
Type VI	F_{d_1,d_2}	$Cy^{d_1/2-1}(d_2+d_1y)^{-(d_1+d_2)/2}$	$d_1^{d_1/2}d_2^{d_2/2}/B(d_1/2,d_2/2))$
Type VII	Student's t_d	$C(1+y^2/d)^{(d+1)/2}$	$\sqrt{d}\,B(d/2,1/2)$

The precise numeric values of these parameters are available as hidden columns named Shap1Val and Shap2Val in the underlying ODS object for this table. For illustration, Table 31.3 shows DATA step code that is equivalent to how PROC SIMSYSTEM generates random variates for these distributions, after it determines the appropriate values of these shape parameters.

Table 31.3 Pearson Sampling for $y = (x - \xi)/\lambda$

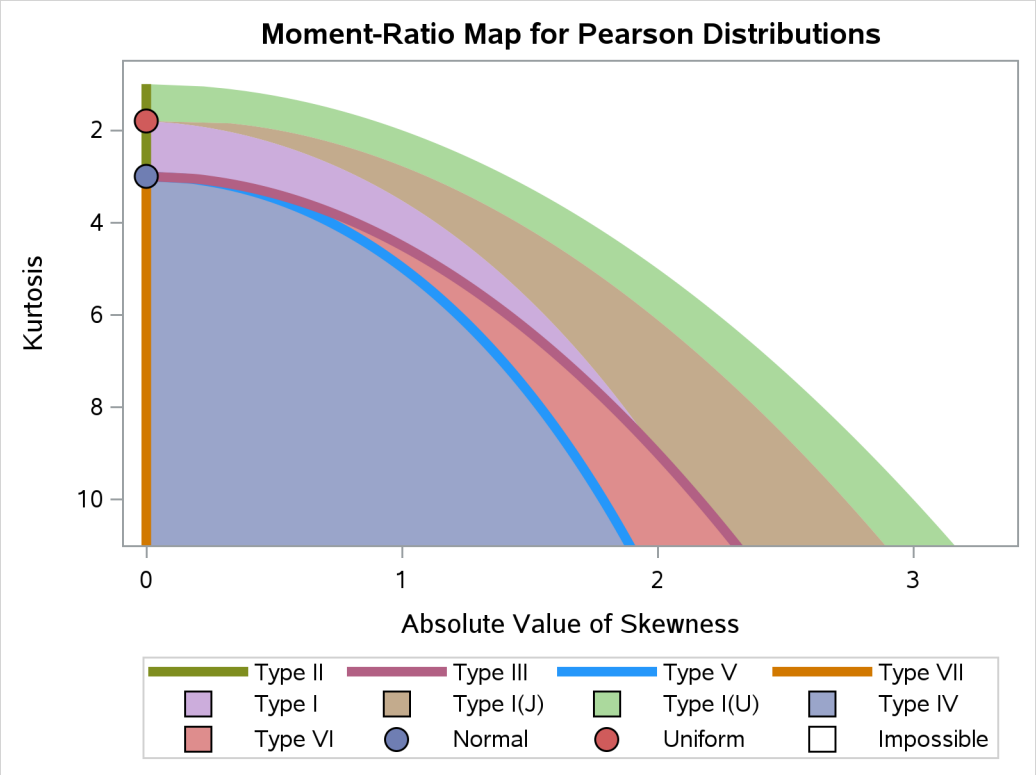
Type	Standard Distribution	Standard Random Variate
Type I	Beta(α,β)	`rand('beta',Shap1Val,Shap2Val)`
Type II	Beta(α,α)	`rand('beta',Shap1Val,Shap1Val)`
Type III	Gamma(α)	`rand('gamma',Shap1Val)`
Type IV		
Type V	Inverse gamma(α)	`1/rand('gamma',Shap1Val)`
Type VI	F_{d_1,d_2}	`rand('f',Shap1Val,Shap2Val)`
Type VII	Student's t_d	`rand('t',Shap1Val)`

Again, the standard random variate for the Type IV distribution is blank because there is no standard distribution that corresponds to this type.

In order to achieve the specified mean and standard deviation, the argument for these standard distributions is further shifted by a value ξ and scaled by a value λ; these values are displayed as the Shift and Scale columns in the "Parameters for Pearson Distributions" table. If the specified value of skewness is negative, then internally, the procedure solves for a distribution that has a mean and skewness of the opposite sign, and it adjusts the sign of ξ and λ values to produce a distribution that has the specified moments.

The manner in which the distributions in the Pearson system partition the space of possible skewness/kurtosis values is shown in Figure 31.10.

Figure 31.10 Pearson Distributions



Johnson System of Distributions

The Johnson system of continuous distributions was introduced in 1949 by Norman L. Johnson, who referred to it as the translation system (Johnson 1949). Johnson derived the system by constructing a set of transformations (translations) to normality. For more information about the derivation, see Chapter 7 of Elderton and Johnson (1969) and Chapter 12 of Johnson, Kotz, and Balakrishnan (1994). Rodriguez (2006) and Bowman and Shenton (2006) provide overviews of the Johnson system.

There are four families of distributions in the Johnson system, each corresponding to a transformation $f(X)$ of a random variable X such that

$$Z = \gamma + \delta f\left(\left(Y - \xi\right)/\lambda\right)$$

has a standard normal distribution.

- The S_N family is generated by the identity transformation $f(X) = X$ and is simply the family of normal distributions with mean ξ and standard deviation λ.
- The S_L family is generated by the transformation

  $$f(X) = \log\left(X\right)$$

  and is the family of lognormal distributions with the shape parameter $1/\delta$.

- The S_B family is generated by the transformation

$$f(X) = \log\left(X + \sqrt{1 + X^2}\right) = \sinh^{-1}(X)$$

and has the shape parameters γ and δ.

- The S_U family is generated by the transformation

$$f(X) = \log\left(\frac{X}{1 - X}\right)$$

and has the shape parameters γ and δ.

Johnson S_N Family

The probability density function for the S_N family, which consists of the normal distributions, is

$$p(y) = \frac{1}{\sqrt{2\pi}\lambda} \exp\left[-\frac{1}{2}\left(\frac{y - \xi}{\lambda}\right)^2\right], \quad -\infty < y < \infty$$

where $\lambda > 0$.

For S_N distributions, $\sqrt{\beta_1} = 0$ and $\beta_2 = 3$.

Johnson S_L Family

The probability density function of the S_L family is

$$p(y) = \frac{\delta}{(y - \xi)\sqrt{2\pi}} \exp\left(-\left(\gamma + \delta \log(y - \xi)\right)^2 / 2\right), \quad y > \xi$$

Johnson S_U Family

The probability density function of the S_U family is

$$p(y) = \frac{\delta}{\lambda\sqrt{2\pi}\sqrt{1 + ((y - \xi)/\lambda)^2}} \exp\left(-\frac{1}{2}\left(\gamma + \delta \sinh^{-1}\left(\frac{y - \xi}{\lambda}\right)\right)^2\right), \quad -\infty < y < \infty$$

Johnson S_B Family

The probability density function of the S_B family is

$$p(y) = \frac{\delta\lambda}{\sqrt{2\pi}\,(y - \xi)\,(\xi + \lambda - y)} \exp\left(-\frac{1}{2}\left(\gamma + \delta \log\left(\frac{y - \xi}{\xi + \lambda - y}\right)\right)^2\right), \quad \xi < y < \xi + \lambda$$

The S_B distribution is bounded below by the parameter ξ and above by the value $\xi + \lambda$.

Parameter Values for Johnson Distributions

The purpose of this section is to explain in detail how parameter values and transformation families that the procedure computes for the Johnson system can be used to transform the normal distribution into variates that have the specified moments. The tables that are provided are helpful for understanding what the procedure is computing in each particular case, and they are also helpful if you want to reuse parameter values from the procedure to compute your own simulations in DATA step code by using the RAND function.

The density functions for the Johnson distributions are shown in Table 31.4. As indicated, the SB and SU types (which PROC SIMSYSTEM uses to denote the S_B and S_U families) do not correspond to any familiar distribution.

Table 31.4 Johnson Distributions for $y = (x - \xi)/\lambda$

Type	Standard Distribution	Probability Density Function
SB		$(\delta/\sqrt{2\pi})\exp(-(\gamma + \delta\log(y/(1-y))/2)^2)/(y(1-y))$
SL	Lognormal	$(\delta/\sqrt{2\pi})\exp(-(\gamma + \delta\log(y)/2)^2)/y$
SU		$(\delta/\sqrt{2\pi})\exp(-(\gamma + \delta\sinh^{-1}(y)/2)^2)/\sqrt{1+y^2}$

PROC SIMSYSTEM displays the specific δ and γ parameter values for achieving the values of skewness and kurtosis that you specify in the "Parameters for Johnson Distributions" table. For illustration, Table 31.5 shows DATA step code that is equivalent to how PROC SIMSYSTEM generates random variates for these distributions, after it determines the appropriate values of δ and γ.

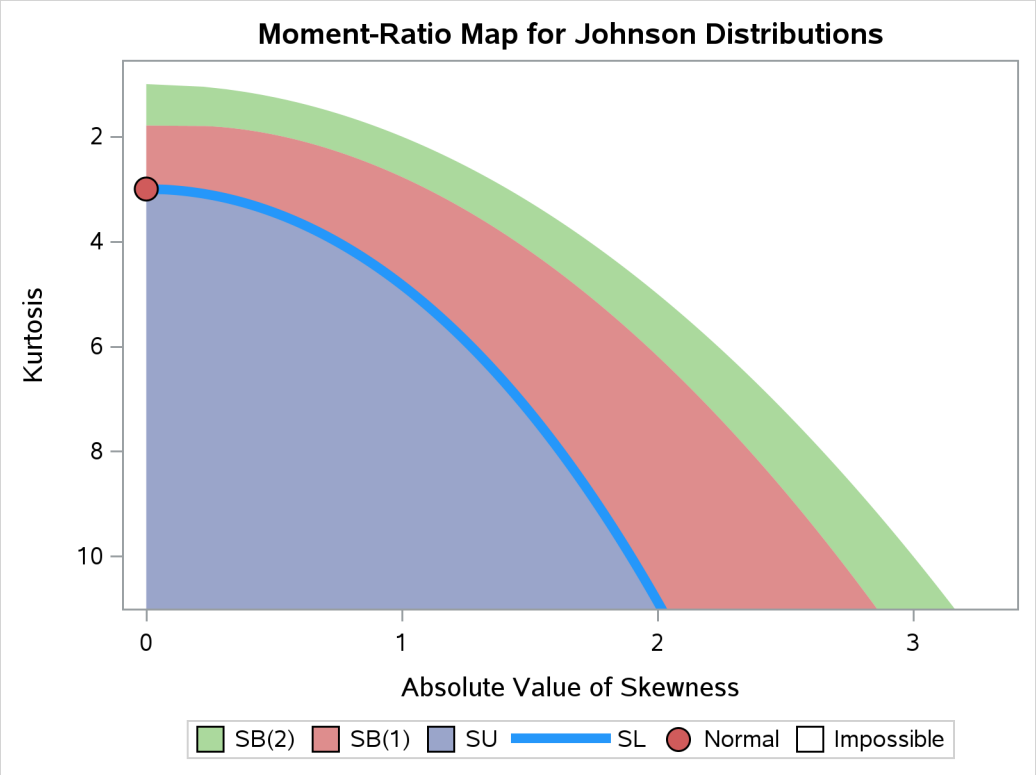
Table 31.5 Johnson Sampling for $y = (x - \xi)/\lambda$

Type	Standard Random Variate
SB	`1/(1 + exp(-(rand('Normal') - Gamma)/Delta))`
SL	`exp(rand('Normal')/Delta)`
SU	`sinh((rand('Normal') - Gamma)/Delta)`

In order to achieve the specified mean and standard deviation, the results of the computations in Table 31.5 are further scaled by a value λ and shifted by a value ξ; these values are displayed as the Shift and Scale columns in the "Parameters for Johnson Distributions" table. If the specified value of skewness is negative, then internally, the procedure solves for a distribution that has a mean and skewness of the opposite sign, and it adjusts the sign of ξ and λ to produce a distribution that has the specified moments.

The way that the distributions in the Johnson system partition the space of possible skewness/kurtosis values is shown in Figure 31.11.

Figure 31.11 Johnson Distributions



Specifying Boundary Widths for Transition Families

Most distributions in both the Pearson and Johnson systems have two parameters on which the skewness and kurtosis depend. The nature of these two parameters depends on the particular distribution family. However, the distributions on the boundaries between the skewness/kurtosis regions of different families can be characterized in a simpler fashion, using only a single parameter. These simpler families are as follows:

- Pearson Type V, between Pearson Types IV and VI
- Pearson Type III, between Pearson Types VI and I
- Johnson SL, between Johnson SB and Johnson SU

Moreover, near these boundaries, the underlying distributional parameters for the more complicated families can grow without bound and calculations can be unstable.

For these reasons, when you specify skewness and kurtosis values that are within a certain tolerance τ of one of these boundaries, PROC SIMSYSTEM generates and samples from a nearby distribution that is actually on the boundary. By default, $\tau = 10^{-4}$; you can adjust this value by using the SIMPLETOL= option in the PROC SIMSYSTEM statement.

Shapes of Pearson and Johnson Density Functions

This section discusses the variety of distribution shapes that both the Pearson and Johnson systems can cover. Figure 31.12 shows the shapes of selected distributions with a skewness of 0, for which the distributions in both systems are symmetric. Pearson distributions move smoothly from the limiting case of a fair-coin binary at a kurtosis of 1, through uniform at kurtosis 1.8 and normal at kurtosis 3, to t distributions with DF $\rightarrow 4$ as the kurtosis goes to ∞. Johnson distributions look similar, but they do not have precisely the same shapes as the familiar distributions in the Pearson system.

Figure 31.12 Centered and Scaled Densities for Skewness 0

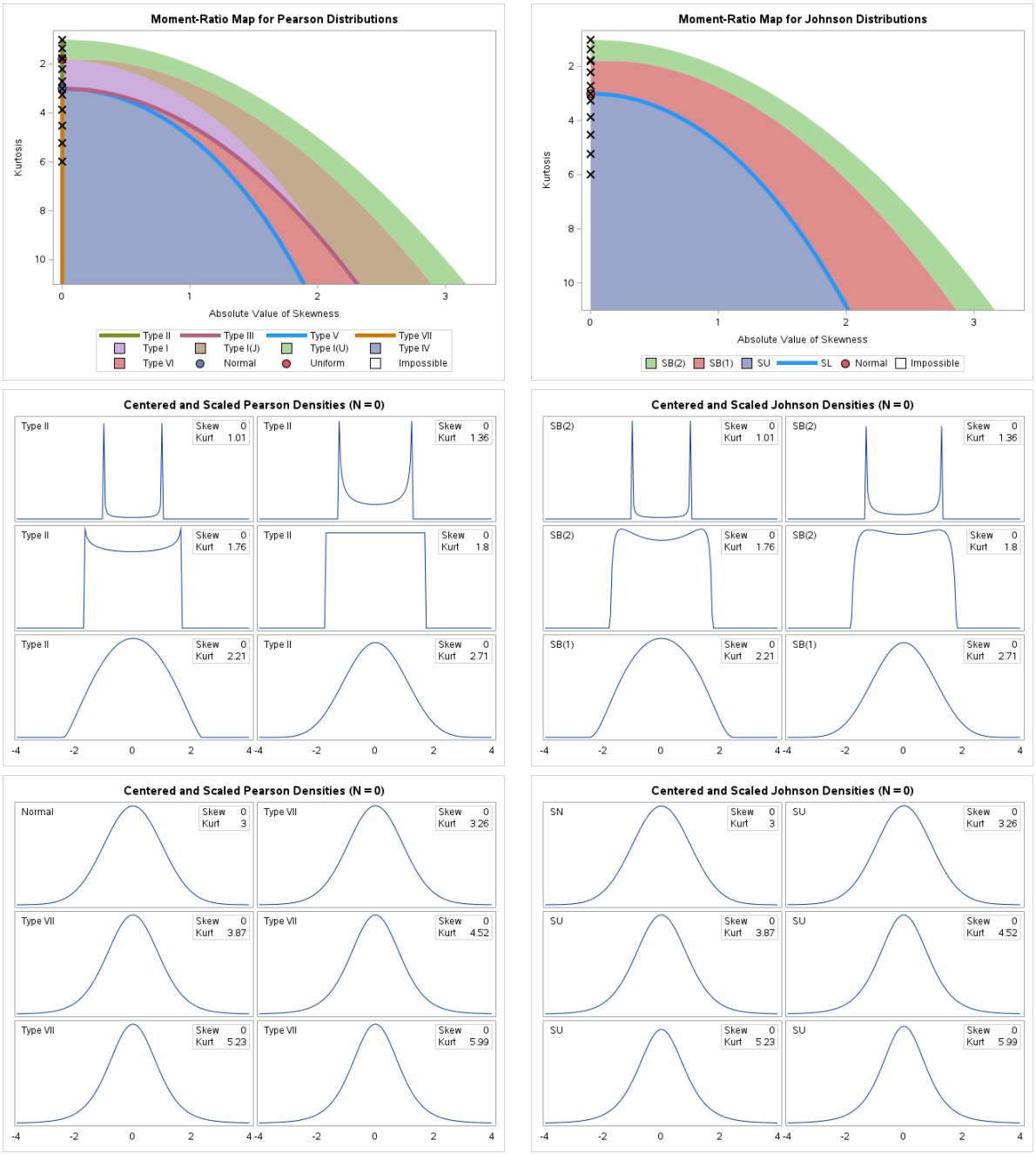


Figure 31.13 shows the shapes of selected distributions with skewness 1. When skewness is nonzero, the shapes in Figure 31.12 are all tilted to the left. For both systems, the fair-coin binary becomes the distribution of a biased coin flip, the uniform becomes approximately a triangular distribution, and for larger kurtosis you essentially get skewed t distributions.

Figure 31.13 Centered and Scaled Densities for Skewness 1

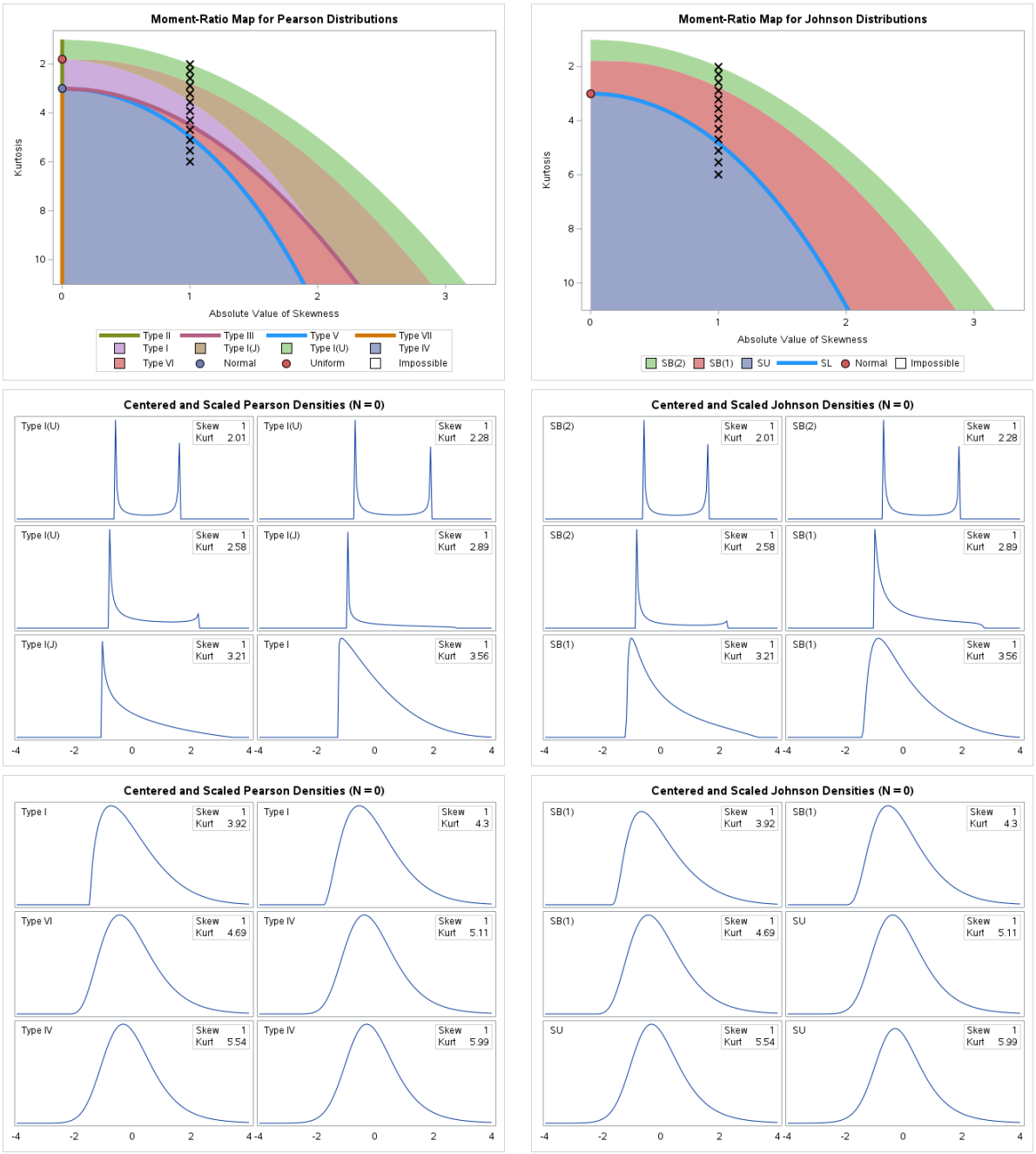
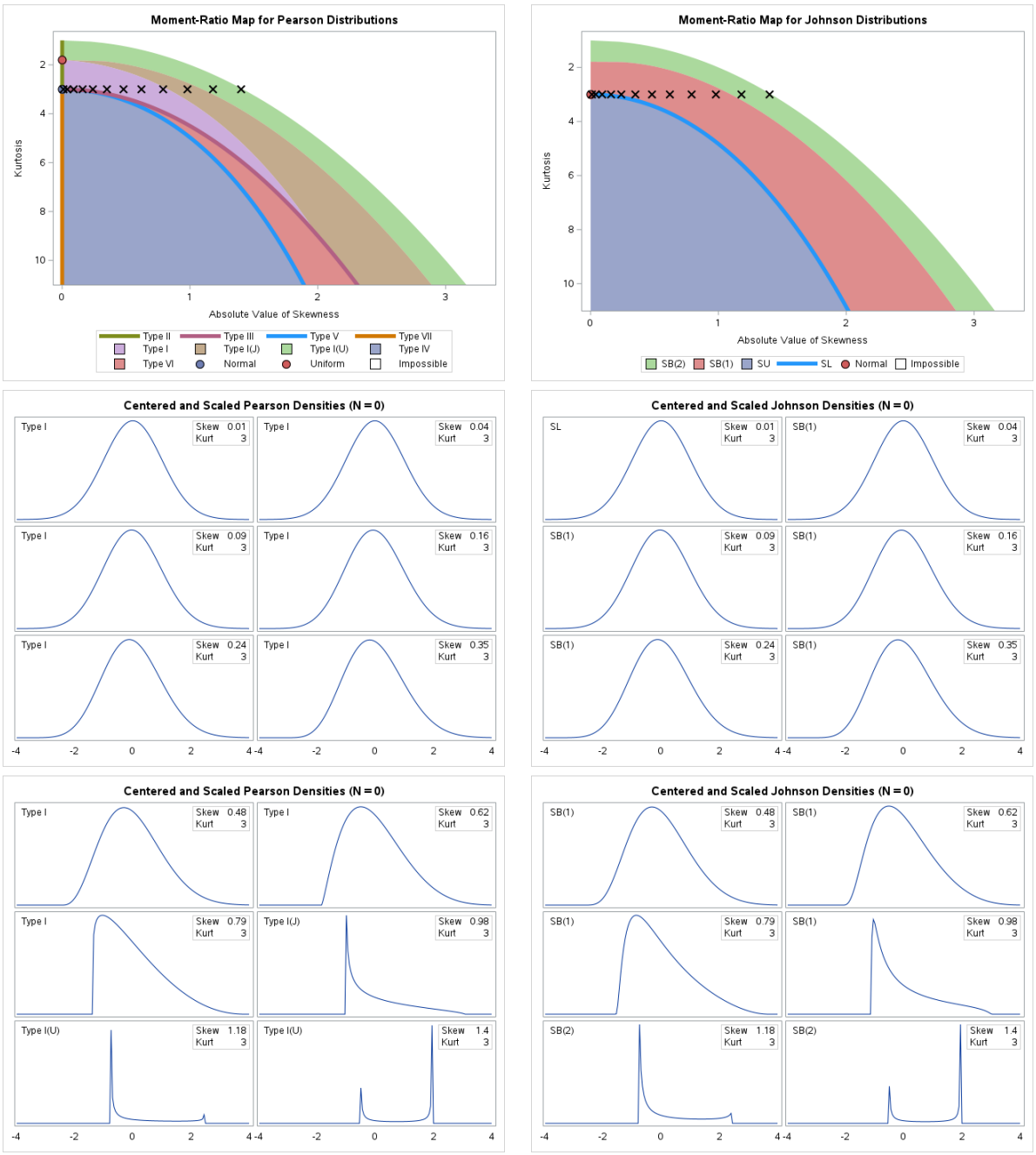


Figure 31.14 shows the shapes of selected distributions with kurtosis 3. At skewness 0, you get the normal distribution for both the Pearson and Johnson systems. As skewness increases to its maximum value of $\sqrt{3-1} \sim 1.4$, the distributions tilt more and more to the left, until they again approximate an unfair-coin binary.

Figure 31.14 Centered and Scaled Densities for Kurtosis 3



Output Data Table

The OUTPUT statement produces a CAS data table that contains simulated values from distributions in the specified system that have the specified moments. The data table contains the following variables:

- SimIndex, a numerical index for the distribution being sampled. You can use the SIMIDX= option in the OUTPUT statement to change the name of this variable.
- Rep, a numerical index for a replicate of the distribution being sampled. This variable is included in the data table only when you specify the NREP= option in the PROC statement. You can use the REPIDX= option in the OUTPUT statement to change the name of this variable.
- iObs, the observation number of a simulated value within the distribution being sampled. If you specify the NREP= option in the PROC statement, this is the number of an observation within a replicate of the distribution. You can use the IOBS= option in the OUTPUT statement to change the name of this variable.
- Mean, the mean value for the distribution being sampled. This variable is included in the output data table only when one or more of the following conditions is true:
  - The MEAN= option is specified in a MOMENTS or MOMENTGRID statement.
  - The variable Mean is included in the data table that is specified in the DATA= option.
  - The MEAN option is specified in the OUTPUT statement.

  You can use the MEAN= option in the OUTPUT statement to change the name of this variable.
- StdDev, the standard deviation for the distribution being sampled. This variable is included in the data table only when one or more of the following conditions is true:
  - The STDDEV= option is specified in a MOMENTS or MOMENTGRID statement.
  - The variable StdDev is included in the data table that is specified in the DATA= option.
  - The STDDEV option is specified in the OUTPUT statement.

  You can use the STDDEV= option in the OUTPUT statement to change the name of this variable.
- Skewness, the skewness value for the distribution being sampled. You can use the SKEW= option in the OUTPUT statement to change the name of this variable.
- Kurtosis, the kurtosis value for the distribution being sampled. You can use the KURT= option in the OUTPUT statement to change the name of this variable.
- Variate, a sampled values from the distribution. You can use the SIMVALUE= option in the OUTPUT statement to change the name of this variable.

Displayed Output

For each set of valid input moments that you specify, PROC SIMSYSTEM displays a table that contains the parameters of the distribution in the specified system that has those moments. The table of parameters includes the specified skewness and kurtosis values, the distribution family within the specified system, and the values Shift and Scale by which to transform the sampled values of the distribution in that family in order to achieve the specified mean and standard deviation. The table also includes columns for the underlying parameters of the distribution family. For the default Pearson system, a column of distribution notes displays these values along with the name of the common distribution for the family, except for the Pearson Type IV distribution, which corresponds to no common distribution. For the Johnson system, PROC SIMSYSTEM displays the values of the parameters δ and γ that define the translation of the underlying normal variate. For more information about the two systems, see "Details: SIMSYSTEM Procedure" on page 1597.

If you use the N= option to specify a number of simulated values to sample for each distribution, then PROC SIMSYSTEM also displays a table of specified and computed moments for those samples.

ODS Table Names

PROC SIMSYSTEM assigns a name to each table that it creates. You can use these names to refer to the tables when you use the Output Delivery System (ODS) to select tables and create output data tables. These names are listed in Table 31.6.

Table 31.6 ODS Tables Produced by PROC SIMSYSTEM

ODS Table Name	Description	Statement
Parameters	Type and parameters of specified distributions	Default
Moments	Specified and observed moments for random sample	Default

ODS Graphics

Statistical procedures use ODS Graphics to create graphs as part of their output. ODS Graphics is described in detail in the "Statistical Graphics Using ODS" chapter in *SAS/STAT User's Guide*.

Before you create graphs, ODS Graphics must be enabled (for example, by using the ODS GRAPHICS ON statement). For more information about enabling and disabling ODS Graphics, see the section "Enabling and Disabling ODS Graphics" in that chapter.

The overall appearance of graphs is controlled by ODS styles. Styles and other aspects of using ODS Graphics are discussed in the section "A Primer on ODS Statistical Graphics" in that chapter.

PROC SIMSYSTEM assigns a name to each graph that it creates using ODS. You can use these names to refer to the graphs when you use ODS. The names are listed in Table 31.7. ODS Graphics must be enabled before you request plots.

Table 31.7 Graphs Produced by PROC SIMSYSTEM

ODS Graph Name	Plot Description	Option
DensityPlot	Expected density and the observed histogram	Default
JohnsonMap	Moment-ratio map for Johnson system	SYSTEM=JOHNSON
PearsonMap	Moment-ratio map for Pearson system	SYSTEM=PEARSON (default)

Examples: SIMSYSTEM Procedure

The examples in this chapter are available in the GitHub repository located at `https://github.com/sascommunities/doc-supplement-casstat`.

Example 31.1: Estimating the Parameters of Johnson Distributions

(View the complete code for this example (ssyex1.sas) in the example repository.)

The UNIVARIATE procedure in Base SAS software can fit the parameters of all three distribution families in the Johnson system. This example demonstrates how to use PROC SIMSYSTEM in conjunction with PROC UNIVARIATE to study these fits.

Suppose you want to work with data from a Johnson SB distribution that has true moments as given in Table 31.8.

Table 31.8 Moments for a Johnson SB Distribution

Mean	5
Standard deviation	2
Skewness	1
Kurtosis	4.5

The following statements use PROC SIMSYSTEM to generate 10,000 observations of such data and to store them in a data table named SB:

```
proc simsystem system=Johnson n=10000 seed=12345;
   moments mean=5 stddev=2 skew=1 kurt=4.5;
   output out=mylib.SB;
run;
```

The SIMSYSTEM procedure displays the parameters of this distribution in the table shown in Output 31.1.1 and the observed moments of the 10,000 sampled values in the table shown in Output 31.1.2.

Output 31.1.1 Parameters for a Johnson Distribution with Specified Skewness and Kurtosis

The SIMSYSTEM Procedure

Parameters for Johnson Distributions							
	Skewness	Kurtosis	Family	Delta	Gamma	Shift	Scale
1	1	4.500	SB(1)	2.124	4.192	0.120	37.31

Output 31.1.2 Moments for a Johnson Distribution with Specified Skewness and Kurtosis

Distribution Moments								
	Specified Moments				Sample Moments			
	Mean	Std Dev	Skewness	Kurtosis	Mean	Std Dev	Skewness	Kurtosis
1	5	2	1	4.500	5.016	2.007	0.957	4.283

The following statements use PROC UNIVARIATE with the SB option in the HISTOGRAM statement to fit the parameters of a Johnson SB distribution to these data. The default method of fitting an SB distribution uses the percentiles of the data, but in this case, because the moments of the distribution are of particular interest, you use the method of moments to fit the distribution.

```
proc univariate data=mylib.SB;
   var variate;
   histogram / sb(theta=est sigma=est fitmethod=moments) noplot;
run;
```

The part of the PROC UNIVARIATE output that shows the fitted parameters for the SB distribution is shown in Output 31.1.3.

Output 31.1.3 Fitted Parameters for a Johnson SB Distribution

The UNIVARIATE Procedure
Fitted SB Distribution for Variate (Random Variate)

Parameters for Johnson SB Distribution		
Parameter	Symbol	Estimate
Threshold	Theta	0.210058
Scale	Sigma	30.82224
Shape	Delta	2.007138
Shape	Gamma	3.555699
Mean		5.016387
Std Dev		2.006555
Skewness		0.956928
Kurtosis		1.283402
Mode		4.021034

As expected from a fit that uses the method of moments, the values of the moments in Output 31.1.3 match those for the sample in Output 31.1.2. The caveat is that PROC SIMSYSTEM works with the direct or raw kurtosis, whereas PROC UNIVARIATE prints the excess kurtosis, which is the raw value minus 3. The values of the four fitted parameters of the distribution match up reasonably well with the true values that they

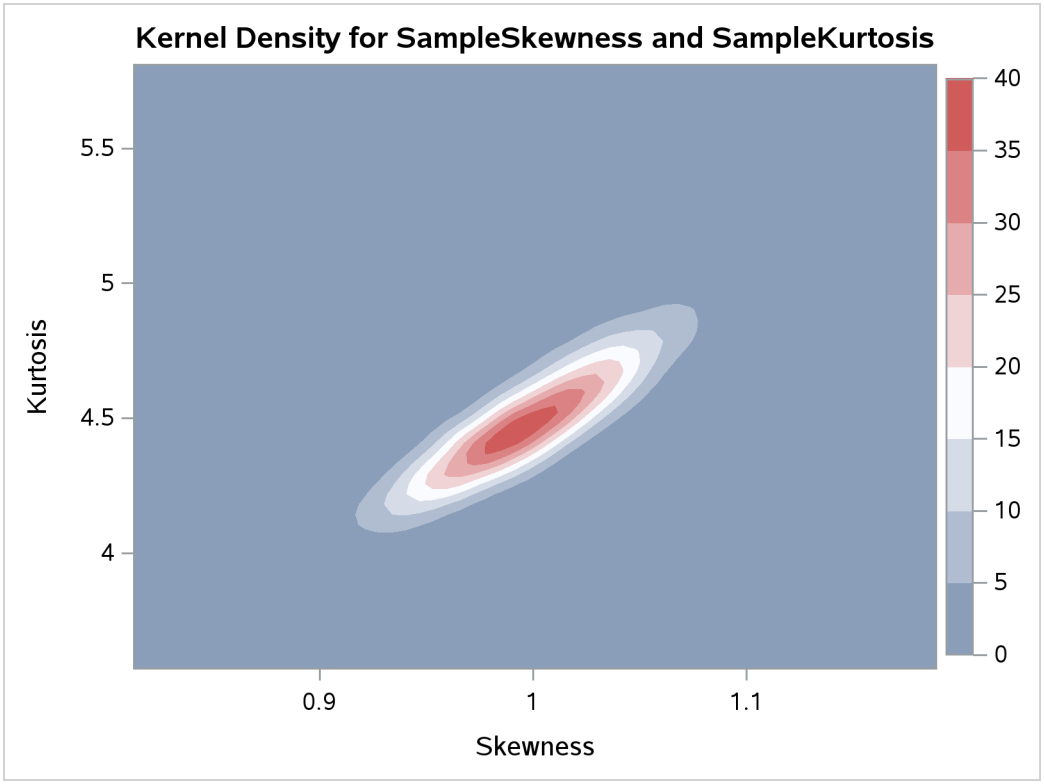
correspond to in the PROC SIMSYSTEM output, with PROC UNIVARIATE's Theta corresponding to PROC SIMSYSTEM's Shift and Sigma corresponding to Scale. Delta and Gamma denote the same parameters in both procedures.

It is not at all surprising that, although the sample moments in Output 31.1.2 are fairly close to the specified moments, they do not match very precisely, and the sample skewness and kurtosis are a good deal less precise than the mean and standard deviation. You can explore how far off they are likely to be by using the following statements to simulate 10,000 replicates of this 10,000-observation sample, by using the NREP=10000 option in the PROC SIMSYSTEM code discussed earlier. What you want to look at in this case is a summary analysis of the 10,000 skewness and kurtosis values for each sample, so you specify the MOMENTREPS option to include all the sample moments in the "Distribution Moments" table, instead of just the default averages, and you also use the ODS SELECT NONE statement to suppress printing this table, instead using the ODS OUTPUT statement to direct it to a data set named Moments. Finally, PROC KDE is used to create a contour plot of the distribution of skewness and kurtosis values.

```
ods select none;
proc simsystem system=johnson n=10000 seed=12345 nrep=10000 momentreps;
   moments mean=5 stddev=2 skew=1 kurt=4.5;
   ods output Moments=Moments;
run;
ods select all;

proc kde data=Moments;
   bivar SampleSkewness SampleKurtosis / plots=contour;
run;
```

Output 31.1.4 Observed Distribution of Sample Skewness and Kurtosis



The resulting density is shown in Output 31.1.4. Perhaps the most notable feature is that, even though the replicated samples are quite large, containing 10,000 observations each, there is considerable variability in the sample skewness and kurtosis values. This explains the magnitude of the default value for the SIMPLETOL= option. When this option comes into play, PROC SIMSYSTEM does not return a sample that has precisely the expected skewness and kurtosis that you specify, but the difference is undetectable in samples of any reasonable size.

Example 31.2: Exploring the Coverage of Three-Sigma Limits

(View the complete code for this example (ssyex2.sas) in the example repository.)

In statistical process control, the three-sigma limits for a random distribution are defined as the mean plus three times the standard deviation and the mean minus three times the standard deviation—symbolically, $\mu \pm 3\sigma$. Three-sigma limits are very commonly used as a rule of thumb for where the vast majority of values from a process should lie; conversely, they are used to say which outlying values might indicate that the process is not in control. Precisely how much of any given distribution is actually covered by the three-sigma limits varies, although by Chebychev's inequality this actual coverage is bounded below by $1 - 1/3^2 \sim 89\%$. This example is patterned after a study of three-sigma coverage by Wheeler (2010).

The following statements use PROC SIMSYSTEM to compute the theoretical three-sigma coverage for a wide variety of distributions. The key element is the CUMPROB= option in the PROC SIMSYSTEM statement, which computes cumulative probabilities.

```
data mylib.Grid;
   do s2 = 0 to 10 by 0.05;
      do Kurtosis = 1 to 11 by 0.05;
         Skewness = sqrt(s2);
         output;
         end;
      end;
run;

ods select none;
proc simsystem data=mylib.Grid system=johnson cumprob=-3 3 plot=none;
   ods output Parameters=SKProbJ;
run;
ods select all;

data SKProbJ; set SKProbJ;
   Coverage = P2 - P1;
   s2 = Skewness*Skewness;
   format Coverage percent6.;
run;
```

The first step creates a fine grid of skewness and kurtosis values. The skewness values are created on a squared scale, to match the results of Wheeler (2010). There are over 40,000 pairs of values in all, although about half of them have kurtosis $\leq$ skewness$^2 + 1$ and thus lie in the "impossible region" for which no distribution is possible. Still, you probably do not want to display all 20,000 of those values, so the subsequent ODS SELECT NONE and ODS SELECT ALL statements suppress this display for the duration of the PROC SIMSYSTEM run, using an ODS OUTPUT statement to save the distribution parameters to a data set named SKProbJ.

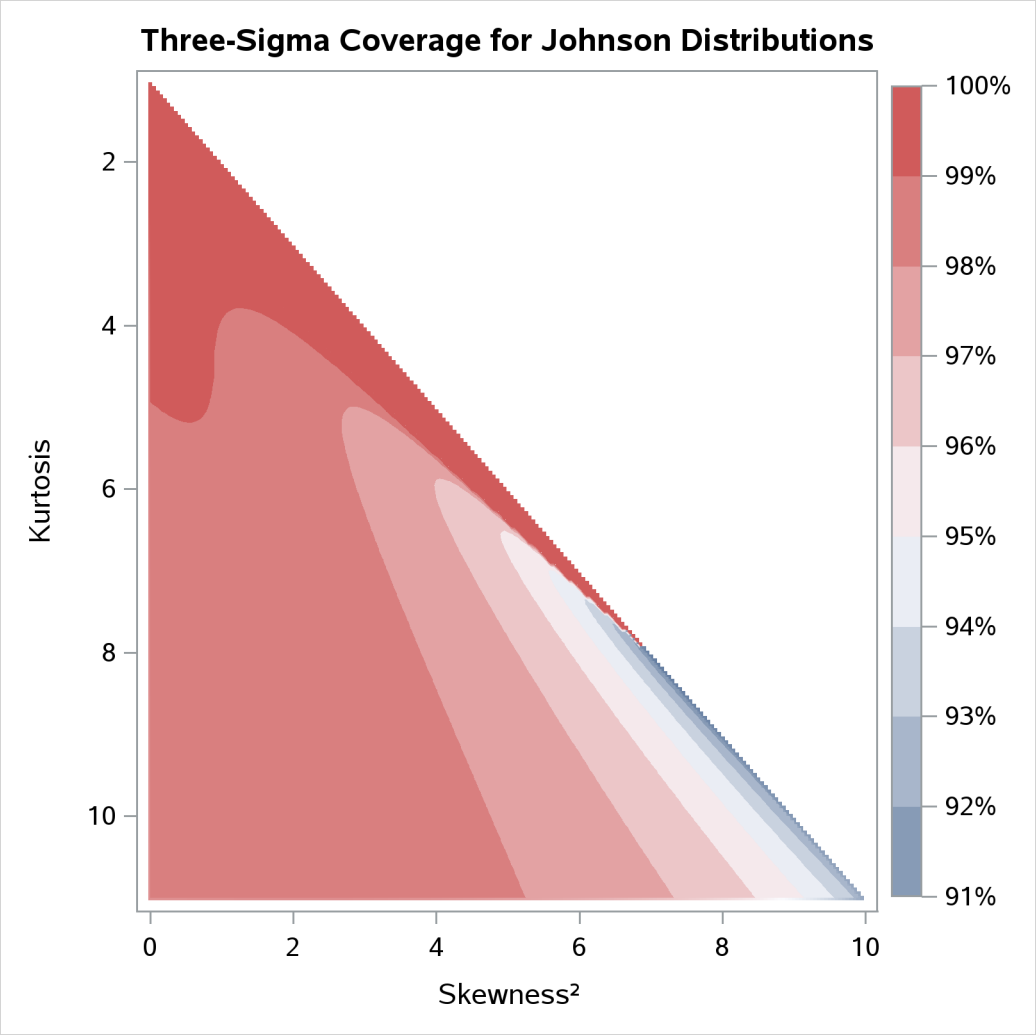
This grid of values is provided as input to PROC SIMSYSTEM. No number of simulations or output data table is specified for the procedure, because you are using the procedure to study the underlying distributions themselves. The SYSTEM=JOHNSON option specifies that this study concerns distributions in the Johnson system; very similar results are obtained using SYSTEM=PEARSON. The CUMPROB= option computes the cumulative probabilities for –3 and 3. Because no value for standard deviation is specified, its value is assumed to be 1 for all distributions, and thus the difference between these two cumulative probabilities is the expected coverage rate. A final DATA step computes this coverage value for each distribution and also recovers the squared skewness values for plotting.

The following statements use PROC SGRENDER to create a plot of the three-sigma coverages computed using the previous statements. They use a predefined graphical template that is specifically designed for plotting the contours of a continuous variable against kurtosis and squared skewness.

```
proc sgrender data=SKProbJ template=acas.simsystem.Graphics.S2KContourMap;
   dynamic _Title = "Three-Sigma Coverage for Johnson Distributions"
           _XVar  = "s2"
           _YVar  = "Kurtosis"
           _ZVar  = "Coverage";
run;
```

The resulting plot is shown in Output 31.2.1.

Output 31.2.1 Exploring the Coverage of Three-Sigma Limits



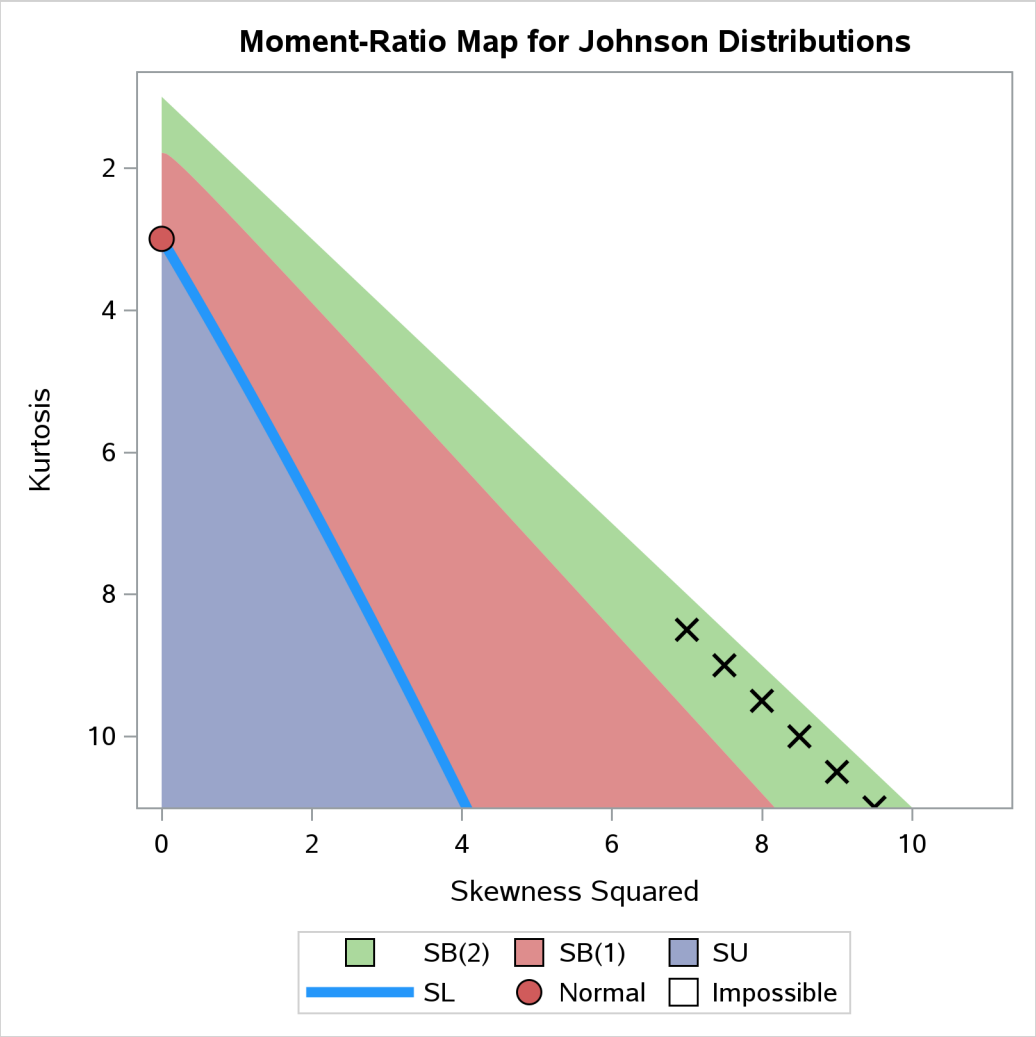
Compare this plot to Figure 3 in Wheeler (2010). As noted there, the coverage is quite high for most of the distributions; only for highly skewed distributions very near the boundary of the "impossible region" does the coverage approach the lower bound of 89%. You can use PROC SIMSYSTEM as follows to see what these distributions look like:

```
data mylib.LowCoverage;
   do s2 = 7 to 10 by 0.5;
      do Kurtosis = 8 to 11 by 0.5;
         Skewness = sqrt(s2);
         if (1 + s2 < Kurtosis <= 1 + s2 + 0.5) then
            output;
         end;
      end;
run;

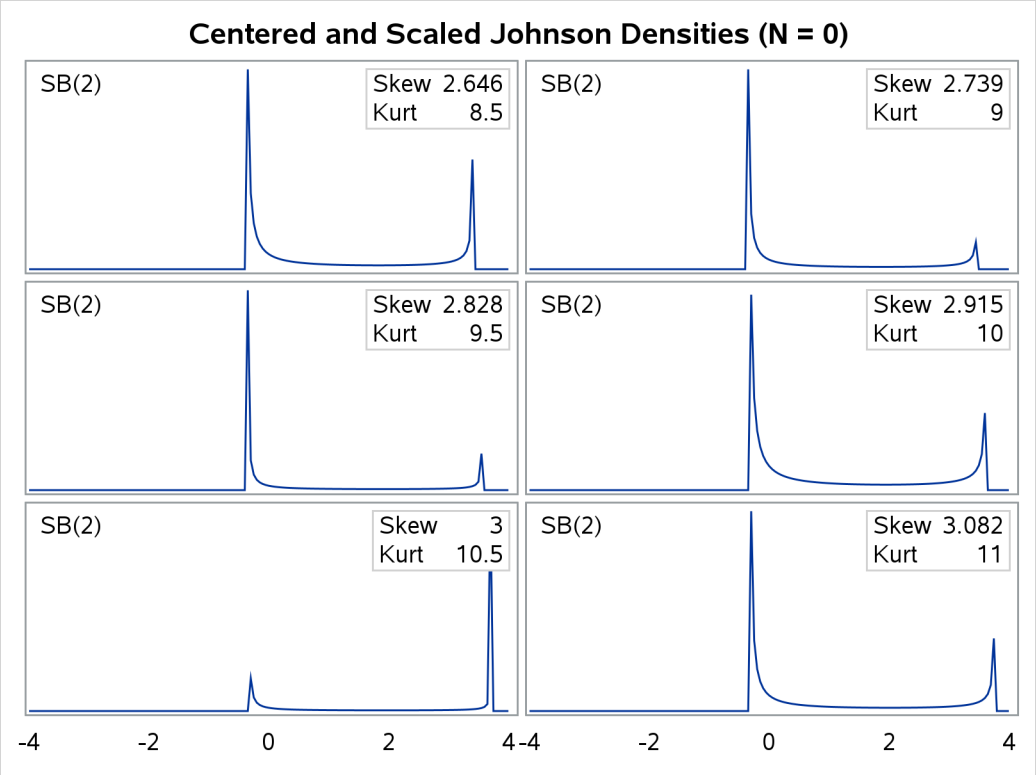
proc simsystem data=mylib.LowCoverage system=johnson
               plots=mrmap(skewscale=square);
run;
```

The DATA step creates another grid of skewness and kurtosis values, which are filtered to lie in the small area of low-coverage distributions shown in Output 31.2.1. The plots that this PROC SIMSYSTEM code produces are shown in Output 31.2.2 and Output 31.2.3.

Output 31.2.2 Low-Coverage Johnson Distribution Map



Output 31.2.3 Low-Coverage Johnson Distribution Densities



Output 31.2.2 confirms that these distributions do indeed lie in the low-coverage region of interest. From Output 31.2.3 you can see that they are all quite extreme. Technically they are "U-shaped" distributions in Johnson's SB family, but in all cases at least one of the two modes is very sharp. As noted by Wheeler (2010), if your process really does have a distribution like this, then it probably indicates an underlying mixture distribution.

Example 31.3: Simulating Portfolio Values

(View the complete code for this example (ssyex3.sas) in the example repository.)

This example demonstrates how you can use PROC SIMSYSTEM to perform an experimental simulation over a grid of potential distributions. The procedure plays two different roles in this application: screening the shapes of potential distributions and simulating samples from selected distributions. Other SAS tools (the DATA step, PROC SUMMARY, and PROC RSREG) are used to complete and analyze the simulation.

You are interested in studying the likely performance over the next 30 years of a $1,000,000 portfolio that is managed to track the performance of the S&P 500 stock index. Of course, key information that you are missing is the future performance of the S&P 500, but you can simulate it for a variety of potential distributions.

Your first question is, What are the potential shapes for the true distribution of total annual returns of the S&P 500? You can use PROC SIMSYSTEM to get an idea of this, by specifying several potential values for skewness and kurtosis that seem feasible according to historical data and reviewing the density plots for them, as shown in the following code:

```
proc simsystem system=johnson;
   momentgrid Skewness = -1.0 -0.5 0.0
              Kurtosis =  2.3  3.1 3.9;
run;
```

The table of parameters for these distributions, shown in Output 31.3.1, indicates that they are mostly of the bounded type. Also, as you can see in Output 31.3.2, most of the densities seem reasonable, except for the first one, with the lowest values of both skewness and kurtosis.

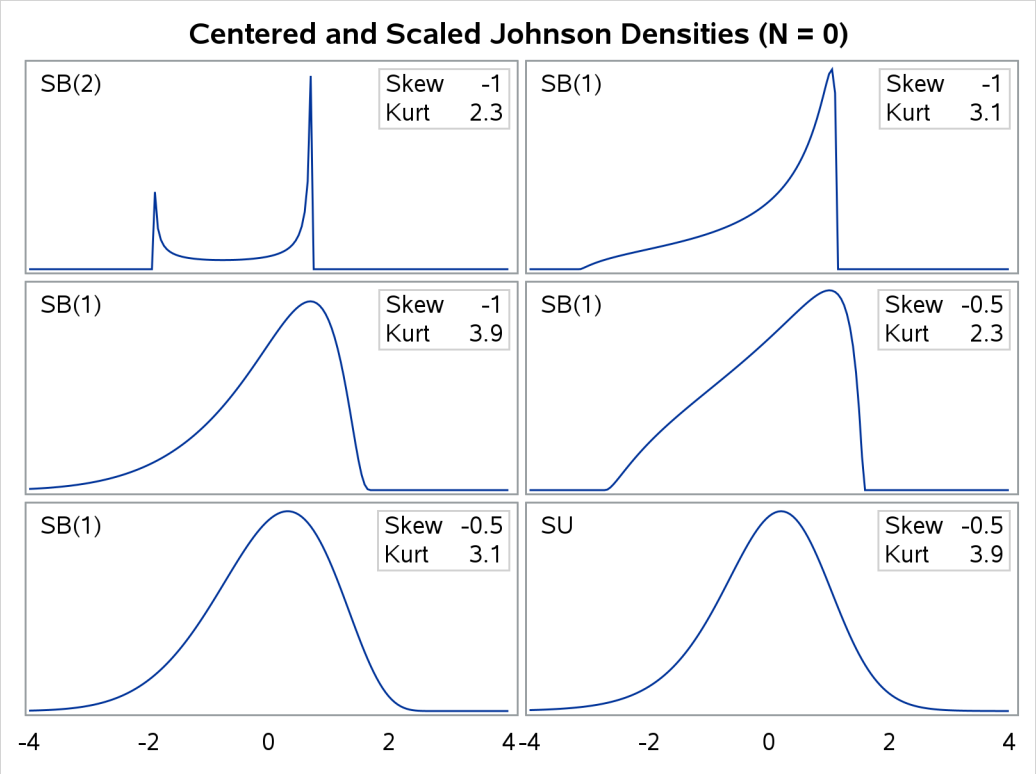
Output 31.3.1 Parameters of Representative Distributions for S&P 500 Returns

The SIMSYSTEM Procedure

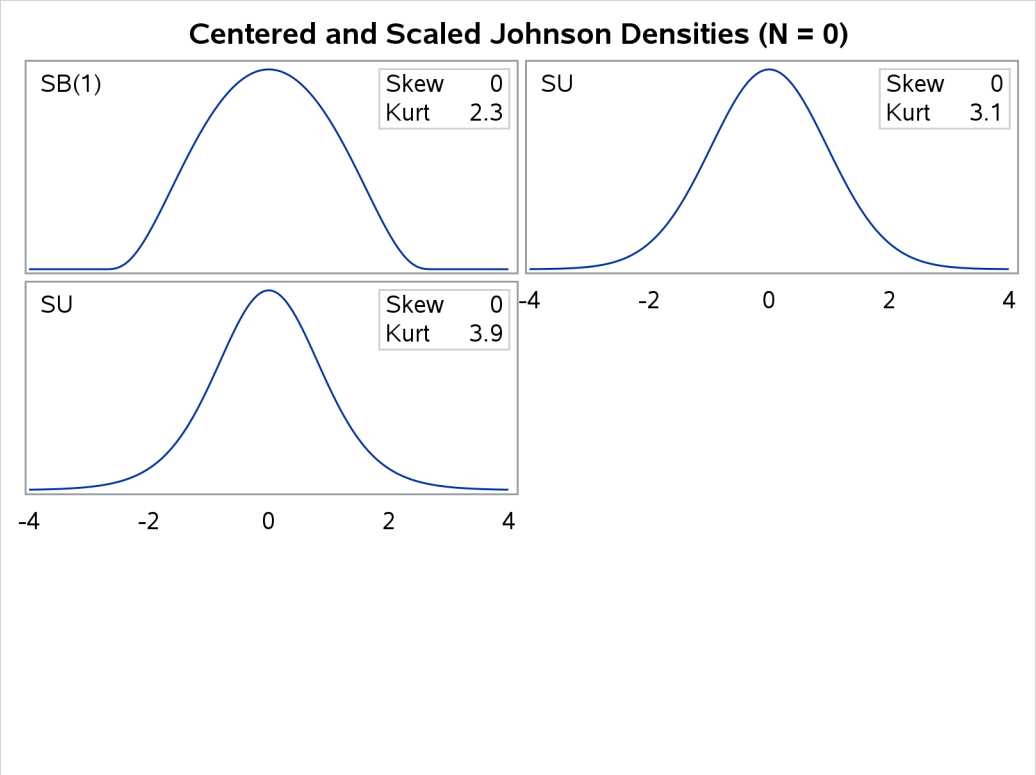
Parameters for Johnson Distributions

	Skewness	Kurtosis	Family	Delta	Gamma	Shift	Scale
1	-1	2.300	SB(2)	0.190	0.627	0.739	-2.674
2	-1	3.100	SB(1)	0.647	0.931	1.146	-4.311
3	-1	3.900	SB(1)	1.259	1.779	1.730	-7.785
4	-0.500	2.300	SB(1)	0.799	0.565	1.598	-4.346
5	-0.500	3.100	SB(1)	2.109	1.917	3.109	-10.49
6	-0.500	3.900	SU	2.931	-1.378	1.276	-2.469
7	0	2.300	SB(1)	1.225	19E-15	-2.796	5.592
8	0	3.100	SU	6.441	0	0	6.363
9	0	3.900	SU	2.417	0	0	2.213

Output 31.3.2 Representative Densities for S&P 500 Returns



Output 31.3.2 *continued*



The extreme density in Output 31.3.2 is omitted by a PROC DS2 step after the PROC SIMSYSTEM code, as shown in the following statements. The grid discussed earlier is expanded with typical values for the mean and standard deviation of annual S&P 500 returns. The code creates, for each set of moment values, 1,000 replicates of potential sequences of N=30 years of total annual returns, using the Johnson system of distributions. You will use these potential sequences of returns to explore the success rate of your portfolio for each distribution, as defined by the moments.

```
proc simsystem n=30 system=johnson seed=12345 noprint nrep=1000;
   momentgrid
      Mean     =   7.5 12.0 16.5
      StdDev   =  17.0 20.0 23.0
      Skewness =  -1.0 -0.5  0.0
      Kurtosis =   2.3  3.1  3.9;
   output out=mylib.Sim_raw Mean=Mean Std=StdDev;
run;

proc ds2 sessref=mysess;
   data Sim / overwrite=yes;
      rename Variate=AnnualReturn;
      method run();
         set Sim_raw;
         if ^((Skewness = -1) & (Kurtosis = 2.3)) then output;
      end;
   enddata;
run;
quit;
```

The resulting data table has over 2,000,000 observations. These observations will be processed in groups defined by the simulation index SimIndex and the replicate number Rep, each group being a potential sequence of returns for the hypothetical 30 years of interest. The value of your portfolio for any particular sequence of returns is adjusted annually by the simulated return and also by a fixed withdrawal of $40,000 per year. The sequence is successful for your portfolio if its value never falls below $0. The following PROC DS2 code evaluates the success of each simulated sequence of 30 returns, using a thread block to process different replicates in parallel. A subsequent PROC MDSUMMARY step computes the mean of these 0/1 success values, which is thus the observed success rate for each set of moment values that are defined by the MOMENTGRID statement.

```
%let Start    = 1000000;
%let Withdraw =   40000;
proc ds2 sessref=mysess;
   thread Success_thd / overwrite=yes;
      dcl double Value minVal;
      dcl int Success;
      retain Value minVal;
      keep SimIndex Mean StdDev Skewness Kurtosis Rep Success;
      method run();
         set Sim;  /*being parallelized*/
         by SimIndex Rep;
         /*BY statement takes the BY-groups as batches in threads*/
         if FIRST.Rep then do;        /* First year in a sequence: */
            Value  = 1000000;         /* Initialize the portfolio  */
            minVal = Value;           /* value.                    */
         end;
         /*
         /  The value at the end of the year is the sum of the
         /  original value and the annual return, minus the amount
         /  withdrawn.
         /-----------------------------------------------------*/
         if Value then do;
            Value  = Value + (AnnualReturn/100)*Value - 40000;
            minVal = min(minVal, Value);
         end;
         if LAST.Rep then do;         /* Last year in a sequence:  */
            Success = (minVal > 0);   /* Evaluate success and      */
            output;                   /* output.                   */
         end;
      end;
   endthread;
   data Success / overwrite=yes;
      dcl thread Success_thd Success_thd_instance;
      method run();
         set from Success_thd_instance threads=1;
      end;
   enddata;
run;
quit;

proc mdsummary data=mylib.Success;
   groupby Mean StdDev Skewness Kurtosis SimIndex;
```

```
   var Success;
   output out=mylib.ProbSuccess;
run;

proc cas;
   session mysess;
   table.alterTable /
      name="ProbSuccess"
      columns={{name="_Mean_" rename="ProbSuccess" label=""}};
quit;
```

Finally, you use PROC RSREG, as in the following statements, to apply response surface analysis methods to the simulated success rates. PROC RSREG fits a quadratic function to these values and plots the predicted surfaces.

```
proc rsreg data=mylib.ProbSuccess plots=contour(nodesign);
   model ProbSuccess = Mean StdDev Skewness Kurtosis;
   ods select FactorANOVA Contour;
run;
```

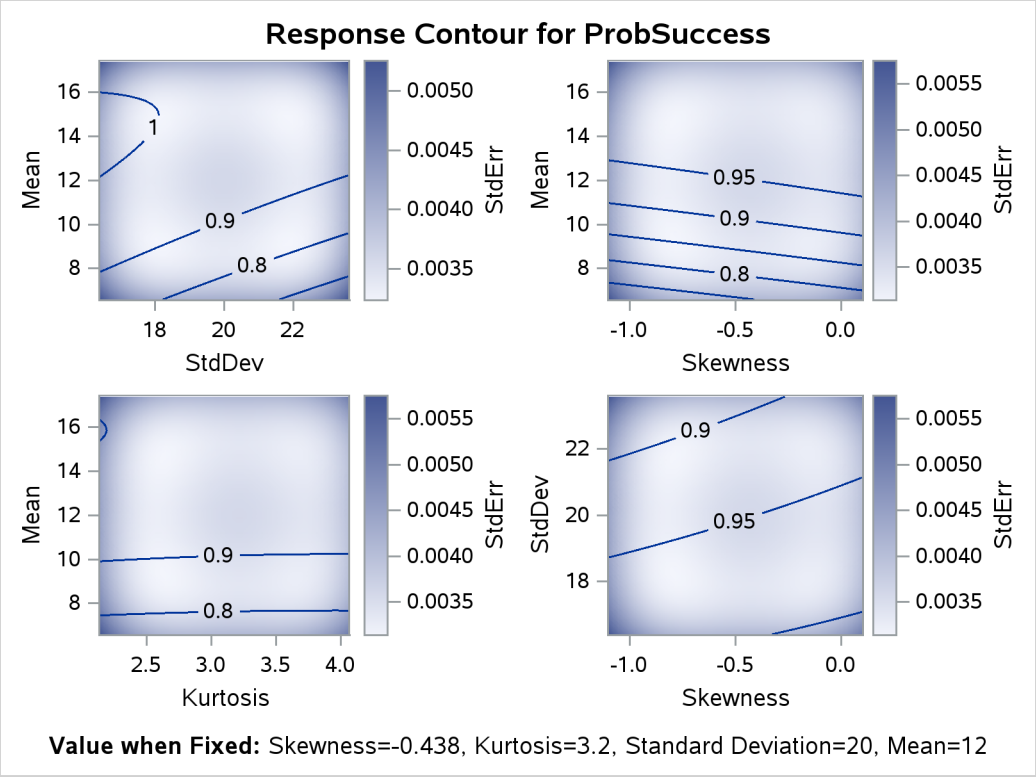
Selected PROC RSREG results are shown in Output 31.3.3 and Output 31.3.4. The ANOVA for individual factors shows that although the mean rate of return is the most influential factor in portfolio success, all four moments of the return distribution have some effect. In the plots of predicted success rates, you can see that a $40,000 withdrawal gives your portfolio about a 95% probability of success in the center of the experimental region, near the historical estimates for the moments of the return distribution. This is encouraging, but the predicted probability of success drops off precipitously as the moments of the return distribution change.

Output 31.3.3 Response Surface Analysis of Simulated S&P 500 Returns

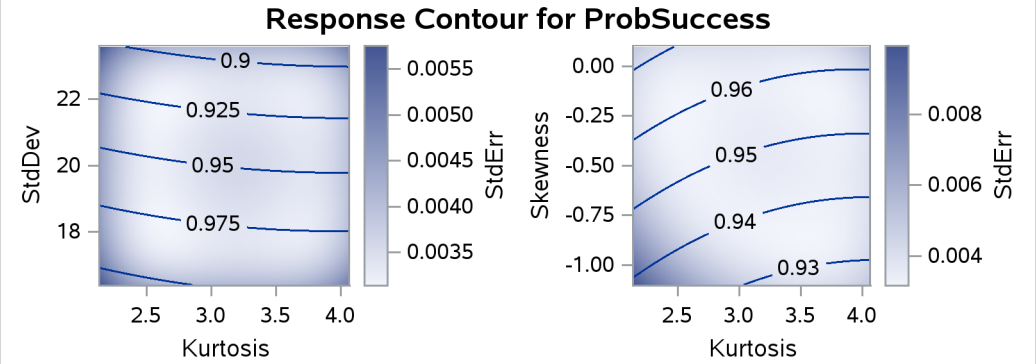
The RSREG Procedure

Factor	DF	Sum of Squares	Mean Square	F Value	Pr > F	Label
Mean	5	0.564070	0.112814	1597.15	<.0001	Mean
StdDev	5	0.141261	0.028252	399.97	<.0001	Standard Deviation
Skewness	5	0.011475	0.002295	32.49	<.0001	Skewness
Kurtosis	5	0.003087	0.000617	8.74	<.0001	Kurtosis

Output 31.3.4 Predicted S&P 500 Returns



Value when Fixed: Skewness=-0.438, Kurtosis=3.2, Standard Deviation=20, Mean=12



Value when Fixed: Mean=12, Skewness=-0.438, Standard Deviation=20

Example 31.4: Simulating Estimation of Value at Risk

(View the complete code for this example (ssyex4.sas) in the example repository.)

The *value at risk* (VaR_τ) of an investment is the most that you can expect to lose in a given future time frame, with a given probability $1 - \tau$. One way to think of this is that if you have $X = \text{VaR}_\tau$ cash on hand, then you can be $100 \times (1 - \tau)\%$ confident that you will be able to cover a loss in that investment. This example uses PROC SIMSYSTEM to explore the accuracy of a common method of estimating the VaR_τ when the assumptions of the method break down.

A popular way to model the periodic returns of an investment is to use a GARCH (generalized autoregressive conditional heteroscedasticity) model, in which, conditional on past returns, future returns are distributed as independent normal values, with a variance that also depends autoregressively on the past. The following code for simulating and fitting a GARCH process is modeled on the first example in SAS Institute Inc. (2019):

```
data Index;
   call streaminit(12345);
   do t = 1 to 10000;
      e = normal(2);
      output;
      end;
run;

data Index; set Index;
   retain ReturnRate Var eLag varLag;
   if (_N_ = 1) then do;
      eLag   = 1;
      VarLag = 1;
      end;
   Var = 0.4 + 0.8 * eLag**2  + 0.1 * VarLag;
   ReturnRate = sqrt(Var) * e;
   output;
   eLag   = ReturnRate;
   varLag = Var;
run;

proc autoreg data=Index;
   model ReturnRate = / noint garch=(p=1, q=1) dist=t;
   output out=o1 alphacli=%sysevalf(2*0.05) lcl=lcl;
run;
```

The OUTPUT statement for PROC AUTOREG computes $-\text{VaR}_{0.05}$ for this sequence of returns as the 95% lower confidence limit for the forecasted returns, and if you use PROC MEANS as follows to count up the proportion of returns that are less than this (that is, that have greater loss than this), you see that this indeed accounts for about 5% of them, as shown in Output 31.4.1:

```
data o1; set o1;
   AtRisk = (ReturnRate < lcl);
proc means data=o1 mean;
   var AtRisk;
run;
```

Output 31.4.1 In-Sample Coverage Rate for GARCH Estimation with Normal Errors

The MEANS Procedure

Analysis Variable : AtRisk
Mean
0.0491000

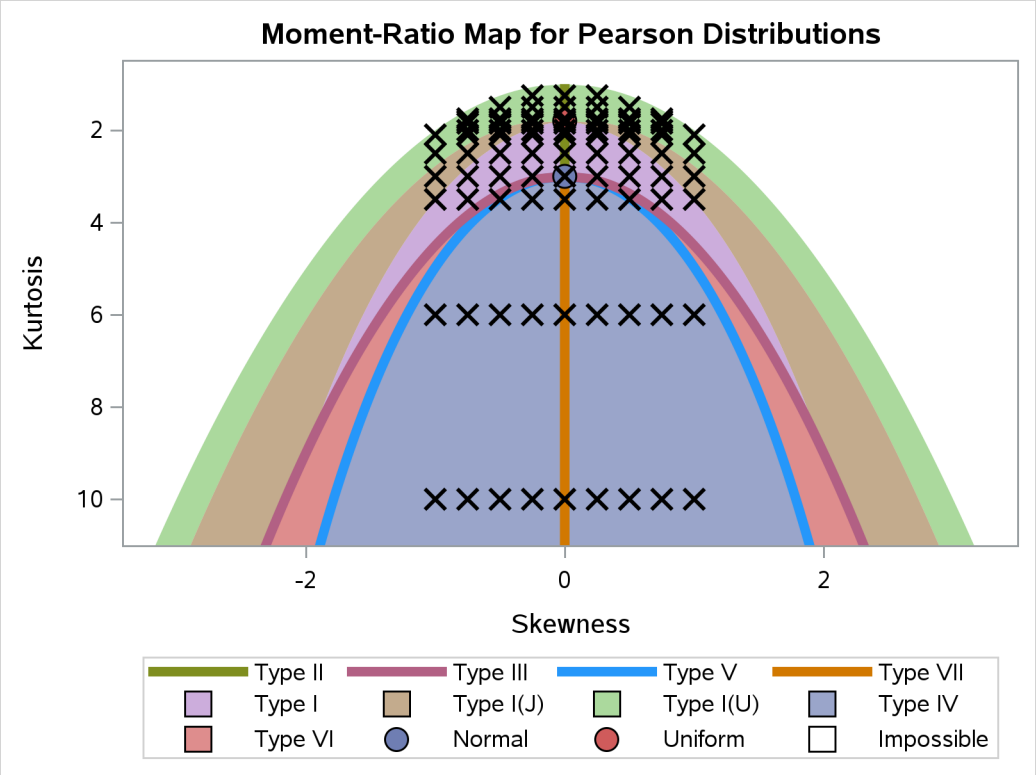
The PROC AUTOREG code given earlier for estimating $-\text{VaR}_{0.05}$ depends on the assumption that the conditional returns are normally distributed, corresponding to the use of the NORMAL() random number generating function in the first DATA step shown previously. How important is this assumption? The following code uses PROC SIMSYSTEM to simulate the random part of 10,000 conditional returns from each of a systematic grid of mostly nonnormal distributions. A subsequent call of the **alterTable** action renames the output variables for convenience.

```
proc simsystem n=10000 system=pearson seed=1 plot(only)=mrmap(skewscale=skewness);
   momentgrid skew=-1 -0.75 -0.5 -0.25 0 0.25 0.5 0.75 1
              kurt=1.25  1.5
                   1.75  1.8
                   1.85  2.0
                   2.1   2.5
                   3     3.5
                   6    10;
   output out=mylib.GARCHSim_error;
run;

proc cas;
   session mysess;
   table.alterTable /
      name="GARCHSim_error"
      columns={
         {name="iObs"    rename="t" label=""},
         {name="Variate" rename="e" label=""}
      };
quit;
```

These statements specify negative as well as positive skewness values and kurtosis values that run from appreciably platykurtic (kurtosis less than 3, the kurtosis of the normal distribution) to appreciably leptokurtic (kurtosis greater than 3). The MOMENTGRID statement specifies a distribution for every pair of these values, although some are ignored because of the constraint that the kurtosis for a random distribution must always be greater than or equal to $1 + \text{skewness}^2$. The SKEWSCALE=SKEWNESS option produces a moment-ratio map that shows both negative and positive skewness values. The placement of this grid of skewness and kurtosis values within the moment-ratio map for the Pearson system is shown in Output 31.4.2.

Output 31.4.2 Systematic Grid of Sampled Errors



The following PROC DS2 code uses these simulated values in the same way that the previous DATA step code used random normal values: to produce many samples of returns. A thread block in the PROC DS2 step processes different samples in parallel. The subsequent code performs the PROC AUTOREG analysis discussed earlier for each sample of 10,000 simulated returns, as indexed by the specified skewness and kurtosis values, and similarly uses PROC SUMMARY to count up the proportion whose loss exceeds the computed $\text{VaR}_{0.05}$.

```
/*using THREAD to parallelize the set statement*/
proc ds2 sessref=mysess;
thread GARCHSim_thd / overwrite=yes;
   dcl double ReturnRate Var eLag VarLag;
   retain ReturnRate Var eLag VarLag;
   method run();
      set GARCHSim_error;
      by SimIndex;
      if FIRST.SimIndex then do;
         eLag   = 1;
         VarLag = 1;
      end;
      Var = 0.4 + 0.8 * eLag**2 + 0.1 * VarLag;
      ReturnRate = sqrt(Var) * e;
      output;
      eLag   = ReturnRate;
      VarLag = Var;
   end;
endthread;
```

```
data GARCHSim / overwrite=yes;
   dcl thread GARCHSim_thd GARCHSim_thd_instance;
   method run();
      set from GARCHSim_thd_instance;
   end;
enddata;
run;
quit;

proc autoreg data=mylib.GARCHSim noprint;
   by notsorted Skewness Kurtosis;
   model ReturnRate = / noint garch=(p=1, q=1);
   output out=mylib.g11 alphacli=%sysevalf(2*0.05) lcl=lcl;
run;

data g11; set mylib.g11;
   AtRisk = (ReturnRate < lcl);
proc summary data=g11 mean;
   class Kurtosis Skewness;
   ways 2;
   var AtRisk;
   output out=AtRisk mean=pAtRisk;
run;
```

Finally, the following code prints these simulated actual coverage values for the nominal $\mathrm{VaR}_{0.05}$, arranging them in a fashion similar to that of the moment-ratio map in Output 31.4.2. The results are shown in Output 31.4.3.

```
proc sort data=AtRisk;
   by Kurtosis Skewness;
proc transpose data=AtRisk(rename=(Skewness=_NAME_)) out=AtRisk;
   by Kurtosis;
   var pAtRisk;
proc print data=AtRisk(drop=_NAME_) noobs label;
   var N1 N0D75 N0D5 N0D25 _0 _0D25 _0D5 _0D75 _1;
   id Kurtosis;
   format _: N: 6.3;
   label
      N1    = "-1"
      N0D75 = "-0.75"
      N0D5  = "-0.5"
      N0D25 = "-0.25"
      _0    =  "0"
      _0D25 =  "0.25"
      _0D5  =  "0.5"
      _0D75 =  "0.75"
      _1    =  "1";
run;
```

Output 31.4.3 In-Sample Coverage Rate for GARCH Estimation with Nonnormal Errors

Kurtosis	-1	-0.75	-0.5	-0.25	0	0.25	0.5	0.75	1
1.25	.	.	.	0.039	0.028	0.017	.	.	.
1.50	.	.	0.049	0.043	0.034	0.026	0.020	.	.
1.75	.	0.055	0.044	0.043	0.038	0.032	0.023	0.013	.
1.80	.	0.057	0.051	0.046	0.035	0.034	0.026	0.015	.
1.85	.	0.052	0.047	0.042	0.036	0.032	0.026	0.017	.
2.00	.	0.052	0.047	0.042	0.036	0.031	0.030	0.021	.
2.10	0.045	0.049	0.044	0.046	0.038	0.034	0.029	0.018	0.006
2.50	0.054	0.044	0.043	0.041	0.039	0.032	0.034	0.026	0.017
3.00	0.032	0.035	0.034	0.026	0.036	0.035	0.030	0.029	0.025
3.50	0.037	0.042	0.020	0.033	0.040	0.034	0.035	0.033	0.026
6.00	0.031	0.042	0.039	0.030	0.039	0.032	0.035	0.032	0.037
10.00	0.036	0.022	0.030	0.027	0.035	0.036	0.036	0.035	0.037

In Output 31.4.3, you can see that the simulated $\text{VaR}_{0.05}$ coverage is reasonably accurate for distributions near skewness 0 and kurtosis 3 of the normal, but the coverage can be severely liberal or conservative for negatively or positively skewed errors, respectively. This fact motivated Koenker and Zhao (1996) to use quantile regression to construct a distribution-free estimate of VaR that is less sensitive to skewness and kurtosis in the underlying error distribution for the returns, as discussed in Rodriguez and Yao (2017).

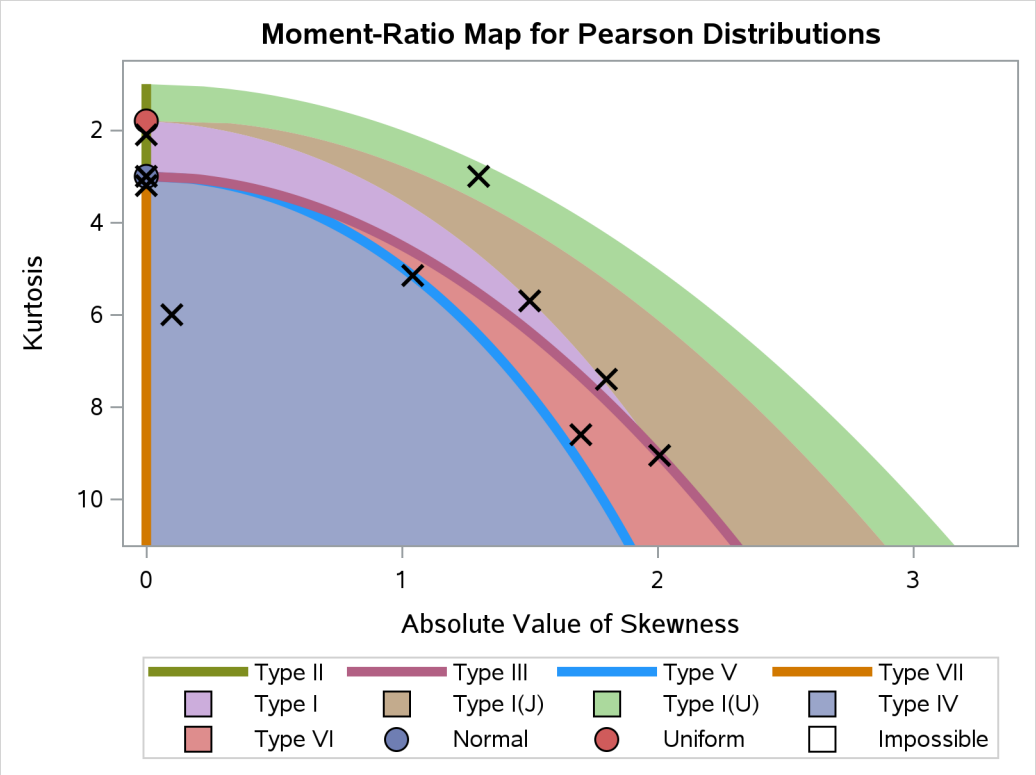
Example 31.5: Interpreting Distribution Parameters

(View the complete code for this example (ssyex5.sas) in the example repository.)

This example demonstrates how to use the table of parameters that PROC SIMSYSTEM displays to reconstruct in DATA step code random variates that have the specified moments for each distribution. Starting with the Pearson system, the moment combinations that are specified for PROC SIMSYSTEM in the following code are chosen to span the families of distributions in the Pearson family, plus the normal distribution, as the parameters table (Output 31.5.2) that PROC SIMSYSTEM produces for these moment combinations confirms:

```
%let Mean   = -4.5 -3.5 -2.5 -1.5 -0.5  0.5        1.5  2.5        3.5  4.5;
%let Stddev =  1    2    3    4    5    6          7    8          9   10;
%let Skew   =  0   -1.5  1.8 -1.3  0   -2.00831604 0.1 -1.04235831 1.7  0;
%let Kurt   =  3    5.7  7.4  3    2.1  9.05       6    5.15       8.6  3.2;
proc simsystem system=pearson;
   moments mean=&Mean StdDev=&Stddev skew=&Skew kurt=&Kurt;
   ods output Parameters=Parm;
run;
```

Output 31.5.1 Selected Pearson Distributions



Output 31.5.2 Parameters for Selected Pearson Distributions

Parameters for Pearson Distributions

	Skewness	Kurtosis	Family	Distribution Notes	Shift	Scale
1	0	3	Normal		-4.500	1
2	-1.500	5.700	Type I	Beta(1.012,9.877)	-1.292	-23.75
3	1.800	7.400	Type I(J)	Beta(0.932,19.68)	-5.535	67.11
4	-1.300	3	Type I(U)	Beta(0.083,0.284)	1.021	-11.20
5	0	2.100	Type II	Beta(1.833,1.833)	-11.30	21.60
6	-2.008	9.050	Type III	Gamma, shape parameter = 0.992	6.475	-6.025
7	0.100	6	Type IV	Alpha = -0.189, m = 3.5075	0.972	14.02
8	-1.042	5.150	Type V	Inverse gamma, shape parameter = 18.67	35.16	-577.0
9	1.700	8.600	Type VI	F(DF1=8.262,DF2=24.34)	-11.41	13.69
10	0	3.200	Type VII	Students t, DF = 34	4.500	9.701

The ODS OUTPUT statement in the preceding PROC SIMSYSTEM code saves the parameters for this sequence of distributions in a data set named Parm. The following PROC PRINT code produces the selected contents of this data set shown in Output 31.5.3. As you can see, the exact numeric values of the parameters that describe each distribution in the Notes column are available in the Shap1Val and Shap2Val variables.

```
proc print data=Parm noobs;
   var Family Notes Shap: Shift Scale;
run;
```

Output 31.5.3 Displayed and Hidden Pearson Distribution Parameters

Family	Notes	Shap1Lab	Shap1Val	Shap2Lab	Shap2Val	Shift	Scale
Normal			.		.	-4.500	1
Type I	Beta(1.012,9.877)	Alpha	1.012230	Beta	9.876659	-1.292	-23.75
Type I(J)	Beta(0.932,19.68)	Alpha	0.931965	Beta	19.676731	-5.535	67.11
Type I(U)	Beta(0.083,0.284)	Alpha	0.082609	Beta	0.284255	1.021	-11.20
Type II	Beta(1.833,1.833)	Alpha	1.833333		1.833333	-11.30	21.60
Type III	Gamma, shape parameter = 0.992	Alpha	0.991736		.	6.475	-6.025
Type IV	Alpha = -0.189, m = 3.5075	Alpha	-0.188789	m	3.507538	0.972	14.02
Type V	Inverse gamma, shape parameter = 18.67	Alpha	18.666036		.	35.16	-577.0
Type VI	F(DF1=8.262,DF2=24.34)	DF1	8.262265	DF2	24.339921	-11.41	13.69
Type VII	Students t, DF = 34	DF	34.000000		.	4.500	9.701

As discussed in "Parameter Values for Pearson Distributions" on page 1605, you can use the values of the Shap1Val and Shap2Val variables in DATA step code with the formulas given in Table 31.3 to compute random samples that have the specified moments, as shown in the following DATA step code:

```
%let N = 1000000;
data mylib.Parm; set Parm; run;
proc ds2 sessref=mysess;
data Sim / overwrite=yes;
   dcl int i;
   dcl double y;
   method run();
      set Parm;
      streaminit(15531);
      do i=1 to &N;
         select (trim(left(Family)));
            when ('Type I'   )
            when ('Type I(J)')
            when ('Type I(U)') y =   rand('beta'  ,Shap1Val,Shap2Val);
            when ('Type II'  ) y =   rand('beta'  ,Shap1Val,Shap1Val);
            when ('Type III' ) y =   rand('gamma' ,Shap1Val);
            when ('Type IV'  ) y = .;
            when ('Type V'   ) y = 1/rand('gamma' ,Shap1Val);
            when ('Type VI'  ) y =   rand('f'     ,Shap1Val,Shap2Val);
            when ('Type VII' ) y =   rand('t'     ,Shap1Val);
            when ('Normal'   ) y =   rand('normal');
         end;
         y = Scale*y + Shift;
         output;
      end;
   end;
enddata;
run;
quit;
```

The following code computes and displays (in Output 31.5.4) the first four moments for each distribution sampled by the preceding PROC DS2 code. These moments match the specified ones reasonably well.

```
proc mdsummary data=mylib.Sim;
   var y;
   groupby Family Notes Mean StdDev Skewness Kurtosis;
   output out=mylib.Sim_summary;
run;

proc cas;
   session mysess;
   table.alterTable /
      name="Sim_summary"
      columns={
         {name="_Mean_"       rename="sMean"},
         {name="_Std_"        rename="sStdDev"},
         {name="_Skewness_"   rename="sSkewness"},
         {name="_Kurtosis_"   rename="sKurtosis"}
      };
   table.update /
      table="Sim_summary"
      set={{var="sKurtosis" value="sKurtosis + 3"}};
quit;

proc sort data=mylib.Sim_summary out=Sim_summary; by Family; run;
proc print data=Sim_summary noobs;
   var Family Mean sMean StdDev sStdDev Skewness sSkewness Kurtosis sKurtosis;
   format _NUMERIC_ 6.3;
run;
```

Output 31.5.4 Moments for PROC DS2 Samples

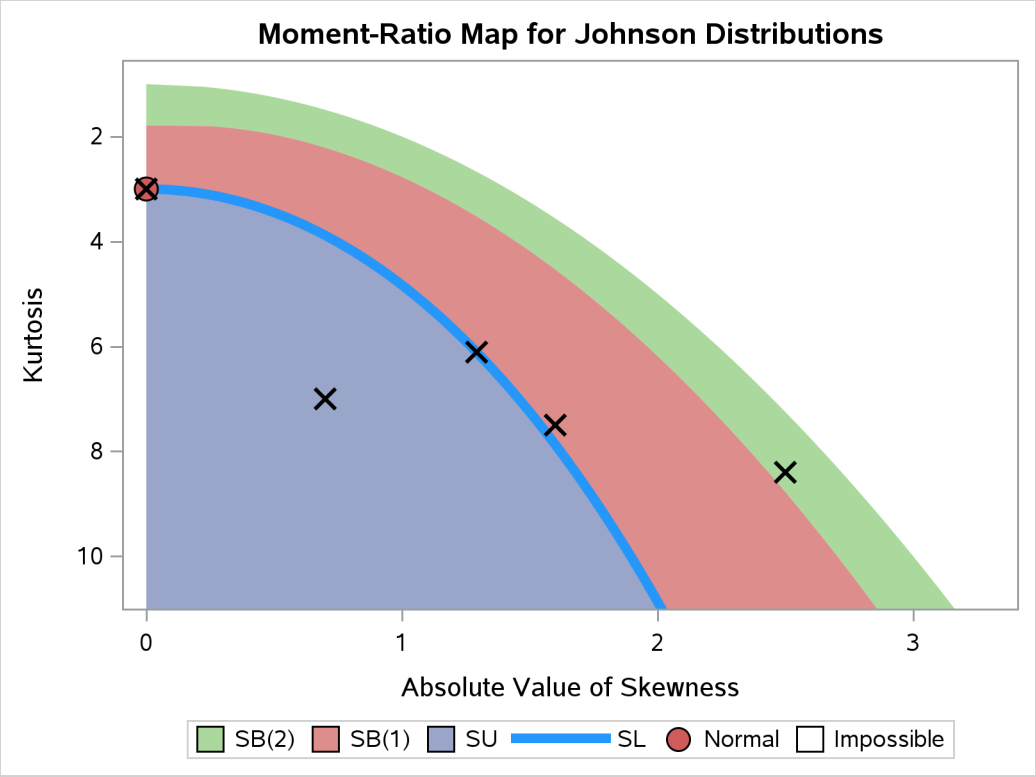
Family	Mean	sMean	StdDev	sStdDev	Skewness	sSkewness	Kurtosis	sKurtosis
Normal	-4.500	-4.501	1.000	1.000	0.000	0.002	3.000	2.997
Type I	-3.500	-3.498	2.000	2.000	-1.500	-1.503	5.700	5.709
Type I(J)	-2.500	-2.505	3.000	2.995	1.800	1.799	7.400	7.389
Type I(U)	-1.500	-1.504	4.000	4.003	-1.300	-1.297	3.000	2.990
Type II	-0.500	-0.503	5.000	5.000	0.000	-0.001	2.100	2.098
Type III	0.500	0.502	6.000	5.998	-2.008	-2.004	9.050	8.997
Type IV	1.500	.	7.000	.	0.100	.	6.000	.
Type V	2.500	2.499	8.000	7.995	-1.042	-1.038	5.150	5.118
Type VI	3.500	3.489	9.000	8.989	1.700	1.690	8.600	8.535
Type VII	4.500	4.486	10.000	10.002	0.000	0.001	3.200	3.192

Finally, for the Johnson system, the following code performs steps precisely like those shown earlier for the Pearson system, except that the moment combinations are chosen to represent the families of the Johnson system, as shown in Output 31.5.5 and Output 31.5.6:

```
%let Mean   =  -2   -1    0            1  2  ;
%let Stddev =   1    2    3            4  5  ;
%let Skew   =  -1.6  2.5 -1.2922847983 0 -0.7;
%let Kurt   =   7.5  8.4  6.1093660061 3  7  ;
proc simsystem system=johnson;
   moments mean=&Mean StdDev=&Stddev skew=&Skew kurt=&Kurt;
   ods output Parameters=Parm;
```

```
run;
```

Output 31.5.5 Selected Johnson Distributions



Output 31.5.6 Parameters for Selected Johnson Distributions

Parameters for Johnson Distributions							
	Skewness	Kurtosis	Family	Delta	Gamma	Shift	Scale
1	-1.600	7.500	SB(1)	1.852	6.017	-0.151	-43.59
2	2.500	8.400	SB(2)	0.282	1.375	-1.975	8.885
3	-1.292	6.109	SL	2.548		7.351	-6.806
4	0	3	SN			1	4
5	-0.700	7	SU	1.571	-0.390	3.876	-6.099

Turning to the Johnson system, as discussed in "Parameter Values for Johnson Distributions" on page 1609, you can use the values of the Gamma and Delta columns of the "Parameters for Johnson Distributions" table in DATA step code with the formulas given in Table 31.5 to compute random samples that have the specified moments, as shown in the following code:

```
%let N = 1000000;
data mylib.Parm; set Parm; run;
proc ds2 sessref=mysess;
data Sim / overwrite=yes;
   dcl int i;
   dcl double y z;
   method run();
      set Parm;
```

```
      streaminit(15531);
      do i = 1 to &N;
         z = rand('normal');
         select (trim(left(Family)));
            when ('SB(1)')
            when ('SB(2)') y = 1/(1 + exp(-(z - Gamma)/Delta));
            when ('SL'   ) y =        exp(  z         /Delta) ;
            when ('SU'   ) y =       sinh( (z - Gamma)/Delta) ;
            when ('SN'   ) y =              z                 ;
         end;
         y = Scale*y + Shift;
         output;
      end;
   end;
enddata;
run;
quit;

proc mdsummary data=mylib.Sim;
   var y;
   groupby Family Mean StdDev Skewness Kurtosis;
   output out=mylib.Sim_summary;
run;

proc cas;
   session mysess;
   table.alterTable /
      name="Sim_summary"
      columns={
         {name="_Mean_"       rename="sMean"},
         {name="_Std_"        rename="sStdDev"},
         {name="_Skewness_"   rename="sSkewness"},
         {name="_Kurtosis_"   rename="sKurtosis"}
      };
   table.update /
      table="Sim_summary"
      set={{var="sKurtosis" value="sKurtosis + 3"}};
quit;

proc sort data=mylib.Sim_summary out=Sim_summary; by Family; run;
proc print data=Sim_summary noobs;
   var Family Mean sMean StdDev sStdDev Skewness sSkewness Kurtosis sKurtosis;
   format _NUMERIC_ 6.3;
run;
```

As was the case with the Pearson system shown earlier, the moments for the samples that the PROC DS2 code computes, shown in Output 31.5.7, match the specified moments reasonably well.

Output 31.5.7 Moments for PROC DS2 Samples

Family	Mean	sMean	StdDev	sStdDev	Skewness	sSkewness	Kurtosis	sKurtosis
SB(1)	-2.000	-1.999	1.000	0.999	-1.600	-1.603	7.500	7.530
SB(2)	-1.000	-1.000	2.000	2.000	2.500	2.500	8.400	8.402
SL	0.000	0.000	3.000	3.000	-1.292	-1.294	6.109	6.116
SN	1.000	0.994	4.000	3.996	0.000	-0.000	3.000	2.994
SU	2.000	2.000	5.000	5.000	-0.700	-0.691	7.000	7.002

References

Balanda, K. P., and MacGillivray, H. L. (1988). "Kurtosis: A Critical Review." *American Statistician* 42:111–119.

Bowman, K. O., and Shenton, L. R. (2006). "Johnson's System of Distributions." In *Encyclopedia of Statistical Sciences*, 2nd ed., vol. 6, edited by S. Kotz, C. B. Read, N. Balakrishnan, and B. Vidakovic. New York: John Wiley & Sons.

Darlington, R. B. (1970). "Is Kurtosis Really 'Peakedness'?" *American Statistician* 24:19–22. https://doi.org/10.1080/00031305.1970.10478885.

Elderton, W. P., and Johnson, N. L. (1969). *Systems of Frequency Curves*. Cambridge: Cambridge University Press.

Johnson, N. L. (1949). "Systems of Frequency Curves Generated by Methods of Translation." *Biometrika* 36:149–176.

Johnson, N. L., Kotz, S., and Balakrishnan, N. (1994). *Continuous Univariate Distributions.* 2nd ed. Vol. 1. New York: John Wiley & Sons.

Johnson, N. L., Kotz, S., and Balakrishnan, N. (1995). *Continuous Univariate Distributions.* 2nd ed. Vol. 2. New York: John Wiley & Sons.

Koenker, R., and Zhao, Q. (1996). "Conditional Quantile Estimation and Inference for ARCH Models." *Econometric Theory* 12:793–813.

Moors, J. J. A. (1986). "The Meaning of Kurtosis: Darlington Reexamined." *American Statistician* 40:283–284. https://doi.org/10.1080/00031305.1986.10475415.

Ord, J. K. (2006). "Pearson System of Distributions." In *Encyclopedia of Statistical Sciences*, 2nd ed., vol. 9, edited by S. Kotz, N. Balakrishnan, C. B. Read, B. Vidakovic, and N. L. Johnson. New York: John Wiley & Sons.

Pearson, K. (1895). "Contributions to the Mathematical Theory of Evolution, II: Skew Variation in Homogeneous Material." *Philosophical Transactions of the Royal Society, Series A* 186:343–414.

Pearson, K. (1901). "Mathematical Contributions to the Theory of Evolution, X: Supplement to a Memoir on Skew Variation." *Philosophical Transactions of the Royal Society, Series A* 197:443–459.

Pearson, K. (1916). "Mathematical Contributions to the Theory of Evolution, XIX: Second Supplement to a Memoir on Skew Variation." *Philosophical Transactions of the Royal Society, Series A* 216:429–457.

Rodriguez, R. N. (2006). "Systems of Frequency Curves." In *Encyclopedia of Statistical Sciences*, 2nd ed., vol. 4, edited by S. Kotz, N. Balakrishnan, C. B. Read, B. Vidakovic, and N. L. Johnson. New York: John Wiley & Sons.

Rodriguez, R. N., and Yao, Y. (2017). "Five Things You Should Know about Quantile Regression." In *Proceedings of the SAS Global Forum 2017 Conference*. Cary, NC: SAS Institute Inc. http://support.sas.com/resources/papers/proceedings17/SAS0525-2017.pdf.

Ross, S. M. (2006). *Simulation*. 4th ed. Orlando, FL: Academic Press.

Ruppert, D. (1987). "What Is Kurtosis? An Influence Function Approach." *American Statistician* 41:1–5. https://doi.org/10.1080/00031305.1987.10475431.

SAS Institute Inc. (2019). "SAS/ETS Web Examples: Estimating GARCH Models." Accessed June 25, 2019. https://support.sas.com/rnd/app/ets/examples/garchex/index.html.

Westfall, P. H. (2014). "Kurtosis as Peakedness, 1905–2014. R.I.P." *American Statistician* 68:191–195. https://doi.org/10.1080/00031305.2014.917055.

Wheeler, D. J. (2010). "Are You Sure We Don't Need Normally Distributed Data? More about the Misuses of Probability Theory." *Quality Digest Daily* 1–9.

Chapter 32
SPC Procedure

Contents

Overview: SPC Procedure **1642**
Uses of Control Charts 1643
Characteristics of Shewhart Charts 1643
Classification of Control Charts 1645
PROC SPC Compared with SAS/QC Procedures 1647
Using CAS Sessions and CAS Engine Librefs 1647
Getting Started: SPC Procedure **1648**
Syntax: SPC Procedure **1651**
PROC SPC Statement 1651
BY Statement 1652
BOXCHART Statement 1653
CCHART Statement 1654
EWMACHART Statement 1654
IRCHART Statement 1655
MACHART Statement 1656
MCHART Statement 1656
MRCHART Statement 1657
NPCHART Statement 1658
PCHART Statement 1659
RCHART Statement 1659
SCHART Statement 1660
UCHART Statement 1661
XCHART Statement 1662
XRCHART Statement 1662
XSCHART Statement 1663
Dictionary of Options 1664
Details: SPC Procedure **1672**
Methods of Estimating the Standard Deviation 1672
Constructing Charts for Individual Measurements and Moving Ranges 1675
Constructing Charts for Means 1677
Constructing Charts for Medians 1678
Constructing Charts for Ranges 1680
Constructing Charts for Standard Deviations 1681
Constructing Charts for Numbers of Nonconformities (c Charts) 1682
Constructing Charts for Number Nonconforming (np Charts) 1684
Constructing Charts for Proportion Nonconforming (p Charts) 1686

Constructing Charts for Nonconformities per Unit (u Charts) 1687
Constructing Uniformly Weighted Moving Average Charts 1689
Constructing Exponentially Weighted Moving Average Charts 1691
Percentile Definitions 1693
Tests for Special Causes 1694
Capability Indices 1700
Input Data Tables 1702
Output Data Tables 1704
ODS Table Names 1715
Examples: SPC Procedure **1716**
Example 32.1: Saving Results in Output Data Tables 1716
Example 32.2: Computing Process Capability Indices 1718
Example 32.3: Reading Preestablished Control Limits 1719
Example 32.4: Applying Tests for Special Causes 1720
Example 32.5: Displaying Control Charts by Using SAS/QC Procedures 1721
References **1723**

Overview: SPC Procedure

The SPC procedure performs control chart analysis.

A control chart is a graphical and analytical tool for determining whether a process is in a state of statistical control. You can use the SPC procedure to compute many different types of control charts, including all commonly used Shewhart charts and moving average charts. In addition, you can use PROC SPC to do the following:

- analyze multiple process variables and produce a table that identifies processes that are out of statistical control
- adjust control limits to compensate for unequal subgroup sizes
- estimate control limits from the data, compute control limits from specified values for population parameters (known standards), or read limits from an input data table
- perform tests for special causes based on runs patterns (Western Electric rules)
- estimate the process standard deviation by using various methods
- specify process specification limits and compute capability indices for processes that are in statistical control
- save chart statistics and control limits in output data tables that you can use as input for the SHEWHART or MACONTROL procedure in SAS/QC software to display the control charts graphically

Uses of Control Charts

The control chart was invented by Walter A. Shewhart (1891–1967), a physicist at Bell Telephone Laboratories, who introduced the method in 1924 and elaborated on it in his book *Economic Control of Quality of Manufactured Product* (1931). Many varieties of control chart have been developed in the years since, but the original types of control chart are called Shewhart charts. The concepts that underlie the control chart are that the natural variability in any process can be quantified by using a set of control limits and that variation exceeding these limits signals a change in the process. Although control charts were developed to improve manufacturing processes, they are used in many other fields, including finance and health care.

In industry, the Shewhart chart is the most commonly applied statistical quality control method of studying the variation in output from a manufacturing process. Shewhart charts are typically used to distinguish variation due to *special causes* from variation due to *common causes*. Special causes, also referred to as *assignable causes*, are local, sporadic problems such as the failure of a particular machine or a mistakenly recorded measurement. Common causes are problems inherent in the system as a whole, such as inadequate product design, inherited defective material, and excessive humidity.

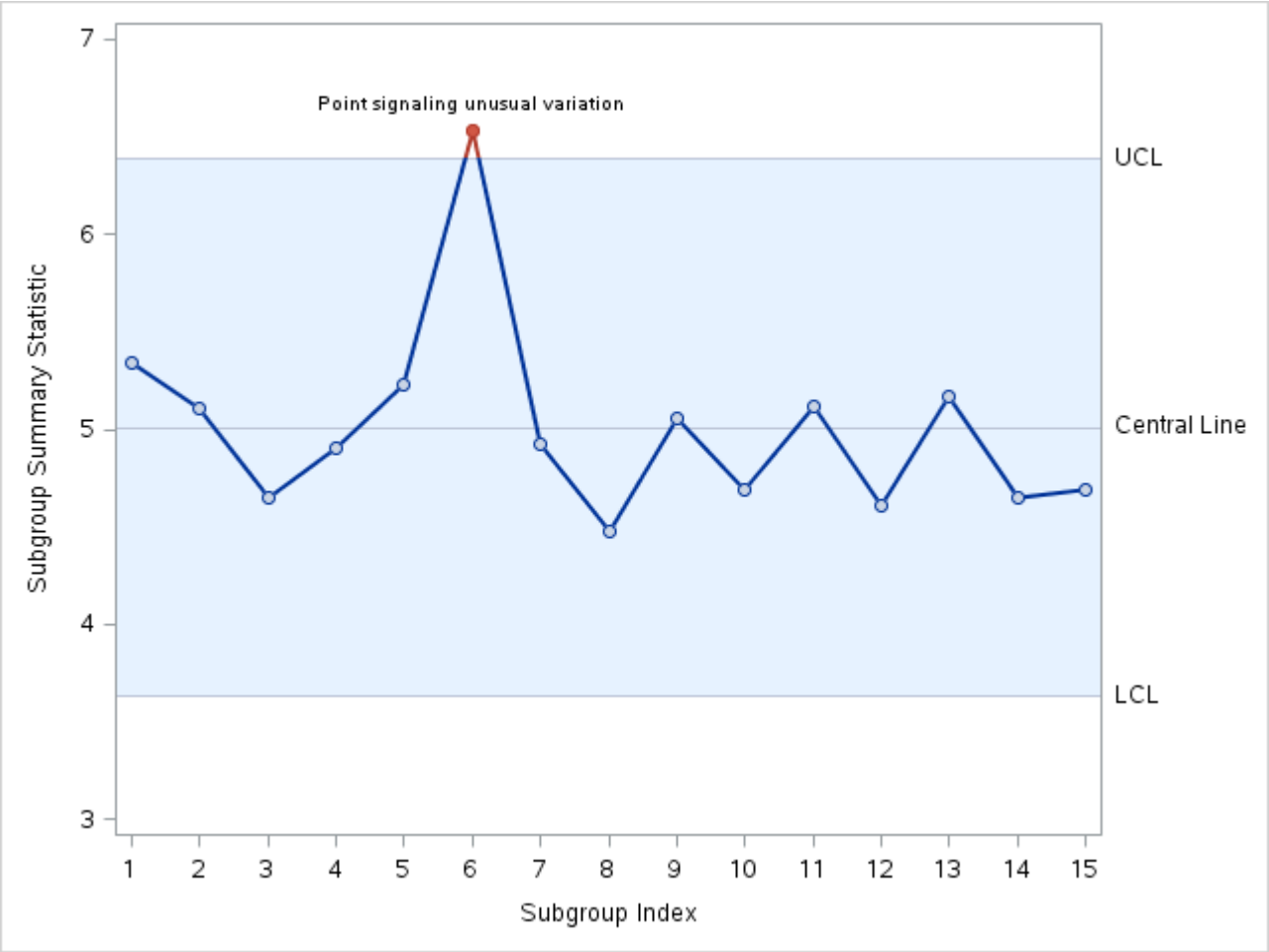
After the special causes have been identified and eliminated, the process is said to be in *statistical control*. When statistical control has been established, Shewhart charts can be used to monitor the process for the occurrence of future special causes and to measure and reduce the effects of common causes.

Deming (1982) emphasized that the improvement of a process can begin only after statistical control has been established. Deming also noted that control chart techniques are applicable to quality improvement in service industries as well as manufacturing industries.

Characteristics of Shewhart Charts

Figure 32.1 illustrates a typical Shewhart chart.

Figure 32.1 A Shewhart Control Chart



All Shewhart charts have the following characteristics:

- Each point represents a *summary statistic* that is computed from a sample of measurements of a quality characteristic. For example, the summary statistic might be the average value of a critical dimension of five items selected at random, or it might be the proportion of nonconforming items in a sample of 100 items.
- The samples from which the summary statistics are computed are referred to as *rational subgroups* or *subgroup samples*. The organization of the data into subgroups is critical to the interpretation of a Shewhart chart. Shewhart (1931) advocated selecting rational subgroups so that variation within subgroups is minimized and variation among subgroups is maximized; this makes the chart more sensitive to shifts in the process level. Various approaches to subgrouping are discussed by Grant and Leavenworth (1988), Montgomery (1996), and Kume (1985).
- The *horizontal axis* of a Shewhart chart identifies the subgroup samples. Frequently, the samples are indexed according to the order in which they were taken or the time at which they were taken. Subgroup samples can also be assigned labels that indicate some other type of classification (such as lot number).

- The *central line* on a Shewhart chart indicates the average (expected value) of the summary statistic when the process is in statistical control.

- The *upper and lower control limits*, labeled UCL and LCL, respectively, indicate the range of variation to be expected in the summary statistic when the process is in statistical control. The control limits are commonly computed as 3σ limits,[1] which represent three standard errors[2] of variation in the summary statistic above and below the central line. However, the limits can also be determined using a multiple of the standard error other than three.

  The control limits are also determined by the subgroup sample size, because the standard error of the summary statistic is a function of sample size. If the sample size is constant across subgroups, the control limits are typically horizontal lines, as in Figure 32.1. However, if the sample size varies from subgroup to subgroup, the limits are usually adjusted to compensate for the effect of sample size, resulting in steplike boundaries.

  Control limits can be estimated from the data that you are analyzing, or they can be standard, previously determined values. Estimated limits are often used when you are establishing statistical control, and standard limits are often used when you are maintaining statistical control.

- *A point outside the control limits* signals the presence of a special cause of variation. In addition, *tests for special causes* (also referred to as *Western Electric rules* and *runs tests*) can signal an out-of-control condition if a statistically unusual pattern of points is observed in the control chart. For example, one pattern that is commonly used to diagnose the existence of a trend is seven consecutive steadily increasing points.

  When the process is in statistical control, a point can fall outside the control limits purely by chance, resulting in a false out-of-control signal. However, when the Shewhart chart correctly signals the presence of a special cause, you must take additional action to determine the nature of the problem and eliminate it.

Classification of Control Charts

PROC SPC calculates control charts that can be broadly classified as follows:

- Shewhart charts for *variables* are used when you are measuring the quality characteristic of a process on a continuous scale.

- Shewhart charts for *attributes* are used when you are measuring the quality characteristic of a process by counting the number of nonconformities (defects) in an item or the number of nonconforming (defective) items in a sample.

- Moving average charts are used to detect small shifts in the average of a quality characteristic measured on a continuous scale.

[1] In this context, σ always stands for the standard error of the subgroup summary statistic that is plotted on the chart. Elsewhere in this chapter, σ is also used to denote the standard deviation of a process, also referred to as the population standard deviation. This dual usage is standard practice.

[2] The term *standard deviation* is also used by some authors to refer to this quantity; see, for example, Montgomery (1996). This chapter uses the term *standard error* for the dispersion of the distribution of a statistic and the term *standard deviation* for the dispersion of a distribution of individual measurements.

Shewhart charts for variables are further classified according to the subgroup summary statistic that is plotted on the chart:

- $\bar{X}$ and R charts display subgroup means (averages) and ranges. You usually present the two charts on the same page, aligning the $\bar{X}$ chart above the R chart to facilitate the simultaneous analysis of the central tendency and variability of the process.
- $\bar{X}$ and s charts display subgroup means (averages) and standard deviations. You usually present the two charts on the same page, aligning the $\bar{X}$ chart above the s chart.
- Median and range charts display subgroup medians and ranges. You usually present the two charts on the same page, aligning the median chart above the R chart.
- Charts for individual measurements and moving ranges display individual measurements and moving ranges of two of more successive measurements. In this case the subgroup sample consists of a single observation.
- Box charts display a box plot (box-and-whisker plot) for each subgroup and control limits for the subgroup means or medians. This type of chart facilitates detailed analysis of the subgroup distributions and is applicable to large subgroup sample sizes (10 or more).

Likewise, Shewhart charts for attributes are classified according to the subgroup summary statistic that is plotted on the chart:

- A p chart displays the proportion of nonconforming (defective) items in a subgroup sample.
- An np chart displays the number of nonconforming (defective) items in a subgroup sample.
- A u chart displays the number of nonconformities (defects) per unit in a subgroup sample that consists of an arbitrary number of units.
- A c chart displays the number of nonconformities (defects) in a unit (here, a subgroup sample typically consists of one unit).

In contrast to a Shewhart chart, in which each point is based on information from a single subgroup sample, each point on a moving average chart combines information from the current sample and past samples. Moving average charts are classified by how the moving averages are computed:

- A uniformly weighted moving average chart (commonly called a moving average chart) computes the average of the w most recent subgroup means, including the present subgroup mean. The constant w is referred to as the *span* of the moving average.
- An exponentially weighted moving average (EWMA) chart computes a weighted average of all previous subgroup means, including the mean of the present subgroup sample. The weights decrease exponentially going backward in time.

PROC SPC Compared with SAS/QC Procedures

The SPC procedure performs univariate control chart analysis that is comparable to that of the SHEWHART and MACONTROL procedures in SAS/QC. Whereas the SAS/QC procedures produce graphical control charts, PROC SPC produces tables that summarize signals of process instability. PROC SPC also produces output data tables that you can read into SAS data sets and use as input to PROC SHEWHART and PROC MACONTROL. See Example 32.5 for an illustration.

PROC SPC is intended to perform control chart analysis on many process variables in parallel. Its input data format and syntax are more conducive to processing large numbers of processes than those of the SAS/QC procedures. You can use PROC SPC to produce concise summaries of process health for a large number of processes. After you identify potential problems, you can use PROC SHEWHART or PROC MACONTROL to produce control chart graphs of selected processes.

Using CAS Sessions and CAS Engine Librefs

SAS Cloud Analytic Services (CAS) is the analytic server and associated cloud services in SAS Viya. This section describes how to create a CAS session and set up a CAS engine libref that you can use to connect to the CAS session. It assumes that you have a CAS server already available; contact your system administrator if you need help starting and terminating a server. This CAS server is identified by specifying the host on which it runs and the port on which it listens for communications. To simplify your interactions with this CAS server, the host information and port information for the server are stored as SAS option values that are retrieved automatically whenever this CAS server needs to be accessed. You can examine the host and port values for the server at your site by using the following statements:

```
proc options option=(CASHOST CASPORT);
run;
```

In addition to starting a CAS server, your system administrator might also have created a CAS session and a CAS engine libref for your use. You can define your own sessions and CAS engine librefs that connect to the CAS server as shown in the following statements:

```
cas mysess;
libname mylib cas sessref=mysess;
```

The CAS statement creates the CAS session named mysess, and the LIBNAME statement creates the mylib CAS engine libref that you use to connect to this session. It is not necessary to explicitly name the CASHOST and CASPORT of the CAS server in the CAS statement, because these values are retrieved from the corresponding SAS option values.

If you have created the mysess session, you can terminate it by using the TERMINATE option in the CAS statement as follows:

```
cas mysess terminate;
```

For more information about the CAS and LIBNAME statements, see the section “Introduction to Shared Concepts” on page 6 in Chapter 2, “Shared Concepts.”

Getting Started: SPC Procedure

(View the complete code for this example at https://github.com/sascommunities/doc-supplement-casstat/tree/main/Examples/spcgs.sas.)

A brewery monitors its bottling process to ensure that each bottle is filled with the proper amount of beer. The following statements create a SAS data set named Beer that contains the amount of beer, recorded in fluid ounces, for 23 sample batches:

```
data Beer;
   input Batch size @;
   do i=1 to size;
      input Amount @@;
      output;
   end;
   drop i size;
   label Batch  ='Batch Number';
   datalines;
 1  5  12.01 11.97 11.93 11.98 12.00
 2  5  11.88 11.98 11.93 12.03 11.92
 3  5  11.93 11.99 12.00 12.03 11.95
 4  5  11.98 11.94 12.02 11.90 11.97
 5  5  12.02 12.02 11.98 12.04 11.90
 6  4  11.98 11.98 12.00 11.93
 7  5  11.93 11.95 12.02 11.91 12.03
 8  5  12.00 11.98 12.02 11.89 12.01
 9  5  11.98 11.93 11.99 12.02 11.91
10  5  11.97 12.02 12.05 12.01 11.97
11  5  12.02 12.01 11.97 12.02 11.94
12  5  11.93 11.83 11.99 12.02 12.01
13  5  12.01 11.98 11.94 12.04 12.01
14  5  11.98 11.96 12.02 12.00 12.00
15  5  11.97 11.99 12.03 11.95 11.96
16  5  11.99 11.95 11.96 12.03 12.01
17  4  11.99 11.97 12.03 12.01
18  5  11.94 11.96 11.98 12.03 11.97
19  5  11.97 11.87 11.90 12.01 11.95
20  5  11.96 11.94 11.96 11.98 12.05
21  3  12.06 12.07 11.98
22  5  12.01 11.98 11.96 11.97 12.00
23  5  12.00 12.02 12.03 11.99 11.96
;
```

The format of the Beer data set makes it suitable for analysis by PROC SHEWHART. The process variable is Amount, and the subgroup variable is Batch. Figure 32.2 shows a partial listing of the data set.

Figure 32.2 Data Set Beer (PROC SHEWHART Format)

Batch	Amount
1	12.01
1	11.97
1	11.93
1	11.98
1	12.00
2	11.88
2	11.98
2	11.93
2	12.03
2	11.92
3	11.93
3	11.99
3	12.00
3	12.03
3	11.95

The following DATA step modifies Beer so that it is suitable for analysis by PROC SPC:

```
data Beer;
   length processname $16 subgroupname $8;
   processname='Amount';
   subgroupname='Batch';
   set Beer(rename=(Amount=process Batch=subgroup));
run;
```

The modified data set is shown in Figure 32.3.

Figure 32.3 Data Set Beer (PROC SPC Format)

processname	subgroupname	subgroup	process
Amount	Batch	1	12.01
Amount	Batch	1	11.97
Amount	Batch	1	11.93
Amount	Batch	1	11.98
Amount	Batch	1	12.00
Amount	Batch	2	11.88
Amount	Batch	2	11.98
Amount	Batch	2	11.93
Amount	Batch	2	12.03
Amount	Batch	2	11.92
Amount	Batch	3	11.93
Amount	Batch	3	11.99
Amount	Batch	3	12.00
Amount	Batch	3	12.03
Amount	Batch	3	11.95

The PROC SPC data format is more conducive than the PROC SHEWHART format to analyzing many processes at once. Because process names are recorded in the process name variable, you do not have to list the processes to be analyzed in the procedure syntax.

The following statements create a data table called mylib.AllProcesses that contains the beer bottle data along with data from seven other SAS data sets (not shown here) that appear in the SHEWHART procedure chapter of *SAS/QC User's Guide*. These statements assume that your CAS engine libref is named mylib, but you can substitute any appropriately defined libref.

```
data mylib.AllProcesses;
   set Beer Detergent Disks Partgaps Times Turbine Wafers Wire;
   format subgroup;
run;
```

NOTE: Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections "Using CAS Sessions and CAS Engine Librefs" on page 7 and "Loading a SAS Data Set onto a CAS Server" on page 7 in Chapter 2, "Shared Concepts."

The following statements use PROC SPC to perform a means and ranges chart analysis of the eight processes.

```
proc spc data=mylib.AllProcesses;
   xrchart;
run;
```

The XRCHART statement performs an $\bar{X}$ and R chart analysis on the process data, calculating control charts for subgroup means and subgroup ranges. By default, control limits are calculated from the data. Figure 32.4 shows the resulting output.

Figure 32.4 PROC SPC Output

The SPC Procedure

Means and Ranges Chart Summary for ALLPROCESSES

			Means		Ranges	
processname	subgroupname	Number of Subgroups	Number > UCL	Number < LCL	Number > UCL	Number < LCL
Amount	Batch	23	0	0	0	0
Breakstrength	Sample	25	0	0	0	0
Delay	Day	11	3	1	2	2
Diameter	Batch	45	0	1	0	0
KWatts	Day	40	0	0	0	0
Partgap	Sample	41	0	0	0	0
Time	Lot	25	0	0	0	0
Weight	Lot	28	0	0	0	0

The default output from PROC SPC is an exception summary table. For each process, the table shows the number of subgroups that were analyzed and the number of subgroup summary statistics that fell outside the control limits of each chart.

The table in Figure 32.4 shows signals of unusual variation in process Delay (from the Times data set) and process Diameter (from the Wafers data set). When unusual variation is signaled in a process, you should investigate to find and correct any special causes of variation.

Syntax: SPC Procedure

PROC SPC < *options* > ;
BY *variables* ;
BOXCHART < / *options* > ;
CCHART < / *options* > ;
EWMACHART < / *options* > ;
IRCHART < / *options* > ;
MACHART < / *options* > ;
MCHART < / *options* > ;
MRCHART < / *options* > ;
NPCHART < / *options* > ;
PCHART < / *options* > ;
RCHART < / *options* > ;
SCHART < / *options* > ;
UCHART < / *options* > ;
XCHART < / *options* > ;
XRCHART < / *options* > ;
XSCHART < / *options* > ;

The PROC SPC statement invokes the procedure.

You use the chart statements to perform control chart analysis. You can specify a BY statement to obtain separate analyses for each BY group.

PROC SPC Statement

PROC SPC < *options* > ;

The PROC SPC statement is required in order to invoke the SPC procedure. By itself, it does not perform any statistical process control analysis.

You can specify the following options in the PROC SPC statement:

DATA=*libref.data-table*
names the input data table for PROC SPC to use. The default is the most recently created data table. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to the data, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 1647.

data-table specifies the name of the input data table.

For more information, see the section "DATA= Data Table" on page 1702.

LIMITS=*libref.data-table*
names an input data table that contains preestablished control limits or the parameters from which control limits can be computed. Each observation in a LIMITS= data table provides control limit information for a process. *libref.data-table* is a two-level name, where *libref* refers to the library, and *data-table* specifies the name of the input data table. For more information about this two-level name, see the DATA= option and the section "Using CAS Sessions and CAS Engine Librefs" on page 1647.

You can create this data table as an OUTLIMITS= data table in a previous run of PROC SPC. For more information, see the section "LIMITS= Data Table" on page 1702.

NTHREADS=*n*
specifies the number of threads per node to be used to perform the analysis. You can specify a number between 1 and 64, inclusive. By default, *n* is the maximum number of threads available on each node.

PROCESSNAME=*variable*
specifies the variable in the input data table that contains the names of processes to be analyzed. Each unique combination of process name and subgroup name in the input data table identifies the data for one control chart. By default, PROC SPC looks for a variable named ProcessName that contains the process names.

PROCESSVALUE=*variable*
specifies the variable in the input data table that contains the process measurements to be analyzed. By default, PROC SPC looks for a variable named Process that contains the process measurements.

SUBGROUPNAME=*variable*
specifies the variable in the input data table that contains the names of subgroup variables. Each unique combination of process name and subgroup name in the input data table identifies the data for one control chart. By default, PROC SPC looks for a variable named SubgroupName that contains the subgroup variable names.

SUBGROUPVALUE=*variable*
specifies the variable in the input data table that contains the subgroup values. Subgroup values identify rational subgroups for control chart analysis. By default, PROC SPC looks for a variable named Subgroup that contains the subgroup values.

BY Statement

BY *variables* **;**

You can specify a BY statement in PROC SPC to obtain separate analyses of observations in groups that are defined by the values of the BY variables. If you specify more than one BY statement, only the last one specified is used. For more information, see the discussion of BY-group processing in the "Grouping Data" section of *SAS Programmers Guide: Essentials*.

BOXCHART Statement

BOXCHART < */ options* > **;**

The BOXCHART statement produces control charts for subgroup means or medians, along with statistics that describe the distribution of data within each subgroup. Table 32.1 lists the options that you can specify in a BOXCHART statement. For detailed descriptions of these options, see the section "Dictionary of Options" on page 1664.

Table 32.1 BOXCHART Statement Options

Option	Description
ALLN	Includes all subgroups in the analysis, regardless of sample size
CIINDICES	Computes capability index confidence limits
CONTROLSTAT=	Specifies whether control limits are computed for subgroup means or subgroup medians
EXCHART	Produces results only for charts in which exceptions occur
LIMITN=	Specifies the nominal sample size for fixed control limits
MEDCENTRAL=	Specifies the method of estimating the process mean μ when control limits are computed for subgroup medians
NO3SIGMACHECK	Enables tests for special causes to be applied when control limits other than 3σ limits are in effect
NOPRINT	Suppresses ODS output
OUTLIMITS=	Creates an output data table that contains control limits
OUTTABLE=	Creates an output data table that contains subgroup summary statistics and control limits
PCTLDEF=	Specifies the percentile definition for box-and-whisker computation
SIGMAS=	Specifies the width of the control limits in terms of a multiple of the standard error of the subgroup summary statistic
SMETHOD=	Specifies the method of estimating the process standard deviation σ
SPECS=	Specifies an input data table that contains process specification limits and computes process capability indices
TEST2RUN=	Specifies the length of the pattern for Test 2
TEST3RUN=	Specifies the length of the pattern for Test 3
TESTNSTD	Applies tests for special causes to standardized chart statistics
TESTOVERLAP	Performs tests for special causes on overlapping patterns of points
TESTS=	Requests tests for special causes for the box chart

CCHART Statement

CCHART < */ options* > **;**

The CCHART statement produces *c* charts for the numbers of nonconformities (defects) in subgroup samples. Table 32.2 lists the options that you can specify in a CCHART statement. For detailed descriptions of these options, see the section "Dictionary of Options" on page 1664.

Table 32.2 CCHART Statement Options

Option	Description
ALLN	Includes all subgroups in the analysis, regardless of sample size
EXCHART	Produces results only for charts in which exceptions occur
LIMITN=	Specifies the nominal sample size for fixed control limits
NO3SIGMACHECK	Enables tests for special causes to be applied when control limits other than 3σ limits are in effect
NOPRINT	Suppresses ODS output
OUTLIMITS=	Creates an output data table that contains control limits
OUTTABLE=	Creates an output data table that contains subgroup summary statistics and control limits
SIGMAS=	Specifies the width of the control limits in terms of a multiple of the standard error of the subgroup summary statistic
SUBGROUPN=	Specifies a variable in the input data table that contains subgroup sample sizes
TEST2RUN=	Specifies the length of the pattern for Test 2
TEST3RUN=	Specifies the length of the pattern for Test 3
TESTNSTD	Applies tests for special causes to standardized chart statistics
TESTOVERLAP	Performs tests for special causes on overlapping patterns of points
TESTS=	Requests tests for special causes for the *c* chart

EWMACHART Statement

EWMACHART < */ options* > **;**

The EWMACHART statement produces exponentially weighted moving average (EWMA) charts of subgroup means. Table 32.3 lists the options that you can specify in an EWMACHART statement. For detailed descriptions of these options, see the section "Dictionary of Options" on page 1664.

Table 32.3 EWMACHART Statement Options

Option	Description
ALLN	Includes all subgroups in the analysis, regardless of sample size
ASYMPTOTIC	Requests constant control limits
EXCHART	Produces results only for charts in which exceptions occur

Table 32.3 *continued*

Option	Description
LIMITN=	Specifies the nominal sample size for fixed control limits
NOPRINT	Suppresses ODS output
OUTLIMITS=	Creates an output data table that contains control limit parameters
OUTTABLE=	Creates an output data table that contains subgroup summary statistics and control limits
RESET	Resets the value of the EWMA after each point outside the control limits
SIGMAS=	Specifies the width of the control limits in terms of a multiple of the standard error of the EWMAs
SMETHOD=	Specifies the method of estimating the process standard deviation σ
WEIGHT=	Specifies the weight to assign to the most recent subgroup mean in the computation of the EWMA

IRCHART Statement

IRCHART < */ options* > **;**

The IRCHART statement produces control charts for individual measurements and moving ranges. Table 32.4 lists the options that you can specify in an IRCHART statement. For detailed descriptions of these options, see the section "Dictionary of Options" on page 1664.

Table 32.4 IRCHART Statement Options

Option	Description
CIINDICES	Computes capability index confidence limits
EXCHART	Produces results only for charts in which exceptions occur
NO3SIGMACHECK	Enables tests for special causes to be applied when control limits other than 3σ limits are in effect
NOPRINT	Suppresses ODS output
OUTLIMITS=	Creates an output data table that contains control limits
OUTTABLE=	Creates an output data table that contains subgroup summary statistics and control limits
SIGMAS=	Specifies the width of the control limits in terms of a multiple of the standard error of the subgroup summary statistic
SMETHOD=	Specifies the method of estimating the process standard deviation σ
SPECS=	Specifies an input data table that contains process specification limits and computes process capability indices
TEST2RUN=	Specifies the length of the pattern for Test 2
TEST3RUN=	Specifies the length of the pattern for Test 3
TESTNSTD	Applies tests for special causes to standardized chart statistics
TESTOVERLAP	Performs tests for special causes on overlapping patterns of points
TESTS=	Requests tests for special causes for the individual measurements chart

MACHART Statement

MACHART < */ options* > **;**

The MACHART statement produces uniformly weighted moving average charts of subgroup means. Table 32.5 lists the options that you can specify in an MACHART statement. For detailed descriptions of these options, see the section "Dictionary of Options" on page 1664.

Table 32.5 MACHART Statement Options

Option	Description
ALLN	Includes all subgroups in the analysis, regardless of sample size
ASYMPTOTIC	Requests constant control limits
EXCHART	Produces results only for charts in which exceptions occur
LIMITN=	Specifies the nominal sample size for fixed control limits
NOPRINT	Suppresses ODS output
OUTLIMITS=	Creates an output data table that contains control limit parameters
OUTTABLE=	Creates an output data table that contains subgroup summary statistics and control limits
SIGMAS=	Specifies the width of the control limits in terms of a multiple of the standard error of the moving averages
SMETHOD=	Specifies the method of estimating the process standard deviation σ
SPAN=	Specifies the number of subgroup means in the moving average

MCHART Statement

MCHART < */ options* > **;**

The MCHART statement produces control charts for subgroup medians. Table 32.6 lists the options that you can specify in an MCHART statement. For detailed descriptions of these options, see the section "Dictionary of Options" on page 1664.

Table 32.6 MCHART Statement Options

Option	Description
ALLN	Includes all subgroups in the analysis, regardless of sample size
CIINDICES	Computes capability index confidence limits
EXCHART	Produces results only for charts in which exceptions occur
LIMITN=	Specifies the nominal sample size for fixed control limits
MEDCENTRAL=	Specifies the method of estimating the process mean μ
NO3SIGMACHECK	Enables tests for special causes to be applied when control limits other than 3σ limits are in effect
NOPRINT	Suppresses ODS output
OUTLIMITS=	Creates an output data table that contains control limits

Table 32.6 *continued*

Option	Description
OUTTABLE=	Creates an output data table that contains subgroup summary statistics and control limits
SIGMAS=	Specifies the width of the control limits in terms of a multiple of the standard error of the subgroup summary statistic
SMETHOD=	Specifies the method of estimating the process standard deviation σ
SPECS=	Specifies an input data table that contains process specification limits and computes process capability indices
TEST2RUN=	Specifies the length of the pattern for Test 2
TEST3RUN=	Specifies the length of the pattern for Test 3
TESTNSTD	Applies tests for special causes to standardized chart statistics
TESTOVERLAP	Performs tests for special causes on overlapping patterns of points
TESTS=	Requests tests for special causes for the median chart

MRCHART Statement

MRCHART < */ options* > **;**

The MRCHART statement produces control charts for subgroup medians and ranges. Table 32.7 lists the options that you can specify in an MRCHART statement. For detailed descriptions of these options, see the section "Dictionary of Options" on page 1664.

Table 32.7 MRCHART Statement Options

Option	Description
ALLN	Includes all subgroups in the analysis, regardless of sample size
CIINDICES	Computes capability index confidence limits
EXCHART	Produces results only for charts in which exceptions occur
LIMITN=	Specifies the nominal sample size for fixed control limits
MEDCENTRAL=	Specifies the method of estimating the process mean μ
NO3SIGMACHECK	Enables tests for special causes to be applied when control limits other than 3σ limits are in effect
NOPRINT	Suppresses ODS output
OUTLIMITS=	Creates an output data table that contains control limits
OUTTABLE=	Creates an output data table that contains subgroup summary statistics and control limits
SIGMAS=	Specifies the width of the control limits in terms of a multiple of the standard error of the subgroup summary statistic
SMETHOD=	Specifies the method of estimating the process standard deviation σ
SPECS=	Specifies an input data table that contains process specification limits and computes process capability indices
TEST2RUN=	Specifies the length of the pattern for Test 2
TEST3RUN=	Specifies the length of the pattern for Test 3

Table 32.7 *continued*

Option	Description
TESTNSTD	Applies tests for special causes to standardized chart statistics
TESTOVERLAP	Performs tests for special causes on overlapping patterns of points
TESTS=	Requests tests for special causes for the median chart
TESTS2=	Requests tests for special causes for the *R* chart

NPCHART Statement

NPCHART < */ options* > **;**

The NPCHART statement produces control charts for the numbers of nonconforming items in subgroup samples. Table 32.8 lists the options that you can specify in an NPCHART statement. For detailed descriptions of these options, see the section "Dictionary of Options" on page 1664.

Table 32.8 NPCHART Statement Options

Option	Description
ALLN	Includes all subgroups in the analysis, regardless of sample size
EXCHART	Produces results only for charts in which exceptions occur
LIMITN=	Specifies the nominal sample size for fixed control limits
NO3SIGMACHECK	Enables tests for special causes to be applied when control limits other than 3σ limits are in effect
NOPRINT	Suppresses ODS output
OUTLIMITS=	Creates an output data table that contains control limits
OUTTABLE=	Creates an output data table that contains subgroup summary statistics and control limits
SIGMAS=	Specifies the width of the control limits in terms of a multiple of the standard error of the subgroup summary statistic
SUBGROUPN=	Specifies a variable in the input data table that contains subgroup sample sizes
TEST2RUN=	Specifies the length of the pattern for Test 2
TEST3RUN=	Specifies the length of the pattern for Test 3
TESTNSTD	Applies tests for special causes to standardized chart statistics
TESTOVERLAP	Performs tests for special causes on overlapping patterns of points
TESTS=	Requests tests for special causes for the *np* chart

PCHART Statement

PCHART < */ options* > **;**

The PCHART statement produces control charts for the proportions of nonconforming items in subgroup samples. Table 32.9 lists the options that you can specify in a PCHART statement. For detailed descriptions of these options, see the section "Dictionary of Options" on page 1664.

Table 32.9 PCHART Statement Options

Option	Description
ALLN	Includes all subgroups in the analysis, regardless of sample size
EXCHART	Produces results only for charts in which exceptions occur
LIMITN=	Specifies the nominal sample size for fixed control limits
NO3SIGMACHECK	Enables tests for special causes to be applied when control limits other than 3σ limits are in effect
NOPRINT	Suppresses ODS output
OUTLIMITS=	Creates an output data table that contains control limits
OUTTABLE=	Creates an output data table that contains subgroup summary statistics and control limits
SIGMAS=	Specifies the width of the control limits in terms of a multiple of the standard error of the subgroup summary statistic
SUBGROUPN=	Specifies a variable in the input data table that contains subgroup sample sizes
TEST2RUN=	Specifies the length of the pattern for Test 2
TEST3RUN=	Specifies the length of the pattern for Test 3
TESTNSTD	Applies tests for special causes to standardized chart statistics
TESTOVERLAP	Performs tests for special causes on overlapping patterns of points
TESTS=	Requests tests for special causes for the *p* chart

RCHART Statement

RCHART < */ options* > **;**

The RCHART statement produces control charts for subgroup ranges. Table 32.10 lists the options that you can specify in an RCHART statement. For detailed descriptions of these options, see the section "Dictionary of Options" on page 1664.

Table 32.10 RCHART Statement Options

Option	Description
ALLN	Includes all subgroups in the analysis, regardless of sample size
CIINDICES	Computes capability index confidence limits
EXCHART	Produces results only for charts in which exceptions occur

Table 32.10 *continued*

Option	Description
LIMITN=	Specifies the nominal sample size for fixed control limits
NO3SIGMACHECK	Enables tests for special causes to be applied when control limits other than 3σ limits are in effect
NOPRINT	Suppresses ODS output
OUTLIMITS=	Creates an output data table that contains control limits
OUTTABLE=	Creates an output data table that contains subgroup summary statistics and control limits
SIGMAS=	Specifies the width of the control limits in terms of a multiple of the standard error of the subgroup summary statistic
SMETHOD=	Specifies the method of estimating the process standard deviation σ
SPECS=	Specifies an input data table that contains process specification limits and computes process capability indices
TEST2RUN=	Specifies the length of the pattern for Test 2
TEST3RUN=	Specifies the length of the pattern for Test 3
TESTNSTD	Applies tests for special causes to standardized chart statistics
TESTOVERLAP	Performs tests for special causes on overlapping patterns of points
TESTS2=	Requests tests for special causes for the *R* chart

SCHART Statement

SCHART < */ options* > **;**

The SCHART statement produces control charts for subgroup standard deviations. Table 32.11 lists the options that you can specify in an SCHART statement. For detailed descriptions of these options, see the section "Dictionary of Options" on page 1664.

Table 32.11 SCHART Statement Options

Option	Description
ALLN	Includes all subgroups in the analysis, regardless of sample size
CIINDICES	Computes capability index confidence limits
EXCHART	Produces results only for charts in which exceptions occur
LIMITN=	Specifies the nominal sample size for fixed control limits
NO3SIGMACHECK	Enables tests for special causes to be applied when control limits other than 3σ limits are in effect
NOPRINT	Suppresses ODS output
OUTLIMITS=	Creates an output data table that contains control limits
OUTTABLE=	Creates an output data table that contains subgroup summary statistics and control limits
SIGMAS=	Specifies the width of the control limits in terms of a multiple of the standard error of the subgroup summary statistic
SMETHOD=	Specifies the method of estimating the process standard deviation σ

Table 32.11 *continued*

Option	Description
SPECS=	Specifies an input data table that contains process specification limits and computes process capability indices
TEST2RUN=	Specifies the length of the pattern for Test 2
TEST3RUN=	Specifies the length of the pattern for Test 3
TESTNSTD	Applies tests for special causes to standardized chart statistics
TESTOVERLAP	Performs tests for special causes on overlapping patterns of points
TESTS2=	Requests tests for special causes for the *s* chart

UCHART Statement

UCHART < */ options* > **;**

The UCHART statement produces control charts for the numbers of nonconformities per inspection unit in subgroup samples. Table 32.12 lists the options that you can specify in a UCHART statement. For detailed descriptions of these options, see the section "Dictionary of Options" on page 1664.

Table 32.12 UCHART Statement Options

Option	Description
ALLN	Includes all subgroups in the analysis, regardless of sample size
EXCHART	Produces results only for charts in which exceptions occur
LIMITN=	Specifies the nominal sample size for fixed control limits
NO3SIGMACHECK	Enables tests for special causes to be applied when control limits other than 3σ limits are in effect
NOPRINT	Suppresses ODS output
OUTLIMITS=	Creates an output data table that contains control limits
OUTTABLE=	Creates an output data table that contains subgroup summary statistics and control limits
SIGMAS=	Specifies the width of the control limits in terms of a multiple of the standard error of the subgroup summary statistic
SUBGROUPN=	Specifies a variable in the input data table that contains subgroup sample sizes
TEST2RUN=	Specifies the length of the pattern for Test 2
TEST3RUN=	Specifies the length of the pattern for Test 3
TESTNSTD	Applies tests for special causes to standardized chart statistics
TESTOVERLAP	Performs tests for special causes on overlapping patterns of points
TESTS=	Requests tests for special causes for the *u* chart

XCHART Statement

XCHART < */ options* > **;**

The XCHART statement produces control charts for subgroup means. Table 32.13 lists the options that you can specify in an XCHART statement. For detailed descriptions of these options, see the section "Dictionary of Options" on page 1664.

Table 32.13 XCHART Statement Options

Option	Description
ALLN	Includes all subgroups in the analysis, regardless of sample size
CIINDICES	Computes capability index confidence limits
EXCHART	Produces results only for charts in which exceptions occur
LIMITN=	Specifies the nominal sample size for fixed control limits
NO3SIGMACHECK	Enables tests for special causes to be applied when control limits other than 3σ limits are in effect
NOPRINT	Suppresses ODS output
OUTLIMITS=	Creates an output data table that contains control limits
OUTTABLE=	Creates an output data table that contains subgroup summary statistics and control limits
SIGMAS=	Specifies the width of the control limits in terms of a multiple of the standard error of the subgroup summary statistic
SMETHOD=	Specifies the method of estimating the process standard deviation σ
SPECS=	Specifies an input data table that contains process specification limits and computes process capability indices
TEST2RUN=	Specifies the length of the pattern for Test 2
TEST3RUN=	Specifies the length of the pattern for Test 3
TESTNSTD	Applies tests for special causes to standardized chart statistics
TESTOVERLAP	Performs tests for special causes on overlapping patterns of points
TESTS=	Requests tests for special causes for the $\bar{X}$ chart

XRCHART Statement

XRCHART < */ options* > **;**

The XRCHART statement produces control charts for subgroup means and ranges. Table 32.14 lists the options that you can specify in an XRCHART statement. For detailed descriptions of these options, see the section "Dictionary of Options" on page 1664.

Table 32.14 XRCHART Statement Options

Option	Description
ALLN	Includes all subgroups in the analysis, regardless of sample size
CIINDICES	Computes capability index confidence limits

Table 32.14 *continued*

Option	Description
EXCHART	Produces results only for charts in which exceptions occur
LIMITN=	Specifies the nominal sample size for fixed control limits
NO3SIGMACHECK	Enables tests for special causes to be applied when control limits other than 3σ limits are in effect
NOPRINT	Suppresses ODS output
OUTLIMITS=	Creates an output data table that contains control limits
OUTTABLE=	Creates an output data table that contains subgroup summary statistics and control limits
SIGMAS=	Specifies the width of the control limits in terms of a multiple of the standard error of the subgroup summary statistic
SMETHOD=	Specifies the method of estimating the process standard deviation σ
SPECS=	Specifies an input data table that contains process specification limits and computes process capability indices
TEST2RUN=	Specifies the length of the pattern for Test 2
TEST3RUN=	Specifies the length of the pattern for Test 3
TESTNSTD	Applies tests for special causes to standardized chart statistics
TESTOVERLAP	Performs tests for special causes on overlapping patterns of points
TESTS=	Requests tests for special causes for the $\bar{X}$ chart
TESTS2=	Requests tests for special causes for the R chart

XSCHART Statement

XSCHART < */ options* > **;**

The XSCHART statement produces control charts for subgroup means and standard deviations. Table 32.15 lists the options that you can specify in an XSCHART statement. For detailed descriptions of these options, see the section "Dictionary of Options" on page 1664.

Table 32.15 XSCHART Statement Options

Option	Description
ALLN	Includes all subgroups in the analysis, regardless of sample size
CIINDICES	Computes capability index confidence limits
EXCHART	Produces results only for charts in which exceptions occur
LIMITN=	Specifies the nominal sample size for fixed control limits
NO3SIGMACHECK	Enables tests for special causes to be applied when control limits other than 3σ limits are in effect
NOPRINT	Suppresses ODS output
OUTLIMITS=	Creates an output data table that contains control limits
OUTTABLE=	Creates an output data table that contains subgroup summary statistics and control limits

Table 32.15 *continued*

Option	Description
SIGMAS=	Specifies the width of the control limits in terms of a multiple of the standard error of the subgroup summary statistic
SMETHOD=	Specifies the method of estimating the process standard deviation σ
SPECS=	Specifies an input data table that contains process specification limits and computes process capability indices
TEST2RUN=	Specifies the length of the pattern for Test 2
TEST3RUN=	Specifies the length of the pattern for Test 3
TESTNSTD	Applies tests for special causes to standardized chart statistics
TESTOVERLAP	Performs tests for special causes on overlapping patterns of points
TESTS=	Requests tests for special causes for the $\bar{X}$ chart
TESTS2=	Requests tests for special causes for the s chart

Dictionary of Options

The following entries provide detailed descriptions of options that are available in the BOXCHART, CCHART, EWMACHART, IRCHART, MACHART, MCHART, MRCHART, NPCHART, PCHART, RCHART, SCHART, UCHART, XCHART, XRCHART, and XSCHART statements. An option is available in all the chart statements unless otherwise noted.

ALLN

includes all subgroups in the analysis, regardless of whether the subgroup sample size equals the nominal control limit sample size n that you specify in the LIMITN= option or the variable _LIMITN_ in the LIMITS= data table. Use the ALLN option in conjunction with the LIMITN= option or the variable _LIMITN_.

ASYMPTOTIC

requests constant lower and upper control limits for all subgroups in a moving average chart.

In a uniformly weighted moving average chart, asymptotic limits have the following values:

$$\text{LCL} = \overline{\overline{X}} - \frac{k\hat{\sigma}}{\sqrt{nw}}$$

$$\text{UCL} = \overline{\overline{X}} + \frac{k\hat{\sigma}}{\sqrt{nw}}$$

Here w is the span of the moving average, and n is the nominal sample size associated with the control limits. When you do not specify this option, the control limits are computed using the exact formulas in Table 32.35.

In an exponentially weighted moving average chart, asymptotic limits have the following values:

$$\text{LCL} = \overline{\overline{X}} - k\hat{\sigma}\sqrt{r/n(2-r)}$$

$$\text{UCL} = \overline{\overline{X}} + k\hat{\sigma}\sqrt{r/n(2-r)}$$

Here r is the weight parameter ($0 < r \leq 1$), and n is the nominal sample size associated with the control limits. When you do not specify the ASYMPTOTIC option, the control limits are computed using the exact formulas in Table 32.37.

Use the ASYMPTOTIC option only if all the subgroup sample sizes are the same or if you specify a nominal subgroup sample size by using the LIMITN= option or by including the _LIMITN_ column in a LIMITS= data table.

This option is available only in the EWMACHART and MACHART statements.

CIINDICES <(< TYPE=LOWER | UPPER | TWOSIDED > < ALPHA=*value* >) >
requests capability index confidence limits that are based on subgroup summary data and are calculated using "effective degrees of freedom," as described by Bissell (1990). These confidence limits are approximate. When you specify the CIINDICES option, the calculated confidence limits are displayed in the "CapabilityIndices" table and are included in the OUTLIMITS= data table, if one is produced.

TYPE=*keyword*
specifies the type of confidence limit. Valid *keywords* are LOWER, UPPER, and TWOSIDED. By default, TYPE=TWOSIDED.

ALPHA=*value*
specifies the default confidence level to compute confidence limits. The percentage for the confidence limits is $(1 - \textit{value}) * 100$. For example, ALPHA=.05 results in a 95% confidence limit. The default value is .05, and the possible range of values is from 0 to 1.

The CIINDICES option is ignored unless you also specify the SPECS= option. These options are available only in the chart statements that produce Shewhart charts for variables: BOXCHART, IRCHART, MCHART, MRCHART, RCHART, SCHART, XCHART, XRCHART, and XSCHART. For more information about capability indices, see the section "Capability Indices" on page 1700.

CONTROLSTAT=MEAN | MEDIAN
specifies whether the control limits that are displayed in a box chart are computed for subgroup means or for subgroup medians. By default, CONTROLSTAT=MEAN. This option is available only in the BOXCHART statement.

EXCHART
includes a control chart in the results only when exceptions occur—specifically, when the control limits are exceeded or when any tests that you request by using the TESTS= or TESTS2= option are positive.

LIMITN=*n*
specifies a nominal sample size for the control limits, where *n* is a whole number.

If you specify LIMITN=*n*, the control limits are computed for the fixed value *n*, and they do not vary with the subgroup sample sizes. Moreover, only those subgroups whose sample size is equal to *n* are included in the analysis. You can specify the ALLN option in conjunction with LIMITN=*n* to force all subgroups to be included, regardless of subgroup sample size.

Note that control limits for EWMA and moving average charts vary according to the number of subgroup means that are included in the moving average, not only because of varying subgroup sample sizes.

If you do not specify LIMITN=*n* and the subgroup sample sizes are constant, the default value of *n* is the constant subgroup sample size.

Depending on which chart statement you use, there are restrictions on the value of n that you can specify in the LIMITN= option:

- For the MRCHART, RCHART, and XRCHART statements, $2 \leq n \leq 25$.
- For the EWMACHART, MACHART, SCHART, and XSCHART statements, $n \geq 2$.
- For the BOXCHART, MCHART, and XCHART statements, $n \geq 1$. If the estimate of the process standard deviation is based on subgroup ranges (see the SMETHOD= option and the section "Methods of Estimating the Standard Deviation" on page 1672), $n \leq 25$.
- For the CCHART, NPCHART, PCHART, and UCHART statements, $n \geq 1$.

This option is not available in the IRCHART statement.

Notice the difference between the LIMITN= option and the SUBGROUPN= option that is available in the CCHART, NPCHART, PCHART, and UCHART statements. The LIMITN= option specifies a nominal sample size for the *control limits*, whereas the SUBGROUPN= option provides the sample sizes for the *data*.

NOTE: As an alternative to specifying the LIMITN= option, you can read the nominal control limit sample size from the variable _LIMITN_ in a LIMITS= data table.

MEDCENTRAL=AVGMEAN | AVGMED | MEDMED

identifies a method of estimating the process mean μ, which is represented by the central line on a median chart. The method that corresponds to each keyword is shown in Table 32.16.

Table 32.16 Methods of Estimating the Process Mean

Keyword	Method
AVGMEAN	Average of subgroup means
AVGMED	Average of subgroup medians
MEDMED	Median of subgroup medians

By default, MEDCENTRAL=AVGMED. This option is available only in the MCHART and MRCHART statements, and in the BOXCHART statement when you specify the CONTROLSTAT=MEDIAN option.

NO3SIGMACHECK

suppresses the check for 3σ limits when you request tests for special causes. This enables tests for special causes to be applied when you use SIGMAS= option to specify control limits other than the default 3σ limits. You should not use the NO3SIGMACHECK option for standard control chart applications, because the standard tests for special causes assume 3σ limits.

This option is not available in the EWMACHART and MACHART statements.

NOPRINT

suppresses the display of ODS output.

OUTLIMITS=*libref.data-table*

creates an output data table that saves the control limits and/or control limit parameters. You can use an OUTLIMITS= data table as an input LIMITS= data table in a subsequent run of the procedure. *libref.data-table* is a two-level name, where *libref* refers to the library, and *data-table* specifies the name of the output data table. For more information about this two-level name, see the DATA= option and the section "Using CAS Sessions and CAS Engine Librefs" on page 1647.

For more information about OUTLIMITS= data tables, see the section "OUTLIMITS= Data Table" on page 1704.

OUTTABLE=*libref.data-table*

creates an output data table that saves the information plotted on the chart, including the subgroup variable values and their corresponding summary statistics and control limits. *libref.data-table* is a two-level name, where *libref* refers to the library, and *data-table* specifies the name of the output data table. For more information about this two-level name, see the DATA= option and the section "Using CAS Sessions and CAS Engine Librefs" on page 1647.

For more information about OUTTABLE= data tables, see the section "OUTTABLE= Data Table" on page 1710.

PCTLDEF=*index*

specifies one of five definitions that are used to calculate percentiles in the construction of box-and-whisker plots that you request in the BOXCHART statement. The *index* can be 1, 2, 3, 4, or 5. The five corresponding percentile definitions are discussed in the section "Percentile Definitions" on page 1693. By default, PCTLDEF=5. This option is available only in the BOXCHART statement.

RESET

resets the value of the EWMA after each out-of-control point. Specifically, when a point exceeds the control limits, the EWMA for the next subgroup is computed as the weighted average of the subgroup mean and the overall mean. By default, the EWMA values are not reset. This option is available only in the EWMACHART statement.

SIGMAS=*k*

specifies the width of the control limits in terms of the multiple k of the standard error of the subgroup summary statistic. The value of k must be positive. By default, $k = 3$, and the control limits are 3σ limits.

SMETHOD=MAD | MMR<(*n*)> | MVGRANGE<(*n*)> | RMSDF | RMVLUE | RNOWEIGHT | SMVLUE | SNOWEIGHT

specifies the method of estimating the process standard deviation, σ, as summarized in Table 32.17.

Table 32.17 Methods of Estimating the Process Standard Deviation

Keyword	Method
MAD	Calculates a median absolute deviation estimate of σ
MMR	Calculates a median moving range estimate of σ
MVGRANGE	Estimates σ based on a moving range of n subgroup means
RMSDF	Calculates a root mean square estimate of σ
RMVLUE	Calculates a minimum variance linear unbiased estimate of σ based on subgroup ranges
RNOWEIGHT	Estimates σ as an unweighted average of unbiased subgroup estimates of σ based on subgroup ranges
SMVLUE	Calculates a minimum variance linear unbiased estimate of σ based on subgroup standard deviations
SNOWEIGHT	Estimates σ as an unweighted average of unbiased subgroup estimates of σ based on subgroup standard deviations

Table 32.18 lists the SMETHOD= option values that are valid for each chart statement. The default value for each statement is shown in bold.

Table 32.18 Valid SMETHOD= Values by Chart Statement

Chart Statement	Methods
BOXCHART	RMSDF, RMVLUE, RNOWEIGHT, SMVLUE, **SNOWEIGHT**
EWMACHART	RMSDF, SMVLUE, **SNOWEIGHT**
IRCHART	MAD, MMR, **MVGRANGE**
MACHART	RMSDF, SMVLUE, **SNOWEIGHT**
MCHART	RMSDF, RMVLUE, **RNOWEIGHT**, SMVLUE, SNOWEIGHT
MRCHART	RMVLUE, **RNOWEIGHT**
RCHART	RMVLUE, **RNOWEIGHT**
SCHART	RMSDF, SMVLUE, **SNOWEIGHT**
XCHART	RMSDF, RMVLUE, **RNOWEIGHT**, SMVLUE, SNOWEIGHT
XRCHART	MVGRANGE, RMVLUE, **RNOWEIGHT**
XSCHART	MVGRANGE, RMSDF, SMVLUE, **SNOWEIGHT**

When you use the MMR or MVGRANGE method, you can specify the number n of consecutive subgroup summary statistics from which moving ranges are computed by appending the number in parentheses to the keyword. For example, you can specify SMETHOD=MVGRANGE(4) to use moving ranges of four subgroup statistics. Values in the range $2 \leq n \leq 25$ are valid. By default, $n = 2$.

This option is not available in the following statements that produce Shewhart charts for attributes: CCHART, NPCHART, PCHART, and UCHART.

For more information about the different methods of estimating the process standard deviation, see the section "Methods of Estimating the Standard Deviation" on page 1672.

SPAN=*w*

specifies the number of terms to use in calculating the moving average for a uniformly weighted moving average chart, where *w* is an integer greater than 1. This option is required in an MACHART statement unless you specify span values in a column named _SPAN_ in a LIMITS= data table.

This option is available only in the MACHART statement.

SPECS=*libref.data-table*

specifies an input data table that contains process specification limits. *libref.data-table* is a two-level name, where *libref* refers to the library, and *data-table* specifies the name of the input data table. For more information about this two-level name, see the DATA= option and the section "Using CAS Sessions and CAS Engine Librefs" on page 1647. PROC SPC calculates capability indices for processes whose specifications are given in *data-table*.

This option is available only in the statements that produce Shewhart charts for variables: BOXCHART, IRCHART, MCHART, MRCHART, RCHART, SCHART, XCHART, XRCHART, and XSCHART. For more information, see the sections "SPECS= Data Table" on page 1703 and "Capability Indices" on page 1700.

SUBGROUPN=*variable*

specifies a variable in the input data table whose values specify the subgroup sample sizes. This option is available only in the CCHART, NPCHART, PCHART, and UCHART statements. You must specify this option in the NPCHART, PCHART, and UCHART statements. For the CCHART statement, the default subgroup sample size is 1.

TEST2RUN=*run-length*

specifies the length of the pattern for Test 2 that you request in the TESTS= and TESTS2= options. The *run-length* values that you can specify are 7, 8, 9, 11, 14, and 20. The form of the test for each *run-length* value is shown in the following table. By default, TEST2RUN=9. For more information, see the section "Tests for Special Causes" on page 1694.

run-length	**Number of Points on One Side of the Central Line**
7	7 in a row
8	8 in a row
9	9 in a row
11	At least 10 out of 11 in a row
14	At least 12 out of 14 in a row
20	At least 16 out of 20 in a row

This option is not available in the EWMACHART and MACHART statements.

TEST3RUN=*run-length*

specifies the length of the pattern for Test 3 that you request in the TESTS= and TESTS2= options. Test 3 searches for a pattern of steadily increasing or decreasing values, where the length of the pattern is at least the value of *run-length*. The values that you can specify for *run-length* are 6, 7, and 8. By default, TEST3RUN=6. For more information, see the section "Tests for Special Causes" on page 1694.

This option is not available in the EWMACHART and MACHART statements.

TESTNSTD

applies the tests for special causes that you request in the TESTS= and TESTS2= options to standardized test statistics when the subgroup sample sizes are not constant. This method was suggested by Nelson (1994). For more information, see the section "Tests for Special Causes" on page 1694. By default, the tests are not applied to data that have varying subgroup sample sizes.

This option is not available in the EWMACHART and MACHART statements.

TESTOVERLAP

applies tests for special causes (which you request in the TESTS= or TESTS2= option) to overlapping patterns of points.

This option modifies the way in which the search for a subsequent pattern is done when a pattern is encountered. If you omit this option, the search begins with the first subgroup after the current pattern ends. If you specify this option, the search begins with the second subgroup in the current pattern.

The following statements request Test 3, which detects six subgroup means in a row that steadily increase or decrease:

```
proc spc;
   xrchart / test=3;
   xrchart / test=3 testoverlap;
run;
```

Suppose that the subgroup means are steadily increasing for subgroups 5, 6, 7, 8, 9, 10, and 11. The first XRCHART statement signals that Test 3 is positive at subgroup 10 but not at subgroup 11, because the search for the next pattern begins with subgroup 11. The second XRCHART statement signals that Test 3 is positive at subgroup 10 and subgroup 11, because the search for the next pattern begins with subgroup 6 and thus finds a second pattern that ends with subgroup 11. For more information, see the section "Tests for Special Causes" on page 1694.

CAUTION: Specifying TESTOVERLAP affects the interpretation of the standard tests for special causes, because a particular point can contribute to more than one positive test. Typically, you should not use this option.

This option is not available in the EWMACHART and MACHART statements.

TESTS=*index-list*

requests one or more tests for special causes, which are also known as *runs tests*, *pattern tests*, and *Western Electric rules*. These tests detect particular nonrandom patterns in the points that are plotted on the primary control chart. The occurrence of a pattern, referred to as a *signal*, suggests the presence of a special cause of variation.

Each pattern is defined in terms of Zones A, B, and C, which are constructed by dividing the interval between the control limits into six equally spaced subintervals. Zone A is the union of the subintervals immediately below the upper control limit and immediately above the lower control limit. Zone C is the union of the subintervals immediately above and below the central line. Zone B is the union of the subintervals between Zones A and C. For an illustration of test zones, see Figure 32.9.

Table 32.19 lists the standard tests that you can request by specifying this option. The tests are indexed according to the sequence used by Nelson (1984, 1985).

Table 32.19 Standard Tests for Special Causes

Index	Pattern Description
1	One point beyond Zone A (outside the control limits)
2	Nine points in a row in Zone C or beyond on one side of the central line (see the TEST2RUN= option)
3	Six points in a row that steadily increase or decrease (see the TEST3RUN= option)
4	Fourteen points in a row that alternate up and down
5	Two out of three points in a row in Zone A or beyond
6	Four out of five points in a row in Zone B or beyond
7	Fifteen points in a row in Zone C on either or both sides of the central line
8	Eight points in a row on either or both sides of the central line, with no points in Zone C

You can specify any combination of the eight *indices* by using an explicit list or an implicit list, as in the following example:

```
proc spc;
   xrchart / tests=1 2 3 4;
   xrchart / tests=1 to 4;
run;
```

The TESTS= option is available in all chart statements except the RCHART and SCHART statements. Use only tests 1, 2, 3, and 4 in the CCHART, NPCHART, PCHART, and UCHART statements. By default, the TESTS= option is not applied in any chart statement unless the control limits are 3σ limits. You can use the NO3SIGMACHECK option to request tests for special causes when you use the SIGMAS= option to specify control limits other than 3σ limits.

For more information about the TESTS= option, see the section "Tests for Special Causes" on page 1694. Related options include TEST2RUN=, TEST3RUN=, TESTNSTD, TESTOVERLAP, and TESTS2=.

This option is not available in the EWMACHART and MACHART statements.

TESTS2=*index-list*

requests one or more tests for special causes for *R* charts or *s* charts. The syntax for the TESTS2= option is identical to the syntax for the TESTS= option. The TESTS2= option is available in the MRCHART, RCHART, SCHART, XRCHART, and XSCHART statements. For more information about the TESTS2= option, see the section "Tests for Special Causes" on page 1694. Related options include TEST2RUN=, TEST3RUN=, TESTNSTD, TESTOVERLAP, and TESTS=.

WEIGHT=*r*

specifies the weight r ($0 < r \leq 1$) assigned to the most recent subgroup mean in the computation of the exponentially weighted moving average in an EWMA chart. The WEIGHT= option is required in an EWMACHART statement unless you specify weight values in a column named _WEIGHT_ in a LIMITS= data table.

This option is available only in the EWMACHART statement.

Details: SPC Procedure

Methods of Estimating the Standard Deviation

You can specify the SMETHOD= option to choose among several methods of estimating the process standard deviation for Shewhart charts for variables and moving average charts. The following sections describe the methods in detail.

MAD Method

If you specify SMETHOD=MAD, a median absolute deviation estimator is computed for σ. This method is described by Boyles (1997) and computed as

$$\hat{\sigma} = \text{median}\{|X_i - \tilde{X}|, 1 \leq i \leq N\}/0.6745$$

where $\tilde{X}$ is the sample median.

The MAD method is available only in the IRCHART statement.

MMR Method

If you specify SMETHOD=MMR, a median moving range estimator is computed for σ. This estimator is described by Boyles (1997) and computed as

$$\hat{\sigma} = \tilde{R}/0.954$$

where $\tilde{R}$ is the median of the nonmissing moving ranges.

The MMR method is available only in the IRCHART statement.

MVGRANGE Method

If you specify SMETHOD=MVGRANGE, σ is estimated by using a moving range of subgroup averages (or individual measurements in an IRCHART statement). The estimate of σ is

$$\hat{\sigma} = \bar{R}/d_2(n)$$

where $\bar{R}$ is the average of the moving ranges; n is the number of consecutive subgroup averages that are used to compute each moving range; and the unbiasing factor $d_2(n)$ is defined so that if the subgroup averages are normally distributed, the expected value of R_i is

$$E(R_i) = d_2(n_i)\sigma$$

This method is described by the American Society for Testing and Materials (1976).

The method is appropriate for constructing the three-way control chart that Wheeler (1995) advocated for this situation. A three-way control chart is useful when sampling (*within-group*) variation is not the only source of variation. A three-way control chart contains a chart of subgroup means, a moving range chart of the subgroup means, and a chart of subgroup ranges. When you specify the SMETHOD=MVGRANGE option, the XRCHART statement produces the appropriate charts of subgroup means and subgroup ranges, and the XSCHART statement produces the appropriate charts of subgroup means and subgroup standard deviations.

The MVGRANGE method is available in the IRCHART, XRCHART, and XSCHART statements. It is the default in the IRCHART statement.

RMSDF Method

If you specify SMETHOD=RMSDF, a weighted root mean square estimate is computed for σ,

$$\hat{\sigma} = \frac{\sqrt{(n_1 - 1)s_1^2 + \cdots + (n_N - 1)s_N^2}}{c_4(n)\sqrt{n_1 + \cdots + n_N - N}}$$

where $n = n_1 + \cdots + n_N - (N - 1)$. The weights are the degrees of freedom $n_i - 1$. A subgroup standard deviation s_i is included in the calculation only if $n_i \geq 2$, and N is the number of subgroups for which $n_i \geq 2$.

If the unknown standard deviation σ is constant across subgroups, the root mean square estimate is more efficient than the minimum variance linear unbiased estimate (MVLUE). However, in process control applications, it is generally not assumed that σ is constant, and if σ varies across subgroups, the root mean square estimate tends to be more inflated than the MVLUE.

The RMSDF method is available in the BOXCHART, EWMACHART, MACHART, MCHART, SCHART, XCHART, and XSCHART statements.

RMVLUE Method

If you specify SMETHOD=RMVLUE, a minimum variance linear unbiased estimate (MVLUE) is computed for σ; see Burr (1969, 1976) and Nelson (1989, 1994). The MVLUE is a weighted average of N unbiased estimates of σ of the form $R_i/d_2(n_i)$, and it is computed as

$$\hat{\sigma} = \frac{f_1 R_1/d_2(n_1) + \cdots + f_N R_N/d_2(n_N)}{f_1 + \cdots + f_N}$$

where

$$f_i = \frac{[d_2(n_i)]^2}{[d_3(n_i)]^2}$$

A subgroup range R_i is included in the calculation only if $n_i \geq 2$, and N is the number of subgroups for which $n_i \geq 2$. The unbiasing factor $d_3(n_i)$ is defined so that, if the observations are normally distributed, the expected value of σ_{R_i} is $d_3(n_i)\sigma$. The MVLUE assigns greater weight to estimates of σ from subgroups that have larger sample sizes, and it is intended for situations where the subgroup sample sizes vary. If the subgroup sample sizes are constant, the RMVLUE estimate reduces to the RNOWEIGHT estimate.

The RMVLUE method is available in the BOXCHART, MCHART, MRCHART, RCHART, XCHART, and XRCHART statements.

RNOWEIGHT Method

If you specify SMETHOD=RNOWEIGHT, the estimate of σ is

$$\hat{\sigma} = \frac{R_1/d_2(n_1) + \cdots + R_N/d_2(n_N)}{N}$$

where N is the number of subgroups for which $n_i \geq 2$, and R_i is the sample range of the observations x_{i1}, $\ldots$, x_{in_i} in the ith subgroup:

$$R_i = \max_{1 \leq j \leq n_i}(x_{ij}) - \min_{1 \leq j \leq n_i}(x_{ij})$$

A subgroup range R_i is included in the calculation only if $n_i \geq 2$. The unbiasing factor $d_2(n_i)$ is defined so that, if the observations are normally distributed, the expected value of R_i is $d_2(n_i)\sigma$. Thus, $\hat{\sigma}$ is the unweighted average of N unbiased estimates of σ. This method is described by the American Society for Testing and Materials (1976).

The RNOWEIGHT method is available in the BOXCHART, MCHART, MRCHART, RCHART, XCHART, and XRCHART statements. It is the default in the MCHART, MRCHART, RCHART, XCHART, and XRCHART statements.

SMVLUE Method

If you specify SMETHOD=SMVLUE, a minimum variance linear unbiased estimate (MVLUE) is computed for σ; see Burr (1969, 1976) and Nelson (1989, 1994). This estimate is a weighted average of N unbiased estimates of σ of the form $s_i/c_4(n_i)$, and it is computed as

$$\hat{\sigma} = \frac{h_1 s_1/c_4(n_1) + \cdots + h_N s_N/c_4(n_N)}{h_1 + \cdots + h_N}$$

where

$$h_i = \frac{[c_4(n_i)]^2}{1 - [c_4(n_i)]^2}$$

A subgroup standard deviation s_i is included in the calculation only if $n_i \geq 2$, and N is the number of subgroups for which $n_i \geq 2$. The MVLUE assigns greater weight to estimates of σ from subgroups with larger sample sizes, and it is intended for situations where the subgroup sample sizes vary. If the subgroup sample sizes are constant, the SMVLUE estimate reduces to the SNOWEIGHT estimate.

The SMVLUE method is available in the BOXCHART, EWMACHART, MACHART, MCHART, SCHART, XCHART, and XSCHART statements.

SNOWEIGHT Method

If you specify SMETHOD=SNOWEIGHT, the estimate for σ is

$$\hat{\sigma} = \frac{s_1/c_4(n_1) + \cdots + s_N/c_4(n_N)}{N}$$

where N is the number of subgroups for which $n_i \geq 2$, s_i is the sample standard deviation of the ith subgroup,

$$s_i = \sqrt{\frac{1}{n_i - 1} \sum_{j=1}^{n_i} (x_{ij} - \bar{X}_i)^2}$$

and

$$c_4(n_i) = \frac{\Gamma(n_i/2)\sqrt{2/(n_i - 1)}}{\Gamma((n_i - 1)/2)}$$

Here $\Gamma(\cdot)$ denotes the gamma function, and $\bar{X}_i$ denotes the ith subgroup mean. A subgroup standard deviation s_i is included in the calculation only if $n_i \geq 2$. If the observations are normally distributed, the expected value of s_i is $c_4(n_i)\sigma$. Thus, $\hat{\sigma}$ is the unweighted average of N unbiased estimates of σ. This method is described by the American Society for Testing and Materials (1976).

The SNOWEIGHT method is available in the BOXCHART, EWMACHART, MACHART, MCHART, SCHART, XCHART, and XSCHART statements. It is the default in the BOXCHART, EWMACHART, MACHART, SCHART, and XSCHART statements.

Constructing Charts for Individual Measurements and Moving Ranges

You produce control charts for individual measurements and moving ranges by using the IRCHART statement. The following notation is used in this section:

μ	Process mean (expected value of the population of measurements)
σ	Process standard deviation (standard deviation of the population of measurements)
X_i	The ith individual measurement
$\bar{X}$	Mean of the individual measurements, computed as $(X_1 + \cdots + X_N)/N$, where N is the number of individual measurements
n	Number of consecutive measurements that are used to calculate the moving ranges (by default, $n = 2$)
R_i	Moving range that is computed for the ith subgroup (corresponding to the ith individual measurement). If $i < n$, then R_i is assigned a missing value. Otherwise, $R_i = \max(X_i, X_{i-1}, \ldots, X_{i-n+1}) - \min(X_i, X_{i-1}, \ldots, X_{i-n+1})$ This formula assumes that $X_i, X_{i-1}, \ldots, X_{i-n+1}$ are nonmissing.
$\bar{R}$	Average of the nonmissing moving ranges, computed as $\frac{R_n + R_{n+1} + \cdots + R_N}{N + 1 - n}$
$d_2(n)$	Expected value of the range of n independent normally distributed variables with unit standard deviation
$d_3(n)$	Standard error of the range of n independent observations from a normal population with unit standard deviation

Subgroup Summary Statistics

Each point on an individual measurements chart indicates the value of a measurement (X_i).

Each point on a moving range chart indicates the value of a moving range (R_i). For $n = 2$, for example, if the first three measurements are 3.4, 3.7, and 3.6, then the first moving range is missing, the second moving range is $|3.7 - 3.4| = 0.3$, and the third moving range is $|3.6 - 3.7| = 0.1$.

Central Lines

By default, the central line on an individual measurements chart indicates an estimate for μ, which is computed as $\bar{X}$. If you specify a known value (μ_0) for μ, the central line indicates the value of μ_0.

The central line on a moving range chart indicates an estimate for the expected moving range, which is computed as $d_2(n)\hat{\sigma}$, where $\hat{\sigma} = \bar{R}/d_2(n)$. If you specify a known value ($\hat{\sigma}_0$) for σ, the central line indicates the value of $d_2(n)\sigma_0$.

Control Limits

The control limits are computed as a specified multiple (k) of the standard errors of X_i and R_i above and below the central line. The default limits are computed using $k = 3$ (these are referred to as 3σ *limits*).

Table 32.21 provides the formulas for the limits.

Table 32.21 Limits for Individual Measurements and Moving Range Charts

Individual Measurements Chart
LCL = lower control limit = $\bar{X} - k\hat{\sigma}$ UCL = upper control limit = $\bar{X} + k\hat{\sigma}$
Moving Range Chart
LCL = lower control limit = $\max(d_2(n)\hat{\sigma} - kd_3(n)\hat{\sigma}, 0)$ UCL = upper control limit = $d_2(n)\hat{\sigma} + kd_3(n)\hat{\sigma}$

The formulas assume that the measurements are normally distributed. If the standard values μ_0 and σ_0 are available for μ and σ, replace $\bar{X}$ with μ_0 and $\hat{\sigma}$ with σ_0 in Table 32.21.

You can specify parameters for the limits as follows:

- Specify k by using the SIGMAS= option or the variable _SIGMAS_ in a LIMITS= data table.
- Specify n by using the LIMITN= option or the variable _LIMITN_ in a LIMITS= data table.
- Specify μ_0 by using the variable _MEAN_ in the LIMITS= data table.
- Specify σ_0 by using the variable _STDDEV_ in the LIMITS= data table.

Constructing Charts for Means

You produce control charts for means by using the BOXCHART, XCHART, XRCHART, or XSCHART statement. The following notation is used in this section:

μ	Process mean (expected value of the population of measurements)
σ	Process standard deviation (standard deviation of the population of measurements)
$\bar{X}_i$	Mean of measurements in ith subgroup
n_i	Sample size of ith subgroup
R_i	Range of measurements in ith subgroup
n_i	Sample size of ith subgroup
N	Number of subgroups
$\overline{\overline{X}}$	Weighted average of subgroup means

Subgroup Summary Statistics

Each point on an $\bar{X}$ chart indicates the value of a subgroup mean ($\bar{X}_i$). For example, if the 10th subgroup contains the values 12, 15, 19, 16, and 14, the subgroup summary statistic is

$$\bar{X}_{10} = \frac{12 + 15 + 19 + 16 + 14}{5} = 15.2$$

Central Line

By default, the central line on an $\bar{X}$ chart indicates an estimate for μ, which is computed as

$$\hat{\mu} = \overline{\overline{X}} = \frac{n_1\bar{X}_1 + \cdots + n_N\bar{X}_N}{n_1 + \cdots + n_N}$$

If you specify a known value (μ_0) for μ, the central line indicates the value of μ_0.

Control Limits

The limits are computed as a specified multiple (k) of the standard error of $\bar{X}_i$ above and below the central line. The default limits are computed using $k = 3$ (these are referred to as 3σ *limits*).

Table 32.23 provides the formulas for the limits.

Table 32.23 Limits for $\bar{X}$ Charts

LCL = lower limit = $\overline{\overline{X}} - k\hat{\sigma}\sqrt{n_i}$
UCL = upper limit = $\overline{\overline{X}} + k\hat{\sigma}\sqrt{n_i}$

Note that the limits vary with n_i. If the standard values μ_0 and σ_0 are available for μ and σ, respectively, replace $\overline{\overline{X}}$ with μ_0 and $\hat{\sigma}$ with σ_0 in Table 32.23.

You can specify parameters for the limits as follows:

- Specify k by using the SIGMAS= option or the variable _SIGMAS_ in a LIMITS= data table.
- Specify a constant nominal sample size $n_i \equiv n$ for the control limits by using the LIMITN= option or the variable _LIMITN_ in a LIMITS= data table.
- Specify μ_0 by using the variable _MEAN_ in a LIMITS= data table.
- Specify σ_0 by using the variable _STDDEV_ in a LIMITS= data table.

Constructing Charts for Medians

You produce control charts for medians by using the MCHART or MRCHART statement, and by using the BOXCHART statement when you specify CONTROLSTAT=MEDIAN. The following notation is used in this section:

μ	Process mean (expected value of the population of measurements)
σ	Process standard deviation (standard deviation of the population of measurements)
n_i	Sample size of ith subgroup
N	Number of subgroups
x_{ij}	jth measurement in the ith subgroup, $j = 1, 2, 3, \ldots, n_i$
$x_{i(j)}$	jth-largest measurement in the ith subgroup. Then $$x_{i(1)} \leq x_{i(2)} \leq \cdots \leq x_{i(n_i)}$$
$\overline{\overline{X}}$	Weighted average of subgroup means
M_i	Median of the measurements in the ith subgroup: $$M_i = \begin{cases} x_{i((n_i+1)/2)} & \text{if } n_i \text{ is odd} \\ (x_{i(n_i/2)} + x_{i((n_i/2)+1)})/2 & \text{if } n_i \text{ is even} \end{cases}$$
$\bar{M}$	Average of the subgroup medians: $$\bar{M} = (n_1 M_1 + \cdots + n_N M_N)/(n_1 + \cdots + n_N)$$
$\tilde{M}$	Median of the subgroup medians. Denote the jth-largest median by $M_{(j)}$ so that $M_{(1)} \leq M_{(2)} \leq \cdots \leq M_{(N)}$. Then $$\tilde{M} = \begin{cases} M_{((N+1)/2)} & \text{if } N \text{ is odd} \\ (M_{(N/2)} + M_{(N/2)+1})/2 & \text{if } N \text{ is even} \end{cases}$$
$e_M(n)$	Standard error of the median of n independent, normally distributed variables with unit standard deviation (the value of $e_M(n)$ can be calculated using the STDMED function in a DATA step)

Subgroup Summary Statistics

Each point on a median chart indicates the value of a subgroup median (M_i). For example, if the 10th contains the values 12, 15, 19, 16, and 14, the subgroup summary statistic is $M_{10} = 15$.

Central Line

The value of the central line indicates an estimate of μ, which is computed as follows:

- $\bar{M}$ by default
- $\overline{\overline{X}}$ when you specify MEDCENTRAL=AVGMEAN
- $\tilde{M}$ when you specify MEDCENTRAL=MEDMED

Control Limits

The control limits are computed as a specified multiple (k) of the standard error of M_i above and below the central line. The default limits are computed using $k = 3$ (these are referred to as 3σ *limits*).

Table 32.25 provides the formulas for the limits.

Table 32.25 Limits for Median Charts

LCL = lower limit = $\bar{M} - k\hat{\sigma}e_M(n_i)$
UCL = upper limit = $\bar{M} + k\hat{\sigma}e_M(n_i)$

Note that the limits vary with n_i. In Table 32.25, replace $\bar{M}$ with $\overline{\overline{X}}$ if you specify MEDCENTRAL=AVGMEAN, and replace $\bar{M}$ with $\tilde{M}$ if you specify MEDCENTRAL=MEDMED. The formulas assume that the data are normally distributed.

You can specify parameters for the limits as follows:

- Specify k by using the SIGMAS= option or the variable _SIGMAS_ in a LIMITS= data table.
- Specify a constant nominal sample size $n_i \equiv n$ for the control limits by using the LIMITN= option or the variable _LIMITN_ in a LIMITS= data table.
- Specify μ_0 by using the variable _MEAN_ in the LIMITS= data table.
- Specify σ_0 by using the variable _STDDEV_ in the LIMITS= data table.

Constructing Charts for Ranges

You produce control charts for ranges by using the MRCHART, RCHART, or XRCHART statement. The following notation is used in this section:

σ	Process standard deviation (standard deviation of the population of measurements)
R_i	Range of measurements in ith subgroup
n_i	Sample size of ith subgroup
$d_2(n)$	Expected value of the range of n independent normally distributed variables with unit standard deviation
$d_3(n)$	Standard error of the range of n independent observations from a normal population with unit standard deviation

Subgroup Summary Statistics

Each point on an R chart indicates the value of a subgroup range (R_i). For example, if the 10th subgroup contains the values 12, 15, 19, 16, and 14, the subgroup summary statistic is $R_{10} = 19 - 12 = 7$.

Central Line

By default, the central line for the ith subgroup indicates an estimate of the expected value of R_i, which is computed as $d_2(n_i)\hat{\sigma}$, where $\hat{\sigma}$ is an estimate of σ. If you specify a known value (σ_0) for σ, the central line indicates the value of $d_2(n_i)\sigma_0$. Note that the central line varies with n_i.

Control Limits

The control limits are computed as a specified multiple (k) of the standard error of R_i above and below the central line. The default limits are computed using $k = 3$ (these are referred to as 3σ *limits*).

Table 32.27 provides the formulas for the limits.

Table 32.27 Control Limits for R Charts

LCL = lower limit = $\max(d_2(n_i)\hat{\sigma} - k d_3(n_i)\hat{\sigma}, 0)$
UCL = upper limit = $d_2(n_i)\hat{\sigma} + k d_3(n_i)\hat{\sigma}$

The formulas assume that the data are normally distributed. Note that the control limits vary with n_i. If a standard value σ_0 is available for σ, replace $\hat{\sigma}$ with σ_0 in Table 32.27.

You can specify parameters for the limits as follows:

- Specify k by using the SIGMAS= option or the variable _SIGMAS_ in a LIMITS= data table.
- Specify a constant nominal sample size $n_i \equiv n$ for the control limits by using the LIMITN= option or the variable _LIMITN_ in a LIMITS= data table.
- Specify σ_0 by using the variable _STDDEV_ in a LIMITS= data table.

Constructing Charts for Standard Deviations

You produce control charts for standard deviations by using the SCHART or XSCHART statement. The following notation is used in this section:

σ	Process standard deviation (standard deviation of the population of measurements)
s_i	Standard deviation of measurements in ith subgroup

$$s_i = \sqrt{(1/(n_i - 1))((x_{i1} - \bar{X}_i)^2 + \cdots + (x_{in_i} - \bar{X}_i)^2)}$$

n_i	Sample size of ith subgroup
$c_4(n)$	Expected value of the standard deviation of n independent normally distributed variables with unit standard deviation
$c_5(n)$	Standard error of the standard deviation of n independent observations from a normal population with unit standard deviation

Subgroup Summary Statistics

Each point on an s chart indicates the value of a subgroup standard deviation (s_i). For example, if the 10th subgroup contains the values 12, 15, 19, 16, and 13, the subgroup summary statistic is

$$s_{10} = \sqrt{((12-15)^2 + (15-15)^2 + (19-15)^2 + (16-15)^2 + (13-15)^2)/4} = 2.739$$

Central Line

By default, the central line for the ith subgroup indicates an estimate for the expected value of s_i, which is computed as $c_4(n_i)\hat{\sigma}$, where $\hat{\sigma}$ is an estimate of σ. If you specify a known value (σ_0) for σ, the central line indicates the value of $c_4(n_i)\sigma_0$. Note that the central line varies with n_i.

Control Limits

The control limits are computed as a specified multiple (k) of the standard error of s_i above and below the central line. The default limits are computed using $k = 3$ (these are referred to as 3σ *limits*).

Table 32.29 provides the formulas for the limits.

Table 32.29 Control Limits for s Charts

LCL = lower limit = $\max(c_4(n_i)\hat{\sigma} - kc_5(n_i)\hat{\sigma}, 0)$
UCL = upper limit = $c_4(n_i)\hat{\sigma} + kc_5(n_i)\hat{\sigma}$

The formulas assume that the data are normally distributed. If a standard value σ_0 is available for σ, replace $\hat{\sigma}$ with σ_0 in Table 32.29. Note that the upper and lower limits vary with n_i.

You can specify parameters for the limits as follows:

- Specify k by using the SIGMAS= option or the variable _SIGMAS_ in a LIMITS= data table.
- Specify a constant nominal sample size $n_i \equiv n$ for the control limits by using the LIMITN= option or the variable _LIMITN_ in a LIMITS= data table.
- Specify σ_0 by using the variable _STDDEV_ in a LIMITS= data table.

Constructing Charts for Numbers of Nonconformities (c Charts)

You produce control charts for numbers of nonconformities by using the CCHART statement.

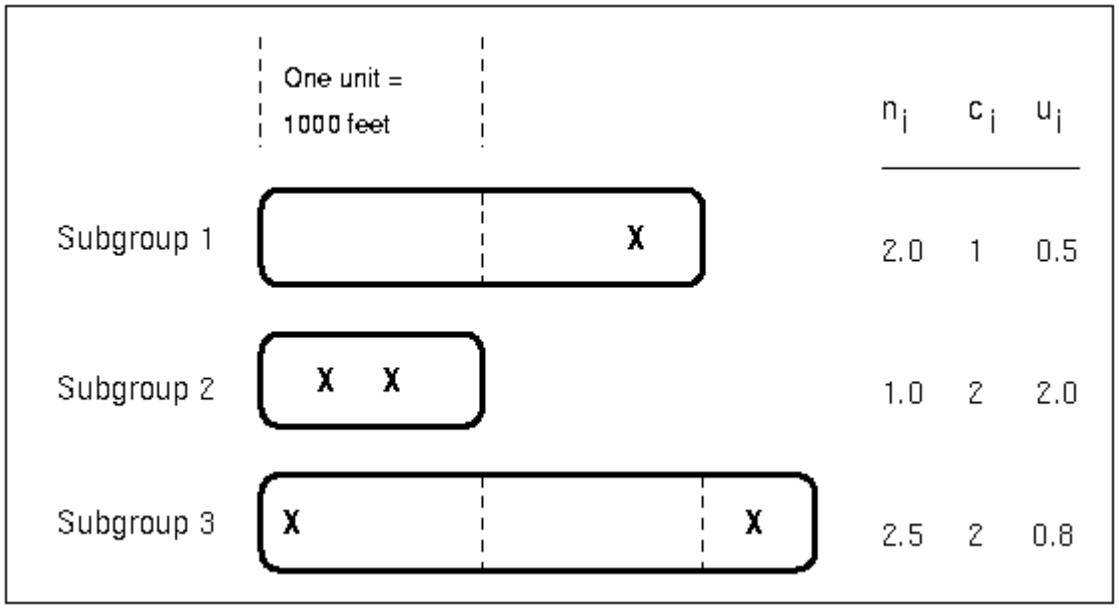
The following notation is used in this section:

u	Expected number of nonconformities per unit that the process produces
u_i	Number of nonconformities per unit in the ith subgroup
c_i	Total number of nonconformities in the ith subgroup
n_i	Number of inspection units in the ith subgroup. Typically, $n_i = 1$ and $u_i = c_i$ for c charts. In general, $u_i = c_i / n_i$.
$\bar{u}$	Average number of nonconformities per unit taken across subgroups. The quantity $\bar{u}$ is computed as a weighted average: $$\bar{u} = \frac{n_1 u_1 + \cdots + n_N u_N}{n_1 + \cdots + n_N} = \frac{c_1 + \cdots + c_N}{n_1 + \cdots + n_N}$$
N	Number of subgroups

Subgroup Summary Statistics

Each point on a c chart represents the total number of nonconformities (c_i) in a subgroup. For example, Figure 32.5 displays three sections of pipeline that are inspected for defective welds (indicated by an X). Each section represents a *subgroup* that consists of a number of *inspection units*, which are 1,000-foot sections. The number of units in the ith subgroup is denoted by n_i, which is the subgroup sample size. The value of n_i can be fractional; Figure 32.5 shows $n_3 = 2.5$ units in the third subgroup.

Figure 32.5 Terminology for c Charts and u Charts



The *number of nonconformities* in the ith subgroup is denoted by c_i. The *number of nonconformities per unit* in the ith subgroup is denoted by $u_i = c_i / n_i$. In Figure 32.5, the number of welds per inspection unit in the third subgroup is $u_3 = 2/2.5 = 0.8$.

A u chart that is created by the UCHART statement plots the quantity u_i for the ith subgroup (see the section "Constructing Charts for Nonconformities per Unit (u Charts)" on page 1687). An advantage of a u chart is that the value of the central line at the ith subgroup does not depend on n_i. This is not the case for a c chart, so a u chart is often preferred when the number of units n_i is not constant across subgroups.

Central Line

On a c chart, the central line indicates an estimate for $n_i u$, which is computed as $n_i \bar{u}$. If you specify a known value (u_0) for u, the central line indicates the value of $n_i u_0$.

Note that the central line varies with subgroup sample size n_i. When $n_i = 1$ for all subgroups, the central line has the constant value $\bar{c} = (c_1 + \cdots + c_N)/N$.

Control Limits

The control limits are computed as a specified multiple (k) of the standard error of c_i above and below the central line. The default limits are computed using $k = 3$ (these are referred to as 3σ *limits*).

The lower and upper control limits, LCL and UCL, respectively, are given by

$$\begin{aligned} \text{LCL} &= \max\left(n_i \bar{u} - k\sqrt{n_i \bar{u}}\ , 0\right) \\ \text{UCL} &= n_i \bar{u} + k\sqrt{n_i \bar{u}} \end{aligned}$$

The lower and upper control limits vary with the number of inspection units per subgroup n_i. If $n_i = 1$ for all subgroups, the control limits have constant values, as follows:

$$\begin{aligned} \text{LCL} &= \max\left(\bar{c} - k\sqrt{\bar{c}}\,, 0\right) \\ \text{UCL} &= \bar{c} + k\sqrt{\bar{c}} \end{aligned}$$

If a standard value u_0 is available for u, replace $\bar{u}$ with u_0 in the formulas for the control limits. You can specify parameters for the limits as follows:

- Specify k by using the SIGMAS= option or the variable _SIGMAS_ in a LIMITS= data table.
- Specify a constant nominal sample size $n_i \equiv n$ for the control limits by using the LIMITN= option or the variable _LIMITN_ in a LIMITS= data table.
- Specify u_0 by using the variable _U_ in a LIMITS= data table.

Constructing Charts for Number Nonconforming (np Charts)

You produce control charts for number nonconforming by using the NPCHART statement.

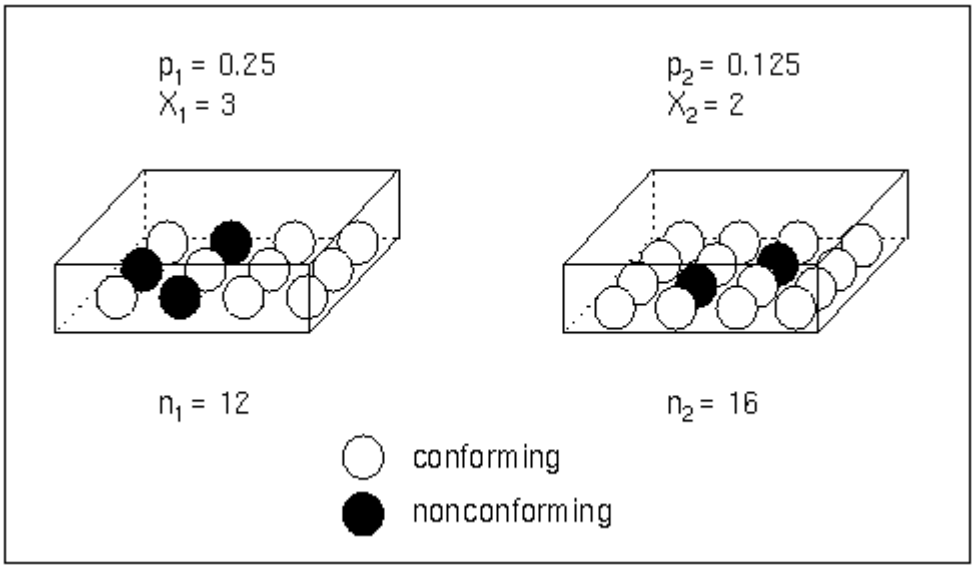
The following notation is used in this section:

p	Expected proportion of nonconforming items that the process produces
p_i	Proportion of nonconforming items in the ith subgroup
X_i	Number of nonconforming items in the ith subgroup
n_i	Number of items in the ith subgroup
$\bar{p}$	Average proportion of nonconforming items taken across subgroups: $\bar{p} = \frac{n_1 p_1 + \cdots + n_N p_N}{n_1 + \cdots + n_N} = \frac{X_1 + \cdots + X_N}{n_1 + \cdots + n_N}$
N	Number of subgroups

Subgroup Summary Statistics

Each point on an np chart represents the observed number (X_i) of nonconforming items in a subgroup. For example, in Figure 32.6 the first subgroup contains 12 items, of which 3 are nonconforming. The summary statistic for the first subgroup is $X_1 = 3$.

Figure 32.6 Proportions versus Counts



Note that a p chart displays the proportion of nonconforming items p_i. You can use the PCHART statement to create p charts; for more information, see the section "Constructing Charts for Proportion Nonconforming (p Charts)" on page 1686.

Central Line

By default, the central line on an np chart indicates an estimate for $n_i\, p$, which is computed as $n_i\, \bar{p}$. If you specify a known value (p_0) for p, the central line indicates the value of $n_i\, p_0$. Note that the central line varies with n_i.

Control Limits

The control limits are computed as a specified multiple (k) of the standard error of X_i above and below the central line. The default limits are computed using $k = 3$ (these are referred to as 3σ *limits*).

The lower and upper control limits, LCL and UCL, respectively, are computed as

$$\text{LCL} = \max\left(n_i\,\bar{p} - k\sqrt{n_i\,\bar{p}(1-\bar{p})}\,, 0\right)$$
$$\text{UCL} = \min\left(n_i\,\bar{p} + k\sqrt{n_i\,\bar{p}(1-\bar{p})}\,, n_i\right)$$

You can specify parameters for the limits as follows:

- Specify k by using the SIGMAS= option or the variable _SIGMAS_ in a LIMITS= data table.
- Specify a constant nominal sample size $n_i \equiv n$ for the control limits by using the LIMITN= option or the variable _LIMITN_ in a LIMITS= data table.
- Specify p_0 by using the variable _P_ in the LIMITS= data table.

Constructing Charts for Proportion Nonconforming (p Charts)

You produce control charts for proportion nonconforming by using the PCHART statement.

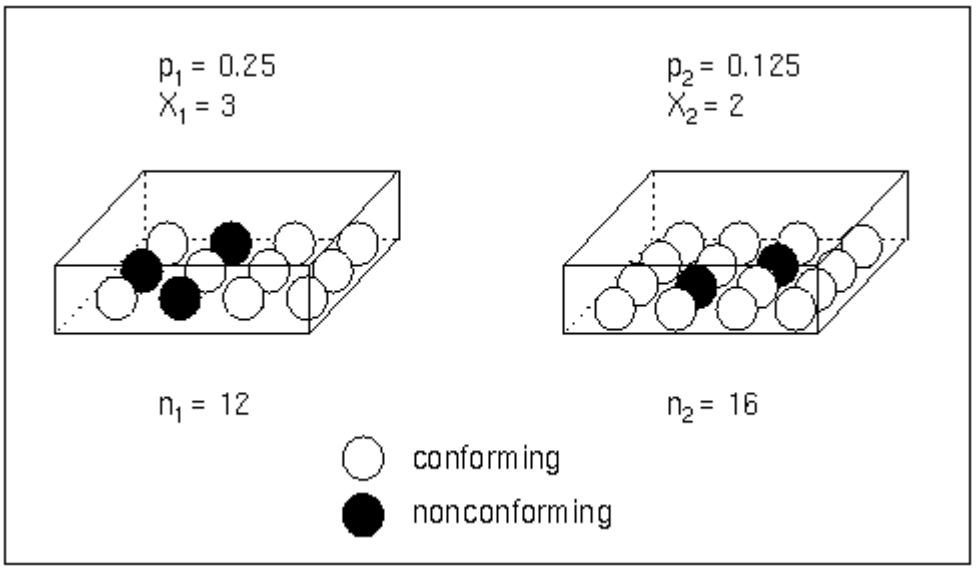
The following notation is used in this section:

p	Expected proportion of nonconforming items that the process produces
p_i	Proportion of nonconforming items in the ith subgroup
X_i	Number of nonconforming items in the ith subgroup
n_i	Number of items in the ith subgroup
$\bar{p}$	Average proportion of nonconforming items taken across subgroups: $\bar{p} = \frac{n_1 p_1 + \cdots + n_N p_N}{n_1 + \cdots + n_N} = \frac{X_1 + \cdots + X_N}{n_1 + \cdots + n_N}$
N	Number of subgroups

Subgroup Summary Statistics

Each point on a p chart represents the observed proportion ($p_i = X_i/n_i$) of nonconforming items in a subgroup. For example, in Figure 32.7 the second subgroup contains 16 items, of which 2 are nonconforming. The summary statistic for the second subgroup is $p_2 = 2/16 = 0.125$.

Figure 32.7 Proportions versus Counts



Note that an np chart displays the number (count) of nonconforming items X_i. You can use the NPCHART statement to create np charts; for more information, see the section "Constructing Charts for Number Nonconforming (np Charts)" on page 1684.

Central Line

By default, the central line on a p chart indicates an estimate of p that is computed as $\bar{p}$. If you specify a known value (p_0) for p, the central line indicates the value of p_0.

Control Limits

The control limits are computed as a specified multiple (k) of the standard error of p_i above and below the central line. The default limits are computed using $k = 3$ (these are referred to as 3σ *limits*).

The lower and upper control limits, LCL and UCL, respectively, are computed as

$$\begin{aligned} \text{LCL} &= \max\left(\bar{p} - k\sqrt{\bar{p}(1-\bar{p})/n_i}\ , 0\right) \\ \text{UCL} &= \min\left(\bar{p} + k\sqrt{\bar{p}(1-\bar{p})/n_i}\ , 1\right) \end{aligned}$$

You can specify parameters for the limits as follows:

- Specify k by using the SIGMAS= option or the variable _SIGMAS_ in a LIMITS= data table.
- Specify a constant nominal sample size $n_i \equiv n$ for the control limits by using the LIMITN= option or the variable _LIMITN_ in a LIMITS= data table.
- Specify p_0 by using the variable _P_ in a LIMITS= data table.

Constructing Charts for Nonconformities per Unit (u Charts)

You produce control charts for nonconformities per unit by using the UCHART statement.

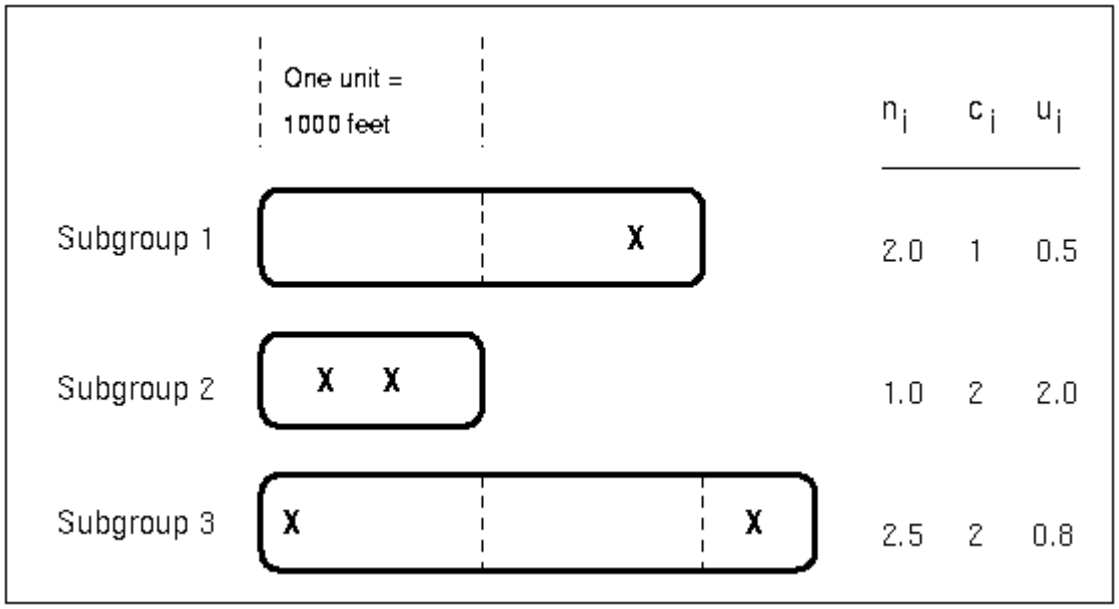
The following notation is used in this section:

u	Expected number of nonconformities per unit that the process produces
u_i	Number of nonconformities per unit in the ith subgroup. In general, $u_i = c_i/n_i$.
c_i	Total number of nonconformities in the ith subgroup
n_i	Number of inspection units in the ith subgroup
$\bar{u}$	Average number of nonconformities per unit taken across subgroups. The quantity $\bar{u}$ is computed as a weighted average: $\bar{u} = \frac{n_1u_1 + \cdots + n_Nu_N}{n_1 + \cdots + n_N} = \frac{c_1 + \cdots + c_N}{n_1 + \cdots + n_N}$
N	Number of subgroups

Subgroup Summary Statistics

Each point on a u chart indicates the number of nonconformities per unit (u_i) in a subgroup. For example, Figure 32.8 displays three sections of pipeline that are inspected for defective welds (indicated by an X). Each section represents a *subgroup* that consists of a number of *inspection units*, which are 1,000-foot sections. The number of units in the ith subgroup is denoted by n_i, which is the subgroup sample size.

Figure 32.8 Terminology for c Charts and u Charts



The *number of nonconformities* in the ith subgroup is denoted by c_i. The *number of nonconformities per unit* in the ith subgroup is denoted by $u_i = c_i/n_i$. In Figure 32.8, the number of defective welds per unit in the third subgroup is $u_3 = 2/2.5 = 0.8$.

A u chart plots the quantity u_i for the ith subgroup. A c chart plots the quantity c_i for the ith subgroup (for more information, see the section "Constructing Charts for Numbers of Nonconformities (c Charts)" on page 1682). An advantage of a u chart is that the value of the central line at the ith subgroup does not depend on n_i. This is not the case for a c chart, so a u chart is often preferred when the number of units n_i is not constant across subgroups.

Central Line

On a u chart, the central line indicates an estimate of u, which is computed as $\bar{u}$ by default. If you specify a known value (u_0) for u, the central line indicates the value of u_0.

Control Limits

The control limits are computed as a specified multiple (k) of the standard error of u_i above and below the central line. The default limits are computed using $k = 3$ (these are referred to as 3σ *limits*).

The lower and upper control limits, LCL and UCL, respectively, are given by

$$\begin{aligned} \text{LCL} &= \max\left(\bar{u} - k\sqrt{\bar{u}/n_i}\ , 0\right) \\ \text{UCL} &= \bar{u} + k\sqrt{\bar{u}/n_i} \end{aligned}$$

The limits vary with n_i.

You can specify parameters for the limits as follows:

- Specify k by using the SIGMAS= option or the variable _SIGMAS_ in a LIMITS= data table.
- Specify a constant nominal sample size $n_i \equiv n$ for the control limits by using the LIMITN= option or the variable _LIMITN_ in a LIMITS= data table.
- Specify u_0 by using the variable _U_ in a LIMITS= data table.

Constructing Uniformly Weighted Moving Average Charts

The following notation is used in this section:

A_i	Uniformly weighted moving average for the ith subgroup
w	Span parameter (number of terms in the moving average)
μ	Process mean (expected value of the population of measurements)
σ	Process standard deviation (standard deviation of the population of measurements)
x_{ij}	jth measurement in the ith subgroup, where j =1, 2, 3, . . . , n_i
n_i	Sample size of the ith subgroup
$\overline{X}_i$	Mean of measurements in the ith subgroup. If $n_i = 1$, then the subgroup mean reduces to the single observation in the subgroup.
$\overline{\overline{X}}$	Weighted average of subgroup means
$\Phi^{-1}(\cdot)$	Inverse standard normal function

Subgroup Summary Statistics

Each point on the chart indicates the value of the uniformly weighted moving average for that subgroup. The moving average for the ith subgroup (A_i) is defined as

$$A_i = (\overline{X}_1 + \cdots + \overline{X}_i)/i \qquad \text{if } i < w$$

$$A_i = (\overline{X}_i + \cdots + \overline{X}_{i-w+1})/w \quad \text{if } i \geq w$$

where w is the span, or number of terms, of the moving average. You can specify the span by using the SPAN= option in the MACHART statement or by including the value of the variable _SPAN_ in a LIMITS= data set.

Central Line

By default, the central line on a moving average chart indicates an estimate of μ, which is computed as

$$\hat{\mu} = \overline{\overline{X}} = \frac{n_1 \bar{X}_1 + \cdots + n_N \bar{X}_N}{n_1 + \cdots + n_N}$$

If you specify a known value (μ_0) for μ, the central line indicates the value of μ_0.

Control Limits

The limits are computed as a specified multiple (k) of the standard error of A_i above and below the central line. The default limits are computed using $k = 3$ (these are referred to as 3σ limits).

Table 32.35 provides the formulas for the limits:

Table 32.35 Limits for Moving Average Charts

$\text{LCL} = \overline{\overline{X}} - k(\hat{\sigma}/\min(i,w))\sqrt{(1/n_i) + (1/n_{i-1}) + \cdots + (1/n_{1+\max(i-w,0)})}$
$\text{UCL} = \overline{\overline{X}} + k(\hat{\sigma}/\min(i,w))\sqrt{(1/n_i) + (1/n_{i-1}) + \cdots + (1/n_{1+\max(i-w,0)})}$

These formulas assume that the data are normally distributed. If the standard values μ_0 and σ_0 are available for μ and σ, respectively, replace $\overline{\overline{X}}$ with μ_0 and replace $\hat{\sigma}$ with σ_0 in Table 32.35. Note that the limits vary with both n_i and i.

If the subgroup sample sizes are constant ($n_i = n$), the formulas for the control limits simplify to

$$\begin{aligned} \text{LCL} &= \overline{\overline{X}} - \frac{k\hat{\sigma}}{\sqrt{n \min(i,w)}} \\ \text{UCL} &= \overline{\overline{X}} + \frac{k\hat{\sigma}}{\sqrt{n \min(i,w)}} \end{aligned}$$

See Montgomery (1996) for more information. When the subgroup sample sizes are constant, the width of the control limits for the first w moving averages decreases monotonically, because each of the first w moving averages includes one more term than the preceding moving average.

If you specify the ASYMPTOTIC option, constant control limits that have the following values are computed:

$$\begin{aligned} \text{LCL} &= \overline{\overline{X}} - \frac{k\hat{\sigma}}{\sqrt{nw}} \\ \text{UCL} &= \overline{\overline{X}} + \frac{k\hat{\sigma}}{\sqrt{nw}} \end{aligned}$$

You can specify parameters for the moving average limits as follows:

- Specify k by using the SIGMAS= option or by including the variable _SIGMAS_ in a LIMITS= data set.

- Specify a constant nominal sample size $n_i \equiv n$ for the control limits by using the LIMITN= option or by including the variable _LIMITN_ in a LIMITS= data set.
- Specify w by using the SPAN= option or by including the variable _SPAN_ in a LIMITS= data set.
- Specify μ_0 by including the variable _MEAN_ in a LIMITS= data set.
- Specify σ_0 by including the variable _STDDEV_ in a LIMITS= data set.

Constructing Exponentially Weighted Moving Average Charts

You produce exponentially weighted moving average (EWMA) charts by using the EWMACHART statement.

The following notation is used in this section:

E_i	Exponentially weighted moving average for the ith subgroup
r	EWMA weight parameter ($0 < r \leq 1$)
μ	Process mean (expected value of the population of measurements)
σ	Process standard deviation (standard deviation of the population of measurements)
x_{ij}	jth measurement in the ith subgroup, where j =1, 2, 3, ..., n_i
n_i	Sample size of the ith subgroup
$\overline{X}_i$	Mean of measurements in the ith subgroup. If $n_i = 1$, then the subgroup mean reduces to the single observation in the subgroup.
$\overline{\overline{X}}$	Weighted average of subgroup means
$\Phi^{-1}(\cdot)$	Inverse standard normal function

Subgroup Summary Statistics

Each point on the chart indicates the value of the exponentially weighted moving average (EWMA) for that subgroup. The EWMA for the ith subgroup (E_i) is defined recursively as

$$E_i = r\overline{X}_i + (1-r)E_{i-1}, \quad i > 0$$

where r is a weight parameter ($0 < r \leq 1$). Some authors (such as Hunter 1986 and Crowder 1987a,b) use the symbol λ instead of r for the weight. You can specify the weight by using the WEIGHT= option in the EWMACHART statement or by including the variable _WEIGHT_ in a LIMITS= data table. If you specify a known value (μ_0) for μ, then $E_0 = \mu_0$; otherwise, $E_0 = \overline{\overline{X}}$.

The preceding equation can be rewritten as

$$E_i = E_{i-1} + r(\overline{X}_i - E_{i-1})$$

which expresses the current EWMA as the previous EWMA plus the weighted error in the prediction of the current mean based on the previous EWMA.

The EWMA for the ith subgroup can also be written as

$$E_i = r\sum_{j=0}^{i-1}(1-r)^j\overline{X}_{i-j} + (1-r)^i E_0$$

which expresses the EWMA as a weighted average of past subgroup means, where the weights decline exponentially and the heaviest weight is assigned to the most recent subgroup mean.

Central Line

By default, the central line on an EWMA chart indicates an estimate of μ, which is computed as

$$\hat{\mu} = \overline{\overline{X}} = \frac{n_1\bar{X}_1 + \cdots + n_N\bar{X}_N}{n_1 + \cdots + n_N}$$

If you specify a known value (μ_0) for μ, the central line indicates the value of μ_0.

Control Limits

The limits are computed as a specified multiple (k) of the standard error of E_i above and below the central line. The default limits are computed using $k = 3$ (these are referred to as 3σ *limits*).

Table 32.37 provides the formulas for the limits.

Table 32.37 Limits for EWMA Charts

LCL = lower limit = $\overline{\overline{X}} - k\hat{\sigma}r\sqrt{\sum_{j=0}^{i-1}(1-r)^{2j}/n_{i-j}}$
UCL = upper limit = $\overline{\overline{X}} + k\hat{\sigma}r\sqrt{\sum_{j=0}^{i-1}(1-r)^{2j}/n_{i-j}}$

These formulas assume that the data are normally distributed. If the standard values μ_0 and σ_0 are available for μ and σ, respectively, replace $\overline{\overline{X}}$ with μ_0 and $\hat{\sigma}$ with σ_0 in Table 32.37. Note that the limits vary with both n_i and i.

If the subgroup sample sizes are constant ($n_i = n$), the formulas for the control limits simplify to

$$\text{LCL} = \overline{\overline{X}} - k\hat{\sigma}\sqrt{r(1-(1-r)^{2i})/n(2-r)}$$

$$\text{UCL} = \overline{\overline{X}} + k\hat{\sigma}\sqrt{r(1-(1-r)^{2i})/n(2-r)}$$

Consequently, when the subgroup sample sizes are constant, the width of the control limits increases monotonically with i. See Roberts (1959) and Montgomery (1996).

As i becomes large, the upper and lower control limits approach constant values:

$$\text{LCL} = \overline{\overline{X}} - k\hat{\sigma}\sqrt{r/n(2-r)}$$

$$\text{UCL} = \overline{\overline{X}} + k\hat{\sigma}\sqrt{r/n(2-r)}$$

Some authors base the control limits for EWMA charts on the asymptotic expressions in the two previous equations. You can display asymptotic limits by specifying the ASYMPTOTIC option.

Uniformly weighted moving average charts and exponentially weighted moving average charts have similar properties, and their asymptotic control limits are identical, provided that

$$r = 2/(w+1)$$

where w is the weight factor for uniformly weighted moving average charts. See Wadsworth, Stephens, and Godfrey (1986) and the American Society for Quality Control (1983).

You can specify parameters for the EWMA limits as follows:

- Specify k by using the SIGMAS= option or by including the variable _SIGMAS_ in a LIMITS= data set.
- Specify a constant nominal sample size $n_i \equiv n$ for the control limits by using the LIMITN= option or by including the variable _LIMITN_ in a LIMITS= data set.
- Specify r by using the WEIGHT= option or by including the variable _WEIGHT_ in a LIMITS= data set.
- Specify μ_0 by including the variable _MEAN_ in a LIMITS= data set.
- Specify σ_0 by including the variable _STDDEV_ in a LIMITS= data set.

Percentile Definitions

You can use the PCTLDEF= option in the BOXCHART statement to specify one of five definitions for computing quantile statistics (percentiles). Let n equal the number of nonmissing values for a variable, and let $x_1, x_2, \ldots, x_n$ represent the ordered values of the process variable. For the tth percentile, set $p = t/100$, and express np as

$$np = j + g$$

where j is the integer part of np, and g is the fractional part of np.

The tth percentile (call it y) can be defined in five ways, as described in the next five sections.

PCTLDEF=1

This definition uses the weighted average at x_{np},

$$y = (1 - g)x_j + gx_{j+1}$$

where x_0 is taken to be x_1.

PCTLDEF=2

This definition uses the observation numbered closest to np,

$$y = x_i$$

where i is the integer part of $np + 1/2$.

PCTLDEF=3

This definition uses the empirical distribution function:

$$y = x_j \quad \text{if } g = 0$$
$$y = x_{j+1} \quad \text{if } g > 0$$

PCTLDEF=4

This definition uses the weighted average aimed at $x_{p(n+1)}$,

$$y = (1 - g)x_j + gx_{j+1}$$

where $(n + 1)p = j + g$, and where x_{n+1} is taken to be x_n.

PCTLDEF=5

This definition uses the empirical distribution function with averaging:

$$y = (x_j + x_{j+1})/2 \quad \text{if } g = 0$$
$$y = x_{j+1} \quad \text{if } g > 0$$

Tests for Special Causes

This section provides details about tests for special causes that you can apply by using the SPC procedure. These tests are not applicable to moving average charts that you compute by using the EWMACHART and MACHART statements.

The SPC procedure provides eight standard *tests for special causes*, also referred to as *rules for lack of control*, *supplementary rules*, *runs tests*, *runs rules*, *pattern tests*, and *Western Electric rules*. These tests improve the sensitivity of a Shewhart chart to small changes in the process. You can also improve the sensitivity of the chart by increasing the rate of sampling, increasing the subgroup sample size, and using control limits that represent less than three standard errors of variation from the central line. However, increasing the sampling rate and sample size is often impractical, and tightening the control limits increases the chances of falsely signaling an out-of-control condition. By detecting particular nonrandom patterns in the subgroup summary statistics, the tests can provide greater sensitivity and useful diagnostic information while incurring a reasonable probability of a false signal.

The patterns that the eight standard tests detect are defined in Table 32.38 and Table 32.39 and illustrated in Figure 32.9 and Figure 32.10. All eight tests were developed for use with fixed 3σ limits. The tests are indexed according to the numbering sequence used by Nelson (1984, 1985). You can request any combination of the eight tests by specifying the test *indices* in the TESTS= option in the BOXCHART, CCHART, IRCHART, MCHART, MRCHART, NPCHART, PCHART, UCHART, XCHART, XRCHART, and XSCHART statements.

The following restrictions apply to the tests:

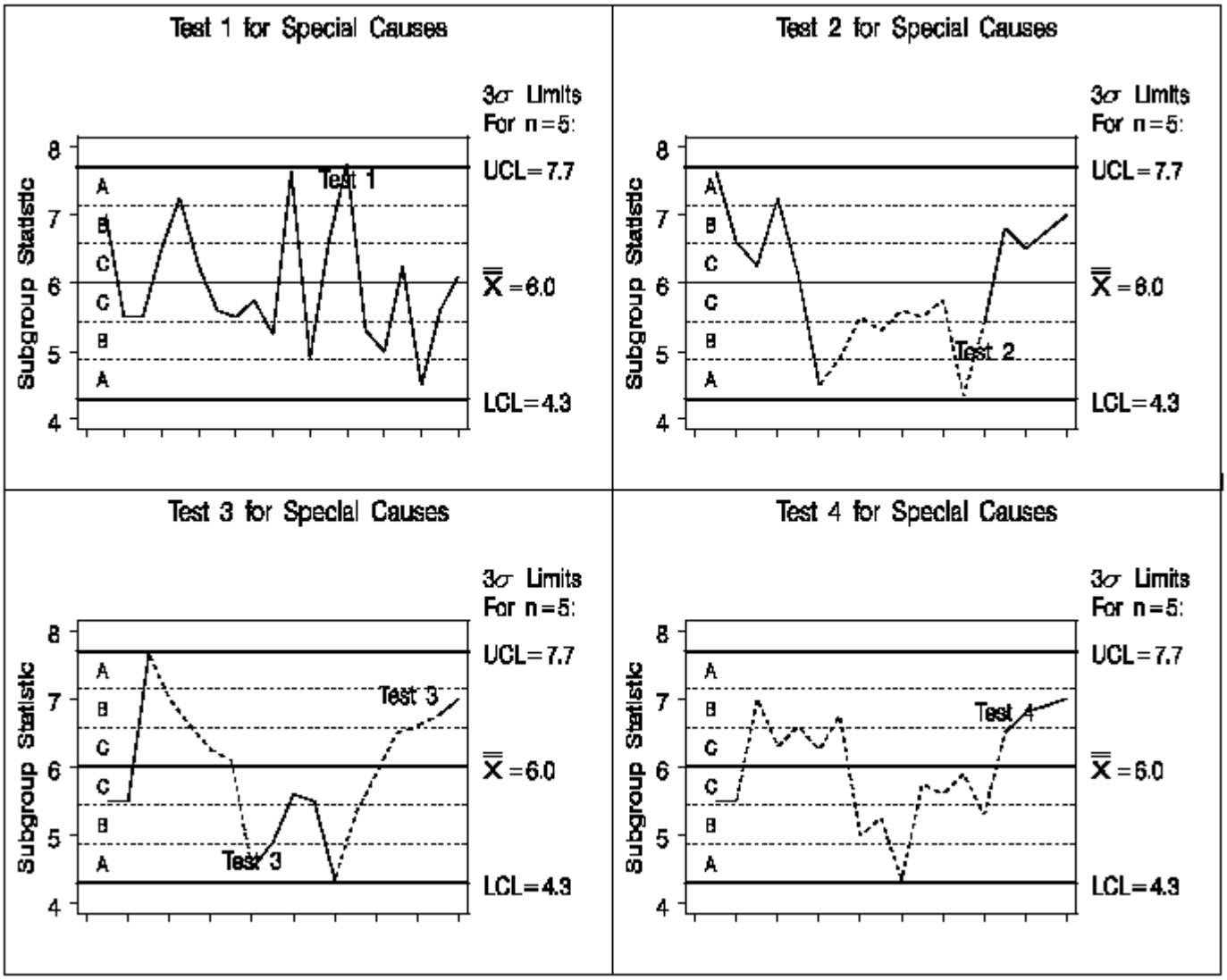
- Only Tests 1, 2, 3, and 4 are recommended for *c* charts, *np* charts, *p* charts, and *u* charts that you create using the CCHART, NPCHART, PCHART, and UCHART statements, respectively. In these four cases, you should not use Test 2 unless the process distribution is symmetric or nearly symmetric.

- By default, the TESTS= option is not applied when the control limits are not 3σ limits or vary with subgroup sample size. You can use the NO3SIGMACHECK option to request tests for special causes when the SIGMAS= option specifies control limits other than 3σ limits. This is not recommended for standard control chart applications, because the standard tests for special causes are based on 3σ limits. You can apply tests for special causes when control limits vary with subgroup sample size by using the LIMITN= or TESTNSTD option (for more information, see the section "Applying Tests with Varying Subgroup Sample Sizes" on page 1699).

Table 32.38 Definitions of Tests 1 to 4

Test Index	Pattern Description
1	One point beyond Zone A (outside the control limits)
2	Nine points in a row in Zone C or beyond on one side of the central line (see Note 1 after Figure 32.9)
3	Six points in a row that steadily increase or steadily decrease (see Note 2 after Figure 32.9)
4	Fourteen points in a row that alternate up and down

Figure 32.9 Examples of Tests 1 to 4



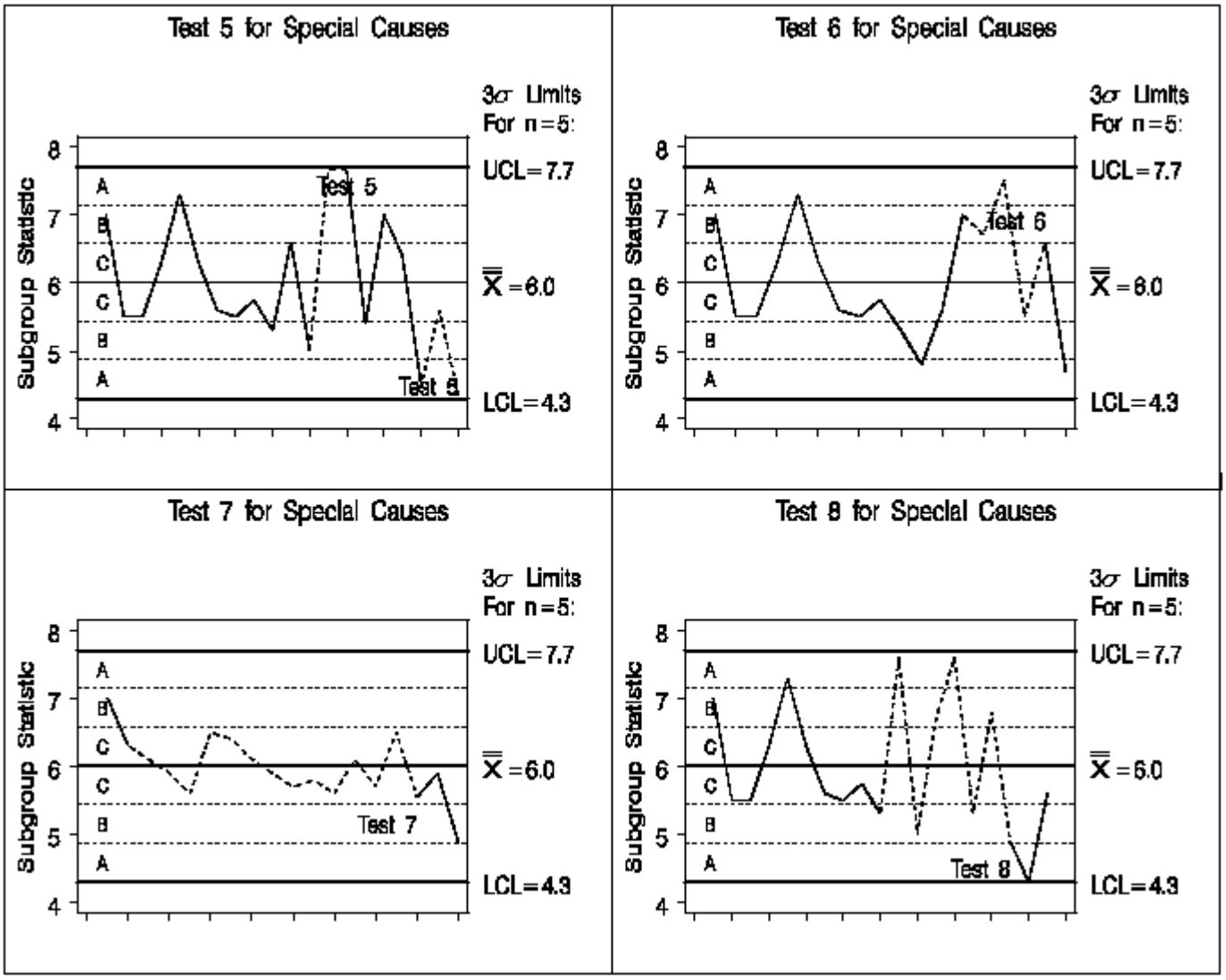
Notes:

1. You can specify the number of points in Test 2 as 7, 8, 9, 11, 14, or 20 in the TEST2RUN= option.
2. You can specify the number of points in Test 3 as 6, 7, or 8 in the TEST3RUN= option.

Table 32.39 Definitions of Tests 5 to 8

Test Index	Pattern Description
5	Two out of three points in a row in Zone A or beyond
6	Four out of five points in a row in Zone B or beyond
7	Fifteen points in a row in Zone C on either or both sides of the central line
8	Eight points in a row on either or both sides of the central line, with no points in Zone C

Figure 32.10 Examples of Tests 5 to 8



Interpreting Tests for Special Causes

Nelson (1984, 1985) makes the following comments about interpreting the tests for special causes:

- When a process is in statistical control, the chance of a false signal for each test is less than 5 in 1,000.
- Test 1 is positive if there is a shift in the process mean, if there is an increase in the process standard deviation, or if there is a "single aberration in the process such as a mistake in calculation, an error in measurement, bad raw material, a breakdown of equipment, and so on" (Nelson 1985).
- Test 2 signals a shift in the process mean. The use of nine points (rather than seven, as in Grant and Leavenworth 1988 for the pattern that defines Test 2 makes the chance of a false signal comparable to that of Test 1. (To control the number of points for the pattern in test 2, use the TEST2RUN= option in the chart statement.)
- Test 3 signals a drift in the process mean. Nelson (1985) states that causes can include "tool wear, depletion of chemical baths, deteriorating maintenance, improvement in skill, and so on."
- Test 4 signals "a systematic effect such as produced by two machines, spindles, operators or vendors used alternately" (Nelson 1985).
- Tests 1, 2, 3, and 4 should be applied routinely; the combined chance of a false signal from one or more of these tests is less than 1 in 100. Nelson (1985) describes these tests as "a good set that will react to many commonly occurring special causes."
- In the case of charts for variables, the first four tests should be augmented by Tests 5 and 6 when earlier warning is desired. The chance of a false signal increases to 2 in 100.
- Tests 7 and 8 indicate stratification (observations in a subgroup have multiple sources with different means). Test 7 is positive when the observations in the subgroup always have multiple sources. Test 8 is positive when the subgroups are taken from one source at a time.

Nelson (1985) also comments that "the probabilities quoted for getting false signals should not be considered to be very accurate" because the probabilities are based on assumptions of normality and independence that might not be satisfied. Consequently, he recommends that the tests "should be viewed as simply practical rules for action rather than tests having specific probabilities associated with them." Nelson cautions that "it is possible, though unlikely, for a process to be out of control yet not show any signals from these eight tests."

Modifying Standard Tests for Special Causes

Some textbooks and references present slightly different versions of Tests 2 and 3. You can use the following options to request these modifications:

- TEST2RUN=*run-length* specifies the length of the pattern for Test 2. The form of the test for each *run-length* is shown in the following table. The default *run-length* is 9.

run-length	**Number of Points on One Side of the Central Line**
7	7 in a row
8	8 in a row
9	9 in a row
11	At least 10 out of 11 in a row
14	At least 12 out of 14 in a row
20	At least 16 out of 20 in a row

- TEST3RUN=*run-length* specifies the length of the pattern for Test 3. The *run-length* values that you can specify are 6, 7, and 8. The default *run-length* is 6.

The Western Electric Company (now AT&T) *Statistical Quality Control Handbook* (1956) and Montgomery (1996) discuss a test that is signaled by eight points in a row in Zone C or beyond (on one side of the central line). You can request this test by specifying TESTS=2 and TEST2RUN=8. The *Handbook* also discusses tests that correspond to Tests 1, 5, 6, 7, and 8.

Kume (1985) recommends a number of tests for special causes that can be regarded as modifications of Tests 2 and 3:

- seven points in a row on one side of the central line. Specify TESTS=2 and TEST2RUN=7.
- at least 10 out of 11 points in a row on one side of the central line. Specify TESTS=2 and TEST2RUN=11.
- at least 12 out of 14 points in a row on one side of the central line. Specify TESTS=2 and TEST2RUN=14.
- at least 16 out of 20 points in a row on one side of the central line. Specify TESTS=2 and TEST2RUN=20.
- seven points in a row that steadily increase or decrease. Specify TESTS=3 and TEST3RUN=7.

Applying Tests with Varying Subgroup Sample Sizes

Nelson (1989, 1994) describes the use of standardization to apply the tests for special causes to data that involve varying subgroup samples. This approach applies the tests to the standardized subgroup statistics, setting the control limits at ± 3 and the zone boundaries at ± 1 and ± 2. For example, for an $\bar{X}$ chart with subgroup means $\bar{X}_i$ and varying subgroup sample sizes n_i, the tests are applied to the standardized values $z_i = (\bar{X}_i - \overline{\overline{X}})/(s/\sqrt{n_i})$, where $\overline{\overline{X}}$ estimates the process mean and s estimates the process standard deviation. You can request this method by using the TESTNSTD option.

Alternatively, you can handle varying subgroup sample sizes by specifying a nominal sample size in the LIMITN= option. If you specify LIMITN=*n*, the control limits are computed for the fixed value n, and they do not vary with the subgroup sample sizes. Moreover, subgroup summary statistics are computed only for those subgroups whose sample size is equal to n; only those subgroups are included in the results. You can specify the ALLN option in conjunction with the LIMITN= option to force subgroup summary statistics to be computed for all subgroups, regardless of subgroup sample size.

Capability Indices

This section provides formulas for process capability indices, which are computed for stable processes when you use the SPECS= option to specify a data table that contains process specification limits. The estimate $\hat{\sigma}$ is computed as described in the section "Methods of Estimating the Standard Deviation" on page 1672.

NOTE: You can specify the SPECS= option to compute capability indices only in the following chart statements that produce Shewhart charts for variables: BOXCHART, IRCHART, MCHART, MRCHART, RCHART, SCHART, XCHART, XRCHART, and XSCHART.

The Index C_p

The process capability index C_p is computed as

$$C_p = (\mathrm{USL} - \mathrm{LSL})/6\hat{\sigma}$$

If you do not specify both LSL and USL, the variable _CP_ is assigned a missing value.

The Index CPL

The process capability index CPL is computed as

$$\mathrm{CPL} = (\overline{\overline{X}} - \mathrm{LSL})/3\hat{\sigma}$$

If you do not specify LSL, the variable _CPL_ is assigned a missing value.

The Index CPU

The process capability index CPU is computed as

$$\mathrm{CPU} = (\mathrm{USL} - \overline{\overline{X}})/3\hat{\sigma}$$

If you do not specify USL, the variable _CPU_ is assigned a missing value.

The Index C_{pk}

The process capability index C_{pk} is computed as

$$C_{pk} = \min(\mathrm{USL} - \overline{\overline{X}}, \overline{\overline{X}} - \mathrm{LSL})/3\hat{\sigma}$$

If you specify only USL, the index C_{pk} is computed as

$$C_{pk} = (\mathrm{USL} - \overline{\overline{X}})/3\hat{\sigma}$$

and if you specify only LSL, the index C_{pk} is computed as

$$C_{pk} = (\overline{\overline{X}} - \mathrm{LSL})/3\hat{\sigma}$$

The Index C_{pm}

The process capability index C_{pm} is computed as

$$C_{pm} = \frac{\min(T - \mathrm{LSL}, \mathrm{USL} - T)}{3\sqrt{\hat{\sigma}^2 + (\overline{\overline{X}} - T)^2}}$$

where T is the target value that is specified in the Target variable in the SPECS= data table.

When you specify a single specification limit (SL) and target, C_{pm} is computed as

$$C_{pm} = \frac{|T - \mathrm{SL}|}{3\sqrt{\hat{\sigma}^2 + (\overline{\overline{X}} - T)^2}}$$

You should verify that a process is in statistical control before interpreting its capability indices, and you should verify that the data are normally distributed. A process must be stable and predictable before you can effectively address its ability to produce output within specifications. PROC SPC computes capability indices only for processes that do not show signs of being out of statistical control.

Some references use different notation and names for capability indices. For example, the manual ASQC Automotive Division/AIAG (1990) uses the term "process capability indices" for the indices that are computed by PROC SPC, and it uses the term "process performance indices" for indices that are computed by using the overall sample standard deviation as the estimate of $\hat{\sigma}$.

Input Data Tables

DATA= Data Table

The SPC procedure accepts a primary input data table that you specify in the DATA= option in the PROC SPC statement. This data table contains measurements of one or more process quality characteristics. Data to be analyzed by using the CCHART, IRCHART, NPCHART, PCHART, or UCHART statement should contain one observation per subgroup. Data to be analyzed by using other chart statements usually contain multiple observations per subgroup.

The DATA= data table must contain the following variables:

- the process name variable, which contains the names of the process variables. You specify it in the PROCESSNAME= option in the PROC SPC statement.
- the process value variable, which contains the process measurements. You specify it in the PROCESSVALUE= option in the PROC SPC statement.
- the subgroup name variables, which contains the names of the subgroup variables. You specify it in the SUBGROUPNAME= option in the PROC SPC statement.
- the subgroup value variable, which contains the subgroup values. You specify it in the SUBGROUPVALUE= option in the PROC SPC statement.

An input data table that contains attribute data (see the section "Classification of Control Charts" on page 1645) can also include a variable that contains the number of inspection units per subgroup. You specify this variable in the SUBGROUPN= option in the chart statement. It is required in the NPCHART, PCHART, and UCHART statements and optional in the CCHART statement.

The DATA= data table can also contain BY variables (see the section "BY Statement" on page 1652).

LIMITS= Data Table

You can read preestablished control limits or parameters from which the control limits can be calculated from a LIMITS= data table that you specify by using the LIMITS= option in the PROC SPC statement. The LIMITS= data table can be an OUTLIMITS= data table that was created in a previous run of the SPC procedure. Such data tables always contain the variables that a LIMITS= data table requires. For more information about the variables that are included in an OUTLIMITS= data table, see the section "OUTLIMITS= Data Table" on page 1704.

The LIMITS= data table must contain the following variables:

- the process name variable, which contains the names of the process variables. You specify it in the PROCESSNAME= option in the PROC SPC statement.
- the subgroup name variable, which contains the names of the subgroup variables. You specify it in the SUBGROUPNAME= option in the PROC SPC statement.

For each control chart type, Table 32.40 lists the variables that are required to specify parameters from which control limits are calculated and the variables that contain preestablished control limits.

Table 32.40 Variables in the LIMITS= Data Table

Chart Type	Parameters	Limits
c	_U_	_LCLC_, _C_, _UCLC_
EWMA	_MEAN_, _STDDEV_, _WEIGHT_	N/A
Individual measurements	_MEAN_, _STDDEV_	_LCLI_, _MEAN_, _UCLI_
Moving average	_MEAN_, _STDDEV_, _SPAN_	N/A
Moving range	_STDDEV_	_LCLR_, _R_, _UCLR_
Medians	_MEAN_, _STDDEV_	_LCLM_, _MEAN_, _UCLM_
np	_P_	_LCLNP_, _NP_, _UCLNP_
p	_P_	_LCLP_, _P_, _UCLP_
R	_STDDEV_	_LCLR_, _R_, _UCLR_
s	_STDDEV_	_LCLS_, _S_, _UCLS_
u	_U_	_LCLU_, _U_, _UCLU_
$\bar{X}$	_MEAN_, _STDDEV_	_LCLX_, _MEAN_, _UCLX_

Notes:

1. You can include the variable _LIMITN_, with or without parameters and limits variables, to specify different nominal subgroup sizes for different processes.
2. The special missing value *V* indicates variables that vary with subgroup sample size. Their values are calculated from the data. Table 32.42 identifies the variables that are assigned the missing value *V* in an OUTLIMITS= data table when subgroup sample sizes vary.
3. The variables _SIGMAS_ and _TYPE_ are optional, but they are recommended to maintain a complete set of control limit information. The variable _TYPE_ must be a character variable of length 8; valid values are 'ESTIMATE', 'STANDARD', 'STDMU', and 'STDSIGMA'.
4. For the EWMA and moving average charts, the LIMITS= data table contains only control limit parameters, not the control limits themselves.
5. If you specify a BY statement, the LIMITS= data table must also contain the specified BY variables.

SPECS= Data Table

You can provide process specification limits in a data table by using the SPECS= option. PROC SPC uses specification limits to calculate capability indices for processes. This option is available only in the following statements that produce Shewhart charts for variables: BOXCHART, IRCHART, MCHART, MRCHART, RCHART, SCHART, XCHART, XRCHART, and XSCHART.

The SPECS= data table must include the process name variable, which, as the name suggests, contains the names of the process variables. You specify it in the PROCESSNAME= option in the PROC SPC statement. Table 32.41 lists the specification limit variables that the data table can include.

Table 32.41 SPECS= Data Table Variables

Variable	Description
LSL	Lower specification limit
Target	Target value
USL	Upper specification limit

PROC SPC computes capability indices for each process that is included in the DATA= data table and whose specification limits are provided in the SPECS= data table. For more information about how capability indices are computed, see the section "Capability Indices" on page 1700.

Output Data Tables

OUTLIMITS= Data Table

An OUTLIMITS= data table contains control limits and control limit parameters. It contains one observation for each process. The following variables are always included in an OUTLIMITS= data table:

- the process name variable, which contains the names of the process variables. You specify it in the PROCESSNAME= option in the PROC SPC statement.
- the subgroup name variable, which contains the names of the subgroup variables. You specify it in the SUBGROUPNAME= option in the PROC SPC statement.

In addition, if you specify any BY variables, they are included in the OUTLIMITS= data table.

The other variables that are included in an OUTLIMITS= data table depend on which chart statement produced that data table. Table 32.42 lists the variables that are always included in OUTLIMITS= data tables for the different chart statements.

Table 32.42 Variables in the OUTLIMITS= Data Table

Variable	Varying	Description
BOXCHART Statement		
LCLM	✓	Lower control limit for subgroup median
LCLR	✓	Lower control limit for subgroup range
LCLS	✓	Lower control limit for subgroup standard deviation
LCLX	✓	Lower control limit for subgroup mean
LIMITN	✓	Nominal sample size associated with the control limits
MEAN		Process mean (value of central line on box chart)
_R_	✓	Value of central line on R chart
_S_	✓	Value of central line on s chart
SIGMAS		Multiple (k) of standard error of $\bar{X}_i$ or M_i
STDDEV		Process standard deviation
TYPE		Type (estimate or standard value) of _MEAN_ and _STDDEV_
UCLM	✓	Upper control limit for subgroup median

Table 32.42 *continued*

Variable	Varying	Description
UCLR	✓	Upper control limit for subgroup range
UCLS	✓	Upper control limit for subgroup standard deviation
UCLX	✓	Upper control limit for subgroup mean
CCHART Statement		
_C_	✓	Value of central line on c chart ($n_i\bar{u}$ or $n_i u_0$)
LCLC	✓	Lower control limit for number of nonconformities
LIMITN	✓	Sample size associated with the control limits
SIGMAS		Multiple (k) of standard error of c_i
TYPE		Type (estimate or standard value) of _U_
_U_		Average number of nonconformities per unit ($\bar{u}$ or u_0)
UCLC	✓	Upper control limit for number of nonconformities
EWMACHART Statement		
LIMITN	✓	Sample size associated with the control limits
MEAN		Process mean
SIGMAS		Multiple (k) of standard error of exponentially weighted moving averages
STDDEV		Process standard deviation
TYPE		Type (estimate or standard value) of _MEAN_ and _STDDEV_
WEIGHT		Weight parameter used to compute the EWMA
IRCHART Statement		
LCLI		Lower control limit for individual measurements
LCLR		Lower control limit for moving ranges
LIMITN		Number of consecutive measurements that are used to compute moving ranges
MEAN		Process mean
_R_		Value of central line on moving range chart
SIGMAS		Multiple (k) of standard error of individual measurement or moving range
STDDEV		Process standard deviation
TYPE		Type (estimate or standard value) of _MEAN_ and _STDDEV_
UCLI		Upper control limit for individual measurements
UCLR		Upper control limit for moving ranges
MACHART Statement		
LIMITN	✓	Sample size associated with the control limits
MEAN		Process mean
SIGMAS		Multiple (k) of standard error of moving averages
SPAN		Number of terms in the moving average
STDDEV		Process standard deviation
TYPE		Type (estimate or standard value) of _MEAN_ and _STDDEV_

Table 32.42 *continued*

Variable	Varying	Description
MCHART Statement		
LCLM	✓	Lower control limit for subgroup median
LCLR	✓	Lower control limit for subgroup range
LCLS	✓	Lower control limit for subgroup standard deviation
LIMITN	✓	Sample size associated with the control limits
MEAN		Value of central line on median chart ($\bar{M}$, $\tilde{M}$, $\overline{\overline{X}}$, or μ_0)
_R_	✓	Value of central line on R chart
_S_	✓	Value of central line on s chart
SIGMAS		Multiple (k) of standard error of M_i
STDDEV		Process standard deviation
TYPE		Type (estimate or standard value) of _MEAN_ and _STDDEV_
UCLM	✓	Upper control limit for subgroup median
UCLR	✓	Upper control limit for subgroup range
UCLS	✓	Upper control limit for subgroup standard deviation
MRCHART Statement		
LCLM	✓	Lower control limit for subgroup median
LCLR	✓	Lower control limit for subgroup range
LIMITN	✓	Sample size associated with the control limits
MEAN		Estimate of process mean ($\bar{M}$, $\tilde{M}$, $\overline{\overline{X}}$, or μ_0)
_R_	✓	Value of central line on range chart
SIGMAS		Multiple (k) of standard error of M_i or R_i
STDDEV		Process standard deviation
TYPE		Type (estimate or standard value) of _MEAN_ and _STDDEV_
UCLM	✓	Upper control limit for subgroup median
UCLR	✓	Upper control limit for subgroup range
NPCHART Statement		
LCLNP	✓	Lower control limit for number of nonconforming items
LIMITN	✓	Sample size associated with the control limits
_NP_	✓	Average number of nonconforming items ($n_i\,\bar{p}$ or $n_i\,p_0$)
_P_		Average proportion of nonconforming items ($\bar{p}$ or p_0)
SIGMAS		Multiple (k) of standard error of X_i
TYPE		Type (standard or estimate) of _NP_
UCLNP	✓	Upper control limit for number of nonconforming items
PCHART Statement		
LCLP	✓	Lower control limit for proportion of nonconforming items
LIMITN	✓	Nominal sample size associated with the control limits
_P_		Average proportion of nonconforming items ($\bar{p}$ or p_0)
SIGMAS		Multiple (k) of standard error of p_i
TYPE		Type (standard or estimate) of _P_
UCLP	✓	Upper control limit for proportion of nonconforming items

Table 32.42 *continued*

Variable	Varying	Description
RCHART Statement		
LCLR	✓	Lower control limit for subgroup range
LCLX	✓	Lower control limit for subgroup mean
LIMITN	✓	Sample size associated with the control limits
MEAN		Process mean ($\overline{\overline{X}}$)
_R_	✓	Value of central line on R chart
SIGMAS		Multiple (k) of standard error of R_i
STDDEV		Process standard deviation
TYPE		Type (estimate or standard value) of _MEAN_ and _STDDEV_
UCLR	✓	Upper control limit for subgroup range
UCLX	✓	Upper control limit for subgroup mean
SCHART Statement		
LCLS	✓	Lower control limit for subgroup standard deviation
LCLX	✓	Lower control limit for subgroup mean
LIMITN	✓	Sample size associated with the control limits
MEAN		Process mean ($\overline{\overline{X}}$ or μ_0)
_S_	✓	Value of central line on s chart
SIGMAS		Multiple (k) of standard error of $\bar{X}_i$ or s_i
STDDEV		Process standard deviation
TYPE		Type (estimate or standard value) of _MEAN_ and _STDDEV_
UCLS	✓	Upper control limit for subgroup standard deviation
UCLX	✓	Upper control limit for subgroup mean
UCHART Statement		
LCLU	✓	Lower control limit for number of nonconformities per unit
LIMITN	✓	Sample size associated with the control limits
SIGMAS		Multiple (k) of standard error of u_i
TYPE		Type (estimate or standard value) of _U_
_U_		Value of central line of u chart ($\bar{u}$ or u_0)
UCLU	✓	Upper control limit for number of nonconformities per unit
XCHART Statement		
LCLR	✓	Lower control limit for subgroup range
LCLS	✓	Lower control limit for subgroup standard deviation
LCLX	✓	Lower control limit for subgroup mean
LIMITN	✓	Sample size associated with the control limits
MEAN		Process mean ($\overline{\overline{X}}$ or μ_0)
_R_	✓	Value of central line on R chart
_S_	✓	Value of central line on s chart
SIGMAS		Multiple (k) of standard error of $\bar{X}_i$
STDDEV		Process standard deviation

Table 32.42 *continued*

Variable	Varying	Description
TYPE		Type (estimate or standard value) of _MEAN_ and _STDDEV_
UCLR	✓	Upper control limit for subgroup range
UCLS	✓	Upper control limit for subgroup standard deviation
UCLX	✓	Upper control limit for subgroup mean
XRCHART Statement		
LCLR	✓	Lower control limit for subgroup range
LCLX	✓	Lower control limit for subgroup mean
LIMITN	✓	Nominal sample size associated with the control limits
MEAN		Process mean ($\overline{\overline{X}}$ or μ_0)
_R_	✓	Value of central line on R chart
SIGMAS		Multiple (k) of standard error of $\bar{X}_i$ or R_i
STDDEV		Process standard deviation
TYPE		Type (estimate or standard value) of _MEAN_ and _STDDEV_
UCLR	✓	Upper control limit for subgroup range
UCLX	✓	Upper control limit for subgroup mean
XSCHART Statement		
LCLS	✓	Lower control limit for subgroup standard deviation
LCLX	✓	Lower control limit for subgroup mean
LIMITN	✓	Nominal sample size associated with the control limits
MEAN		Process mean ($\overline{\overline{X}}$ or μ_0)
_S_	✓	Value of central line on s chart
SIGMAS		Multiple (k) of standard error of $\bar{X}_i$ or s_i
STDDEV		Process standard deviation
TYPE		Type (estimate or standard value) of _MEAN_ and _STDDEV_
UCLS	✓	Upper control limit for subgroup standard deviation
UCLX	✓	Upper control limit for subgroup mean

Notes:

1. If the control limits vary with subgroup sample size, the special missing value V is assigned to the variables that are indicated by a check mark (✓) in the Varying column of Table 32.42.

2. For all Shewhart charts for variables, the OUTLIMITS= data table includes control limits for means or medians and for ranges or standard deviations. This enables you to use the OUTLIMITS= data table as a LIMITS= data table for different charts for variables.

3. For the BOXCHART statement, the variables _LCLM_ and _UCLM_ are included if you specify CONTROLSTAT=MEDIAN; otherwise, the variables _LCLX_ and _UCLX_ are included. The variables _LCLR_, _R_, and _UCLR_ are included if you specify SMETHOD=RMVLUE or RNOWEIGHT; otherwise, the variables _LCLS_, _S_, and _UCLS_ are included.

4. For the MCHART and XCHART statements, the variables _LCLS_, _S_, and _UCLS_ are included if you specify SMETHOD=RMSDF, SMVLUE, or SNOWEIGHT; otherwise, the variables _LCLR_, _R_, and _UCLR_ are included.

5. For the EWMACHART and MACHART statements, OUTLIMITS= data tables contain only control limit parameters, not the control limits themselves.

If you specify the SPECS= option, the OUTLIMITS= data table also includes the variables that are listed in Table 32.43.

Table 32.43 Additional OUTLIMITS= Data Table Variables

Variable	Description
LSL	Lower specification limit
TARGET	Target value
USL	Upper specification limit
_CP_	Process capability index C_p
CPL	Process capability index CPL
CPU	Process capability index CPU
CPK	Process capability index C_{pk}
CPM	Process capability index C_{pm}

NOTE: Capability indices are computed only for processes that are in statistical control.

If you specify both the SPECS= and CIINDICES options, the OUTLIMITS= data table includes the additional variables that are listed in Table 32.44.

Table 32.44 Additional OUTLIMITS= Data Table Variables

Variable	Description
_CP_LCL_	Lower confidence limit for C_p
_CP_UCL_	Upper confidence limit for C_p
_CPL_LCL_	Lower confidence limit for CPL
_CPL_UCL_	Upper confidence limit for CPL
_CPU_LCL_	Lower confidence limit for CPU
_CPU_UCL_	Upper confidence limit for CPU
_CPK_LCL_	Lower confidence limit for C_{pk}
_CPK_UCL_	Upper confidence limit for C_{pk}
_CPM_LCL_	Lower confidence limit for C_{pm}
_CPM_UCL_	Upper confidence limit for C_{pm}

NOTE: Confidence limits are computed for C_{pm} only when both an LSL and a USL are specified and when the target value is halfway between the LSL and the USL.

OUTTABLE= Data Table

An OUTTABLE= data table contains subgroup summary statistics, control limits, and related information. The following variables are always included in an OUTTABLE= data table:

- _VAR_, which contains the process variable name
- the subgroup name variable, which contains subgroup variable names. You specify the name of the subgroup name variable in the SUBGROUPNAME= option in the PROC statement.
- the subgroup variable, which contains subgroup variable values. You specify the name of the subgroup variable in the SUBGROUPVALUE= option in the PROC statement.

In addition, if you specify any BY variables, they are included in the OUTTABLE= data table.

The other variables that are included in an OUTTABLE= data table depend on which chart statement produced that data table. Table 32.45 lists the variables that are included in OUTTABLE= data tables for the different chart statements.

Table 32.45 Variables in the OUTTABLE= Data Table

Variable	Description
BOXCHART Statement	
EXLIM	Control limit exceeded on box chart
LCLM	Lower control limit for median
LCLX	Lower control limit for mean
LIMITN	Nominal sample size associated with the control limits
MEAN	Process mean
SIGMAS	Multiple (k) of the standard error associated with the control limits
STDDEV	Process standard deviation ($\hat{\sigma}$ or σ_0)
SUBMAX	Subgroup maximum
SUBMED	Subgroup median
SUBMIN	Subgroup minimum
SUBN	Subgroup sample size
SUBQ1	Subgroup first quartile (25th percentile)
SUBQ3	Subgroup third quartile (75th percentile)
SUBX	Subgroup mean
TESTS	Tests for special causes signaled on box chart
UCLM	Upper control limit for median
UCLX	Upper control limit for mean
CCHART Statement	
_C_	Average number of nonconformities
EXLIM	Control limit exceeded on c chart
LCLC	Lower control limit for number of nonconformities
LIMITN	Nominal sample size associated with the control limits
SIGMAS	Multiple (k) of the standard error associated with the control limits
SUBC	Subgroup number of nonconformities
SUBN	Subgroup sample size

Table 32.45 *continued*

Variable	Description
TESTS	Tests for special causes signaled on c chart
UCLC	Upper control limit for number of nonconformities
EWMACHART Statement	
EWMA	Exponentially weighted moving average
EXLIM	Control limit exceeded on EWMA chart
LCLE	Lower control limit for EWMA
LIMITN	Nominal sample size associated with control limits
MEAN	Process mean
SIGMAS	Multiple (k) of the standard error associated with control limits
SUBN	Subgroup sample size
SUBS	Subgroup standard deviation
SUBX	Subgroup mean
UCLE	Upper control limit for EWMA
WEIGHT	Weight parameter used to compute EWMA
IRCHART Statement	
EXLIM	Control limit exceeded on individual measurements chart
EXLIMR	Control limit exceeded on moving range chart
LCLI	Lower control limit for individual measurements
LCLR	Lower control limit for moving range
LIMITN	Number of consecutive measurements that are used to compute moving ranges
MEAN	Process mean
_R_	Average range
SIGMAS	Multiple (k) of the standard error associated with the control limits
STDDEV	Process standard deviation ($\hat{\sigma}$ or σ_0)
SUBI	Individual measurement
SUBR	Moving range
TESTS	Tests for special causes signaled on individual measurements chart
UCLI	Upper control limit for individual measurements
UCLR	Upper control limit for moving range
MACHART Statement	
EXLIM	Control limit exceeded on moving average chart
LCLA	Lower control limit for moving average
LIMITN	Nominal sample size associated with control limits
MEAN	Process mean
SIGMAS	Multiple (k) of the standard error associated with control limits
SPAN	Number of terms in moving average
SUBN	Subgroup sample size
SUBS	Subgroup standard deviation
SUBX	Subgroup mean
UCLA	Upper control limit for moving average

Table 32.45 *continued*

Variable	Description
UWMA	Uniformly weighted moving average
MCHART Statement	
EXLIM	Control limit exceeded on median chart
LCLM	Lower control limit for median
LIMITN	Nominal sample size associated with the control limits
MEAN	Estimate of process mean ($\bar{\tilde{M}}$, $\tilde{\tilde{M}}$, $\overline{\overline{X}}$, or μ_0)
SIGMAS	Multiple (k) of the standard error associated with the control limits
STDDEV	Process standard deviation ($\hat{\sigma}$ or σ_0)
SUBMED	Subgroup median
SUBN	Subgroup sample size
TESTS	Tests for special causes signaled on median chart
UCLM	Upper control limit for median
MRCHART Statement	
EXLIM	Control limit exceeded on median chart
EXLIMR	Control limit exceeded on range chart
LCLM	Lower control limit for median
LCLR	Lower control limit for range
LIMITN	Nominal sample size associated with the control limits
MEAN	Estimate of process mean ($\bar{\tilde{M}}$, $\tilde{\tilde{M}}$, $\overline{\overline{X}}$, or μ_0)
_R_	Average range
SIGMAS	Multiple (k) of the standard error associated with the control limits
STDDEV	Process standard deviation ($\hat{\sigma}$ or σ_0)
SUBM	Subgroup median
SUBN	Subgroup sample size
SUBR	Subgroup range
TESTS	Tests for special causes signaled on median chart
TESTS2	Tests for special causes signaled on range chart
UCLM	Upper control limit for mean
UCLR	Upper control limit for range
NPCHART Statement	
EXLIM	Control limit exceeded on np chart
LCLNP	Lower control limit for number of nonconforming items
LIMITN	Nominal sample size associated with the control limits
_NP_	Average number of nonconforming items
SIGMAS	Multiple (k) of the standard error of X_i associated with the control limits
SUBNP	Subgroup number of nonconforming items
SUBN	Subgroup sample size
TESTS	Tests for special causes signaled on np chart
UCLNP	Upper control limit for number of nonconforming items

Table 32.45 *continued*

Variable	Description
PCHART Statement	
EXLIM	Control limit exceeded on p chart
LCLP	Lower control limit for proportion of nonconforming items
LIMITN	Nominal sample size associated with the control limits
_P_	Average proportion of nonconforming items
SIGMAS	Multiple (k) of the standard error of p_i associated with the control limits
SUBP	Subgroup proportion of nonconforming items
SUBN	Subgroup sample size
TESTS	Tests for special causes signaled on p chart
UCLP	Upper control limit for proportion of nonconforming items
RCHART Statement	
EXLIM	Control limit exceeded on R chart
LCLR	Lower control limit for range
LIMITN	Nominal sample size associated with the control limits
_R_	Average range
SIGMAS	Multiple (k) of the standard error associated with the control limits
STDDEV	Process standard deviation ($\hat{\sigma}$ or σ_0)
SUBN	Subgroup sample sizes
SUBR	Subgroup range
TESTS2	Tests for special causes signaled on R chart
UCLR	Upper control limit for range
SCHART Statement	
EXLIM	Control limit exceeded on s chart
LCLS	Lower control limit for standard deviation
LIMITN	Nominal sample size associated with the control limits
_S_	Average standard deviation
SIGMAS	Multiple (k) of the standard error associated with the control limits
STDDEV	Process standard deviation ($\hat{\sigma}$ or σ_0)
STDDEV	Process standard deviation ($\hat{\sigma}$ or σ_0)
SUBN	Subgroup sample size
SUBS	Subgroup standard deviation
TESTS2	Tests for special causes signaled on s chart
UCLS	Upper control limit for standard deviation
UCHART Statement	
EXLIM	Control limit exceeded on u chart
LCLU	Lower control limit for number of nonconformities per unit
LIMITN	Nominal sample size associated with the control limits
SIGMAS	Multiple (k) of the standard error associated with the control limits
SUBU	Subgroup number of nonconformities per unit
SUBN	Subgroup sample size

Table 32.45 *continued*

Variable	Description
TESTS	Tests for special causes signaled on u chart
_U_	Average number of nonconformities per unit
UCLU	Upper control limit for number of nonconformities per unit
XCHART Statement	
EXLIM	Control limit exceeded on $\bar{X}$ chart
LCLX	Lower control limit for mean
LIMITN	Nominal sample size associated with the control limits
MEAN	Process mean
SIGMAS	Multiple (k) of the standard error associated with the control limits
STDDEV	Process standard deviation ($\hat{\sigma}$ or σ_0)
SUBN	Subgroup sample size
SUBX	Subgroup mean
TESTS	Tests for special causes signaled on $\bar{X}$ chart
UCLX	Upper control limit for mean
XRCHART Statement	
EXLIM	Control limit exceeded on $\bar{X}$ chart
EXLIMR	Control limit exceeded on R chart
LCLR	Lower control limit for range
LCLX	Lower control limit for mean
LIMITN	Nominal sample size associated with the control limits
MEAN	Process mean
_R_	Average range
SIGMAS	Multiple (k) of the standard error associated with the control limits
STDDEV	Process standard deviation ($\hat{\sigma}$ or σ_0)
SUBN	Subgroup sample size
SUBR	Subgroup range
SUBX	Subgroup mean
TESTS	Tests for special causes signaled on $\bar{X}$ chart
TESTS2	Tests for special causes signaled on R chart
UCLR	Upper control limit for range
UCLX	Upper control limit for mean
XSCHART Statement	
EXLIM	Control limit exceeded on $\bar{X}$ chart
EXLIMS	Control limit exceeded on s chart
LCLS	Lower control limit for standard deviation
LCLX	Lower control limit for mean
LIMITN	Nominal sample size associated with the control limits
MEAN	Process mean
_S_	Average standard deviation
SIGMAS	Multiple (k) of the standard error associated with the control limits
STDDEV	Process standard deviation ($\hat{\sigma}$ or σ_0)

Table 32.45 *continued*

Variable	Description
SUBN	Subgroup sample size
SUBS	Subgroup standard deviation
SUBX	Subgroup mean
TESTS	Tests for special causes signaled on $\bar{X}$ chart
TESTS2	Tests for special causes signaled on s chart
UCLS	Upper control limit for standard deviation
UCLX	Upper control limit for mean

Notes:

1. The variable _TESTS_ is saved if you specify the TESTS= option. The kth character of a value of _TESTS_ is k if Test k is positive at that subgroup. For example, if you request all eight tests and Tests 2 and 8 are positive for a given subgroup, the value of _TESTS_ has a 2 for the second character, an 8 for the eighth character, and blanks for the other six characters.
2. The variable _TESTS2_ is saved if you specify the TESTS2= option.
3. The variables _EXLIM_, _EXLIMR_, _EXLIMS_, _TESTS_, and _TESTS2_ are character variables of length 8. All other variables are numeric.

ODS Table Names

Each table that the SPC procedure creates has a name associated with it. You must use this name to refer to the table when you use ODS statements. The name of this table and a short description of its contents are listed in Table 32.46.

Table 32.46 ODS Table Produced by PROC SPC

Table Name	Description	Statements	Option
BoxchartExceptionSummary	Summary of exceptions	BOXCHART	Default output
CapabilityIndices	Specification limits and capability indices	BOXCHART, IRCHART, MCHART, MRCHART, RCHART, SCHART, XCHART, XRCHART, XSCHART	SPECS=
CChartExceptionSummary	Summary of exceptions	CCHART	Default output
EWMAChartExceptionSummary	Summary of exceptions	EWMACHART	Default output
IRChartExceptionSummary	Summary of exceptions	IRCHART	Default output
MAChartExceptionSummary	Summary of exceptions	MACHART	Default output
MChartExceptionSummary	Summary of exceptions	MCHART	Default output
MRChartExceptionSummary	Summary of exceptions	MRCHART	Default output

Table 32.46 *continued*

Table Name	Description	Statement	Option
NPChartExceptionSummary	Summary of exceptions	NPCHART	Default output
PChartExceptionSummary	Summary of exceptions	PCHART	Default output
RChartExceptionSummary	Summary of exceptions	RCHART	Default output
SChartExceptionSummary	Summary of exceptions	SCHART	Default output
UChartExceptionSummary	Summary of exceptions	UCHART	Default output
XChartExceptionSummary	Summary of exceptions	XCHART	Default output
XRChartExceptionSummary	Summary of exceptions	XRCHART	Default output
XSChartExceptionSummary	Summary of exceptions	XSCHART	Default output

Examples: SPC Procedure

The examples in this chapter are available in the GitHub repository located at https://github.com/sascommunities/doc-supplement-casstat.

NOTE: Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections "Using CAS Sessions and CAS Engine Librefs" on page 7 and "Loading a SAS Data Set onto a CAS Server" on page 7 in Chapter 2, "Shared Concepts."

Example 32.1: Saving Results in Output Data Tables

(View the complete code for this example (spcex1.sas) in the example repository.)

The following statements perform $\bar{X}$ and R chart analysis on the data table mylib.AllProcesses that is created in the section "Getting Started: SPC Procedure" on page 1648 and save the results in OUTTABLE= and OUTLIMITS= data tables:

```
proc spc data=mylib.AllProcesses;
   xrchart / noprint
             outtable  = mylib.AllTable
             outlimits = mylib.AllLimits;
run;
```

The NOPRINT option suppresses the display of the exception summary table. Output 32.1.1 shows a partial listing of the mylib.AllTable data table.

Output 32.1.1 Partial Listing of the mylib.AllTable Data Table

VAR	subgroupname	subgroup	_SIGMAS_	_LIMITN_	_SUBN_	_LCLX_	_SUBX_	_MEAN_	_UCLX_
Time	Lot	1	3	6	6	7.94314	8.00833	8.00307	8.06299
Time	Lot	2	3	6	6	7.94314	8.02167	8.00307	8.06299
Time	Lot	3	3	6	6	7.94314	7.97833	8.00307	8.06299
Time	Lot	4	3	6	6	7.94314	8.00667	8.00307	8.06299
Time	Lot	5	3	6	6	7.94314	8.01833	8.00307	8.06299
Time	Lot	6	3	6	6	7.94314	8.00667	8.00307	8.06299
Time	Lot	7	3	6	6	7.94314	7.98500	8.00307	8.06299
Time	Lot	8	3	6	6	7.94314	8.03000	8.00307	8.06299
Time	Lot	9	3	6	6	7.94314	8.03000	8.00307	8.06299
Time	Lot	10	3	6	6	7.94314	7.99167	8.00307	8.06299

STDDEV	_EXLIM_	_LCLR_	_SUBR_	_R_	_UCLR_	_EXLIMR_
0.048927	0	0.16	0.124	0.24847		
0.048927	0	0.09	0.124	0.24847		
0.048927	0	0.07	0.124	0.24847		
0.048927	0	0.10	0.124	0.24847		
0.048927	0	0.16	0.124	0.24847		
0.048927	0	0.12	0.124	0.24847		
0.048927	0	0.15	0.124	0.24847		
0.048927	0	0.11	0.124	0.24847		
0.048927	0	0.15	0.124	0.24847		
0.048927	0	0.10	0.124	0.24847		

Output 32.1.2 displays the mylib.AllLimits data table.

Output 32.1.2 mylib.AllLimits Data Table

processname	subgroupname	_SIGMAS_	_LIMITN_	_LCLX_	_MEAN_	_UCLX_	_STDDEV_
Time	Lot	3	6	7.94	8.00	8.06	0.049
Diameter	Batch	3	5	34.98	34.99	35.01	0.010
Partgap	Sample	3	5	241.94	260.00	278.05	13.454
Breakstrength	Sample	3	V	V	59.98	V	2.111
Amount	Batch	3	V	V	11.98	V	0.044
Delay	Day	3	V	V	9.53	V	8.512
KWatts	Day	3	20	3362.10	3497.56	3633.02	201.937
Weight	Lot	3	5	21.49	24.61	27.74	2.330

LCLR	_R_	_UCLR_	_TYPE_
0.000	0.124	0.25	ESTIMATE
0.000	0.022	0.05	ESTIMATE
0.000	31.293	66.17	ESTIMATE
V	V	V	ESTIMATE
V	V	V	ESTIMATE
V	V	V	ESTIMATE
312.779	754.225	1195.67	ESTIMATE
0.000	5.420	11.46	ESTIMATE

Note that limits variables for three of the processes have the special missing value *V*, because the data for those processes have varying subgroup sample sizes.

The mylib.AllLimits data table has the same format as a LIMITS= data table, so it could provide limits for these processes in a later run of PROC SPC.

Example 32.2: Computing Process Capability Indices

(View the complete code for this example (spcex2.sas) in the example repository.)

You can use the SPC procedure to compute process capability indices for stable processes. The following statements create a data table that contains process specification limits for the processes in the data table mylib.AllProcesses:

```
data mylib.AllSpecs;
   length processname $16;
   input processname LSL Target USL;
   datalines;
Amount           11.95   12.00   12.05
Breakstrength    52.0    60.0      .
Delay              .      0      30
Diameter         34.98   35.00   35.01
KWatts         3400    3500    4000
Partgap         250     260     270
Time              7.85    8.0     8.15
Weight           23.0    24.0    25.0
;
```

Not all processes have both lower and upper specification limits. For example, the Breakstrength process variable measures the breaking strength of metal wire and has no upper specification limit.

The following statements read the specification limits and compute capability indices for processes that the $\bar{X}$ and *R* chart analysis indicates are in statistical control:

```
proc spc data=mylib.AllProcesses;
   xrchart / exchart specs=mylib.AllSpecs;
run;
```

The EXCHART option includes only processes whose exceptions are signaled in the exception summary table. The resulting output is shown in Output 32.2.1.

Output 32.2.1 Exception Summary and Capability Indices

The SPC Procedure

Means and Ranges Chart Summary for ALLPROCESSES

			Means		Ranges	
processname	subgroupname	Number of Subgroups	Number > UCL	Number < LCL	Number > UCL	Number < LCL
Delay	Day	11	3	1	2	2
Diameter	Batch	45	0	1	0	0

Output 32.2.1 *continued*

Specification Limits and Capability Indices for ALLPROCESSES									
processname	subgroupname	LSL	Target	USL	Cp	CPL	CPU	Cpk	Cpm
Amount	Batch	11.950000	12.000000	12.050000	0.379318	0.228958	0.529679	0.228958	0.345768
Breakstrength	Sample	52.000000	60.000000	.	.	1.259258	.	1.259258	1.262873
KWatts	Day	3400.000000	3500.000000	4000.000000	0.495204	0.161038	0.829369	0.161038	0.165056
Partgap	Sample	250.000000	260.000000	270.000000	0.247761	0.247640	0.247882	0.247640	0.247761
Time	Lot	7.850000	8.000000	8.150000	1.021941	1.042834	1.001048	1.001048	1.019939
Weight	Lot	23.000000	24.000000	25.000000	0.143037	0.230790	0.055284	0.055284	0.138324

When you use the EXCHART option in combination with the SPECS= option, each process is listed in either the exception summary table or the capability indices table, but not both.

Example 32.3: Reading Preestablished Control Limits

(View the complete code for this example (spcex3.sas) in the example repository.)

You can apply the control limits that are saved in the data table mylib.AllLimits to new data for those processes. The exception summary table in the section "Getting Started: SPC Procedure" on page 1648 shows unusual variation in the processes Delay and Diameter, whereas the other six processes are stable. The data table mylib.StableProcesses contains additional data for the stable processes. The following statements read the control limits that are saved in the mylib.AllLimits data table and apply them to these new data:

```
proc spc data=mylib.StableProcesses limits=mylib.AllLimits;
   xrchart;
run;
```

The resulting exception summary table is shown in Output 32.3.1.

Output 32.3.1 Exception Summary of Additional Process Data

The SPC Procedure

Means and Ranges Chart Summary for STABLEPROCESSES						
			Means		Ranges	
processname	subgroupname	Number of Subgroups	Number > UCL	Number < LCL	Number > UCL	Number < LCL
Amount	Batch	35	0	0	0	0
Breakstrength	Sample	28	0	0	0	0
KWatts	Day	55	0	0	0	0
Partgap	Sample	36	0	0	1	0
Time	Lot	19	0	0	0	0
Weight	Lot	40	0	0	0	0

The analysis of the additional process data shows unusual variation in the Partgap process.

Example 32.4: Applying Tests for Special Causes

(View the complete code for this example (spcex4.sas) in the example repository.)

The following statements produce a data table for 100 simulated process measurements, called Process001 through Process100:

```
data mylib.Random;
   length processname $16 subgroupname $16;
   do i = 1 to 100;
      processname  = 'Process'  || left( put( i, z3. ) );
      subgroupname = 'Subgroup' || left( put( i, z3. ) );
      do subgroup = 1 to 30;
         do j = 1 to 5;
            process = rannor(123);
            output;
         end;
      end;
   end;
   drop i j;
run;
```

The following statements use PROC SPC to do an $\bar{X}$ chart analysis of the simulated data and apply the first through fourth tests for special causes:

```
proc spc data=mylib.Random;
   xchart / exchart
            tests    = 1 to 4
            outtable = mylib.RandomTests;
run;
```

The TESTS= option requests the tests for special causes. Output 32.2 shows the resulting exception summary table.

Output 32.4.1 Results of Tests for Special Causes

The SPC Procedure

Means Chart Summary for RANDOM

processname	subgroupname	Number of Subgroups	Number > UCL	Number < LCL	Test 1 Signals	Test 2 Signals	Test 3 Signals	Test 4 Signals
Process006	Subgroup006	30	0	1	1	0	0	0
Process011	Subgroup011	30	1	0	1	0	1	0
Process015	Subgroup015	30	0	1	1	0	0	0
Process021	Subgroup021	30	1	0	1	0	0	0
Process031	Subgroup031	30	0	0	0	1	0	0
Process032	Subgroup032	30	1	0	1	0	0	0
Process034	Subgroup034	30	0	0	0	0	1	0
Process047	Subgroup047	30	0	0	0	0	2	0
Process048	Subgroup048	30	0	0	0	0	0	1
Process066	Subgroup066	30	0	0	0	0	0	1
Process069	Subgroup069	30	0	0	0	0	1	0
Process074	Subgroup074	30	0	0	0	0	0	1
Process078	Subgroup078	30	0	1	1	0	0	0
Process079	Subgroup079	30	0	0	0	0	1	0
Process080	Subgroup080	30	0	1	1	0	0	0
Process081	Subgroup081	30	0	0	0	0	1	0
Process089	Subgroup089	30	1	0	1	0	0	0
Process092	Subgroup092	30	1	0	1	0	0	0
Process099	Subgroup099	30	0	0	0	0	1	0

Nineteen of the 100 processes are flagged for at least one exception: a subgroup mean outside the control limits or a positive test result. Because these data were simulated from a normal distribution, they represent processes that are in statistical control. Therefore, you should regard all these exceptions as false positives.

The combined chance of a false signal when Tests 1–4 are applied together is less than 1 in 100 (see the section "Interpreting Tests for Special Causes" on page 1698). Therefore, finding 21 positive tests in an analysis of 3,000 total subgroups is not unexpected.

Example 32.5: Displaying Control Charts by Using SAS/QC Procedures

(View the complete code for this example (spcex5.sas) in the example repository.)

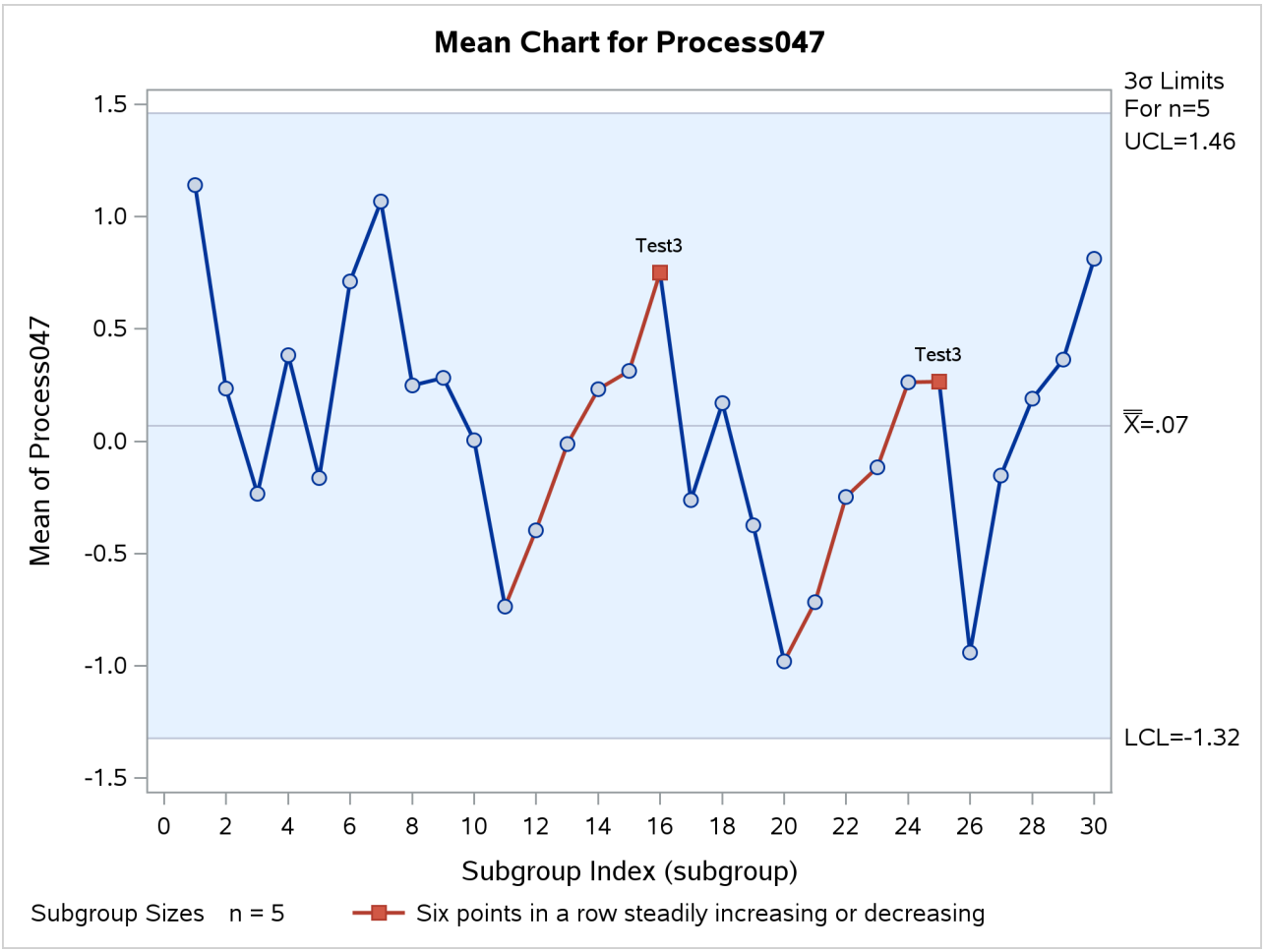
You can use data tables that are produced by the OUTTABLE= option in the SPC procedure's chart statements as input to SAS/QC procedures that can display control charts graphically. You can use PROC SHEWHART to display Shewhart charts that are computed by PROC SPC's BOXCHART, CCHART, IRCHART, MCHART, MRCHART, NPCHART, PCHART, RCHART, SCHART, UCHART, XCHART, XRCHART, and XSCHART statements. You can use PROC MACONTROL to display moving average charts that are computed by PROC SPC's EWMACHART and MACHART statements.

The exception summary table that is displayed in Output 32.5.1 shows that the process variable Process047 had two signals of Test 3, indicating a drift in the process mean. The following statements copy the data for Process047 from the data table mylib.RandomTests to a SAS data set and run PROC SHEWHART to produce an $\bar{X}$ chart of the Process047 data:

```
data RandomTests;
   set mylib.RandomTests;
proc shewhart table=RandomTests(where=(_VAR_ eq 'Process047'));
   xchart Process047 * Subgroup /
            tests = 1 to 4
            markers;
run;
```

The $\bar{X}$ chart for Process047 is shown in Output 32.5.1.

Output 32.5.1 $\bar{X}$ Chart for Process047 Data



The sawtooth pattern of points in Output 32.5.1 indicates an alternating instability in the process. However, it is known that the simulated data are stable, and therefore this must be a random effect. Control chart analysis detects unusual variation in a process measurement, but if you analyze enough data, you will find unusual variation even in a stable process.

References

American Society for Quality Control (1983). *ASQC Glossary and Tables for Statistical Quality Control*. Milwaukee: ASQC.

American Society for Testing and Materials (1976). *ASTM Manual on Presentation of Data and Control Chart Analysis*. Philadelphia: ASTM.

ASQC Automotive Division/AIAG (1990). *Fundamental Statistical Process Control: Reference Manual*. Southfield, MI: Automotive Industry Action Group.

Bissell, A. F. (1990). "How Reliable Is Your Capability Index?" *Journal of the Royal Statistical Society, Series C* 39:331–340.

Boyles, R. A. (1997). "Estimating Common-Cause Sigma in the Presence of Special Causes." *Journal of Quality Technology* 29:381–395.

Burr, I. W. (1969). "Control Charts for Measurements with Varying Sample Sizes." *Journal of Quality Technology* 1:163–167.

Burr, I. W. (1976). *Statistical Quality Control Methods*. New York: Marcel Dekker.

Deming, W. E. (1982). *Out of the Crisis*. Cambridge, MA: Center for Advanced Engineering Study, Massachusetts Institute of Technology.

Grant, E. L., and Leavenworth, R. S. (1988). *Statistical Quality Control*. 6th ed. New York: McGraw-Hill.

Kume, H. (1985). *Statistical Methods for Quality Improvement*. Tokyo: AOTS Chosakai.

Montgomery, D. C. (1996). *Introduction to Statistical Quality Control*. 3rd ed. New York: John Wiley & Sons.

Nelson, L. S. (1984). "The Shewhart Control Chart—Tests for Special Causes." *Journal of Quality Technology* 15:237–239.

Nelson, L. S. (1985). "Interpreting Shewhart $\bar{X}$ Control Charts." *Journal of Quality Technology* 17:114–116.

Nelson, L. S. (1989). "Standardization of Shewhart Control Charts." *Journal of Quality Technology* 21:287–289.

Nelson, L. S. (1994). "Shewhart Control Charts with Unequal Subgroup Sizes." *Journal of Quality Technology* 26:64–67.

Roberts, S. W. (1959). "Control Chart Tests Based on Geometric Moving Averages." *Technometrics* 1:239–250.

Shewhart, W. A. (1931). *Economic Control of Quality Manufactured Product*. New York: D. Van Nostrand; republished in 1980 by the American Society for Quality Control.

Wadsworth, H. M., Stephens, K. S., and Godfrey, A. B. (1986). *Modern Methods for Quality Control and Improvement*. New York: John Wiley & Sons.

Western Electric Company (1956). *Statistical Quality Control Handbook*. Indianapolis: Western Electric Company.

Wheeler, D. J. (1995). *Advanced Topics in Statistical Process Control*. Knoxville, TN: SPC Press.

Chapter 33
SUPERLEARNER Procedure

Contents

Overview: SUPERLEARNER Procedure **1726**
PROC SUPERLEARNER Features 1727
Using CAS Sessions and CAS Engine Librefs 1728
Getting Started: SUPERLEARNER Procedure **1729**
Syntax: SUPERLEARNER Procedure **1731**
PROC SUPERLEARNER Statement 1732
BASELEARNER Statement 1734
BART 1735
BNET 1736
FACTMAC 1738
FOREST 1739
GAMMOD 1740
GAMSELECT 1741
GPCLASS 1743
GPREG 1744
GRADBOOST 1745
LIGHTGRADBOOST 1746
LOGSELECT 1748
NNET 1749
REGSELECT 1751
SVMACHINE 1753
TREESPLIT 1755
CROSSVALIDATION Statement 1756
INPUT Statement 1756
MARGIN Statement 1757
OUTPUT Statement 1757
STORE Statement 1758
TARGET Statement 1758
Details: SUPERLEARNER Procedure **1759**
Super Learner Algorithm 1759
Meta-learning Methods 1760
K-Fold Cross-Validation 1761
Cross-Validated Risk 1762
Supported Base Learner Model Types 1762
Supported Model Selection Suboptions 1762
Missing Values 1766

Displayed Output 1766
Model Information 1766
Number of Observations 1766
Super Learner Coefficients 1766
Scoring Information 1766
Margins 1767
OutputCasTables Table 1767
ODS Table Names 1767
Examples: SUPERLEARNER Procedure **1768**
Example 33.1: Storing and Scoring 1768
Example 33.2: Causal Effect Estimation 1773
References **1778**

Overview: SUPERLEARNER Procedure

The SUPERLEARNER procedure implements the super learner model in SAS Viya (Van der Laan and Rose 2011; Phillips et al. 2023; Naimi and Balzer 2018). In the terminology of machine learning, super learner models are predictive models. The super learner model constructs a predictive model by combining predictions from other models. You can use PROC SUPERLEARNER to train and score predictive models for continuous and binary response variables and to compute predictive margins.

There is no single best machine learning model for all data sets and prediction problems. Given a specific prediction task, it is often difficult to choose a predictive model a priori that will perform best. Wolpert (1992) proposed an approach to increase prediction accuracy by combining multiple models into a single model. Later, this approach became known as "stacking." Breiman (1996) demonstrated additional improvements to predictive performance by using nonnegative least squares for stacking. Van der Laan and colleagues proved the theoretical optimality of stacking in large samples and coined an additional name, "super learner" (Van der Laan and Dudoit 2003; Dudoit and van der Laan 2005; Van der Vaart, Dudoit, and van der Laan 2006; Van der Laan, Polley, and Hubbard 2007). Van der Laan and Rose (2011) provided a detailed description of the super learner model and conducted simulation studies to evaluate its finite sample performance. Phillips et al. (2023) provided practical recommendations for specifying super learner models. The super learner model considers a collection of candidate models, or base learners, and systematically chooses the best combination of them. Optimality theory shows that in large samples, the super learner model performs at least as well as the best base learner model in the ensemble, where predictive performance is defined in terms of a bounded loss function (Van der Laan, Polley, and Hubbard 2007).

When you train a super learner model, you must specify a list of predictive models (also called base learners), a cross-validation scheme, and a meta-learner model. The base learners that make up the ensemble can range from parametric regressions to nonparametric machine learning algorithms. The super learner model uses cross-validation, a sample splitting technique, to assess each base learner's performance. This is an iterative process in which each base learner is trained on a portion of the input data and predicted responses are generated for observations that were not used in the training process. The meta-learner model then computes the best weighted combination of base learners. The output of the training process is an ensemble predictive model that uses a vector of input variables (also called predictor variables) to predict the value of a response variable (also called a target variable). The purpose of a predictive model is to use a set of input variables

to predict a target variable. You create a predictive model by using training data in which the target values are known. If the model is able to accurately predict the value of the target for observations that were not included in the training data, then the model is said to generalize well. Good generalization is an important property of a predictive model. Internally, the super learner model uses cross-validation to mitigate overfitting of the training data and obtain good generalization. The internal cross-validation estimates the out-of-sample performance of each base learner. Base learners that demonstrate better out-of-sample performance tend to receive larger coefficients and generally contribute more to the final predictions.

For more information about super learner models, see the section "Details: SUPERLEARNER Procedure" on page 1759.

PROC SUPERLEARNER Features

PROC SUPERLEARNER has the following basic features:

- trains super learner models for continuous response variables and binary response variables
- specifies and customizes base learner models
- controls the details of cross-validation
- creates an item store for the model and saves it as a binary object in a data table
- produces a data table that contains predicted values
- supports meta-learning methods such as convex-constrained least squares and binomial likelihood maximization
- supports fitting a cross-validated selector (discrete super learner model)

Because PROC SUPERLEARNER runs on SAS Cloud Analytic Services (CAS), it also does the following:

- enables you to run on a cluster of machines that distribute the data and the computations
- enables you to run in single-machine mode
- exploits available cores and concurrent threads

Using CAS Sessions and CAS Engine Librefs

SAS Cloud Analytic Services (CAS) is the analytic server and associated cloud services in SAS Viya. This section describes how to create a CAS session and set up a CAS engine libref that you can use to connect to the CAS session. It assumes that you have a CAS server already available; contact your system administrator if you need help starting and terminating a server. This CAS server is identified by specifying the host on which it runs and the port on which it listens for communications. To simplify your interactions with this CAS server, the host information and port information for the server are stored as SAS option values that are retrieved automatically whenever this CAS server needs to be accessed. You can examine the host and port values for the server at your site by using the following statements:

```
proc options option=(CASHOST CASPORT);
run;
```

In addition to starting a CAS server, your system administrator might also have created a CAS session and a CAS engine libref for your use. You can define your own sessions and CAS engine librefs that connect to the CAS server as shown in the following statements:

```
cas mysess;
libname mylib cas sessref=mysess;
```

The CAS statement creates the CAS session named mysess, and the LIBNAME statement creates the mylib CAS engine libref that you use to connect to this session. It is not necessary to explicitly name the CASHOST and CASPORT of the CAS server in the CAS statement, because these values are retrieved from the corresponding SAS option values.

If you have created the mysess session, you can terminate it by using the TERMINATE option in the CAS statement as follows:

```
cas mysess terminate;
```

For more information about the CAS and LIBNAME statements, see the section “Introduction to Shared Concepts” on page 6 in Chapter 2, “Shared Concepts.”

Getting Started: SUPERLEARNER Procedure

(View the complete code for this example at https://github.com/sascommunities/doc-supplement-casstat/tree/main/Examples/superlearnergs.sas.)

NOTE: Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections "Using CAS Sessions and CAS Engine Librefs" on page 7 and "Loading a SAS Data Set onto a CAS Server" on page 7 in Chapter 2, "Shared Concepts."

This example demonstrates how you can use the SUPERLEARNER procedure to train a super learner model.

The Sashelp.Baseball data table contains salary and performance information for Major League Baseball players (excluding pitchers) who played at least one game in both the 1986 and 1987 seasons (Time Inc. 1987). You can load the Sashelp.Baseball data set into your CAS session by naming your CAS engine libref in the first statement of the following DATA step:

```
data mylib.baseball;
  set sashelp.baseball;
run;
```

The following statements train a super learner model:

```
proc superlearner data=mylib.baseball seed=24893;
   target logSalary / level=interval;
   input nAtBat nHits nHome nRuns nRBI nBB
         yrMajor crAtBat crHits crHome
         crRuns crRbi crBB nOuts nAssts nError
         / level=interval;
   input league division / level=nominal;
   baselearner 'lm' regselect;
   baselearner 'dtree' treesplit;
   baselearner 'bart' bart;
   crossvalidation kfold=6;
   output out=mylib.predout copyvar=logSalary;
run;
```

When you use PROC SUPERLEARNER for training, you must specify the TARGET statement, at least one INPUT statement, and at least two BASELEARNER statements.

You use the TARGET statement to specify the target variable. You use the LEVEL= option to specify that the target variable is treated as a continuous (interval) variable for the analysis. You specify two INPUT statements in this example; one statement contains continuous predictors, and the other statement contains categorical predictors. You specify whether the variables in each INPUT statement are treated as interval (continuous) or nominal (categorical) variables by using the LEVEL= option.

When you use the BASELEARNER statement to specify a base learner model, you begin by specifying a name of the base learner in single quotes. After the name, you must specify a model type. The BASELEARNER statement supports various model types that you can use to build a base learner. It also supports a large number of options that you can use to customize each base learner. For more information, see the BASELEARNER statement.

In this example, you specify three base learners by using three BASELEARNER statements. You use the REGSELECT, TREESPLIT, and BART model types to specify a linear regression model, a decision tree, and a Bayesian additive regression trees model, respectively.

You split the input data into five folds for cross-validation by specifying the KFOLD= option in the CROSSVALIDATION statement. If you omit this statement, cross-validation is still conducted, and the number of folds is computed on the basis of the size of the input data set. For more information, see the section "*K*-Fold Cross-Validation" on page 1761.

To form the final predictive model, PROC SUPERLEARNER uses a meta-learning step to form a weighted combination of the predictions from the base learners. In this example, the target variable is continuous, so the meta-learning step uses the convex-constrained least squares method by default. For more information, see the section "Meta-learning Methods" on page 1760.

Finally, you use the OUTPUT statement to create an output data table named predout that contains predicted response values from the super learner. You use the OUT= option to specify the name of the output data table. You use the COPYVAR= option to copy the target variable from the input data table to the output data table.

The output from this analysis is presented in Figure 33.1 and Figure 33.2. The "Model Information" table in Figure 33.1 summarizes important information about the input data and primary analysis options. Information includes the input data source, response (target) variable and its type, number of folds used in cross-validation, number of base learners, loss function, meta-learning method, and random number seed.

The second table in Figure 33.1 shows the number of observations in the input data table. PROC SUPERLEARNER uses only complete cases in the input data for training, so an observation is excluded from training if any analysis variable contains a missing value. For more information, see the section "Missing Values" on page 1766.

Figure 33.1 Model Information and Number of Observations

Model Information	
Data Source	BASEBALL
Target Variable	logSalary
Target Variable Type	INTERVAL
Number of Folds	6
Number of Base Learners	3
Loss Function	Quadratic Loss
Meta-learning Method	Convex-constrained Least Squares
Random Number Seed	24893

Number of Observations Read	322
Number of Observations Used for Training	263
Number of Observations Used for Scoring	322

Figure 33.2 displays information about the super learner model coefficients, which describe the contribution of each base learner model in the super learner model ensemble. The cross-validated risk measures each base learner's predictive performance in cross-validation. (For more information about cross-validated risk, see the section "Cross-Validated Risk" on page 1762.) In this example, the "bart" base learner model has the smallest cross-validated risk and therefore shows the best out-of-sample performance. It receives the largest super learner model coefficient.

Figure 33.2 Super Learner Model Coefficients

Super Learner Model Coefficients

Name	Model Type	Coefficient	Cross-Validated Risk
lm	Ordinary Linear Least Squares	0.06908	0.35223
dtree	Decision Tree	0.12885	0.26394
bart	Bayesian Additive Regression Trees	0.80207	0.17854

The following PROC PRINT statements print the first 10 observations of the predicted and observed response values. The results are shown in Figure 33.3.

```
proc print data=mylib.predout(obs=10);
run;
```

Figure 33.3 First 10 Observations in the Output Data Table

Obs	P_logSalary	logSalary
1	6.76028	6.65286
2	5.72326	5.70378
3	5.99384	.
4	6.06349	5.76832
5	4.59300	4.70048
6	6.72476	.
7	5.95649	5.95324
8	6.62632	6.74524
9	4.92826	4.38203
10	4.34254	4.31749

Syntax: SUPERLEARNER Procedure

PROC SUPERLEARNER < *options* > **;**
BASELEARNER '*name*' *model-type* **<(***model-options***)>** < *input-options* > **;**
CROSSVALIDATION < *options* > **;**
INPUT *variables* < */option* > **;**
MARGIN '*name*' *variable*='*value*' < . . . *variable*='*value*' > **;**
OUTPUT OUT=*libref.data-table* < *options* > **;**
STORE < **OUT=** >*libref.data-table* **;**
TARGET *variable* < */option* > **;**

The PROC SUPERLEARNER statement, INPUT statement, TARGET statement, and at least two BASELEARNER statements are required. You can specify the BASELEARNER, INPUT, and MARGIN statements multiple times. You can specify all other statements only once. The following sections describe the PROC SUPERLEARNER statement and then describe the other statements in alphabetical order.

PROC SUPERLEARNER Statement

PROC SUPERLEARNER < *options* > **;**

The PROC SUPERLEARNER statement invokes the procedure. Table 33.1 summarizes the available options in this statement.

Table 33.1 PROC SUPERLEARNER Statement Options

Option	Description
APPLYROWORDER	Uses group and order information from the input data table
DATA=	Specifies the input data table
METHOD=	Specifies the method to use for fitting the meta-learner model
RESTORE=	Specifies the input item store
SEED=	Specifies the seed for the pseudorandom number generator

You can specify the following *options*:

APPLYROWORDER
uses group and order information from the input data table. You can add this information to the input table by using the `partition` action in the `table` action set. For more information, see the section "The APPLYROWORDER Option" on page 107 in Chapter 2, "Shared Concepts." This option is applicable to models that require the simulation of latent variables; by default, it is not enabled. If you specify this option but the table does not contain group and order information, the procedure terminates with an error.

DATA=*libref.data-table*
names the input data table for PROC SUPERLEARNER to use. The default is the most recently created data table. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to the data, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 1728.

data-table specifies the name of the input data table.

METHOD=*method* **<(***method-option***)>**
specifies the meta-learning method to be used to estimate the coefficients of the super learner model. By specifying a meta-learning method, you specify an algorithm that is used to combine the individual base learner models in the library. For more information, see the section "Meta-learning Methods" on page 1760. You can specify one of the following *methods*:

CCAUC estimates the coefficients by maximizing the convex-constrained area under the ROC curve (AUC). This option is available only if the response variable is binary.

CCLOGLIK estimates the coefficients by maximizing the convex-constrained binomial log likelihood. This option is available only if the response variable is binary.

CCLS estimates the coefficients by fitting a convex-constrained least squares regression with no intercept.

CVSELECTOR estimates the coefficients by assigning a coefficient of 1 to a single base learner model on the basis of cross-validated performance and setting coefficients of all other base learner models to 0.

By default, METHOD=CCLS if the response variable is continuous and METHOD=CCLOGLIK if the response variable is binary.

You can specify the following *method-option* when METHOD=CVSELECTOR:

LOSS=*keyword*
specifies the loss function to use for computing cross-validated risks. You can specify the following *keywords*:

AUC uses the area under the ROC curve loss function. This option is available only if the response variable is binary.

LOG uses the log loss function. This option is available only if the response variable is binary.

QUADRATIC uses the quadratic loss function.

By default, LOSS=QUADRATIC if the response variable is continuous and LOSS=LOG if the response variable is binary.

RESTORE=*libref.data-table*
specifies the name of the item store that contains a model that is fitted and stored from a previous analysis. The item store is created by a STORE statement from a previous PROC SUPERLEARNER call or by a **`store`** parameter that is specified in a previous **`slTrain`** action call. *libref.data-table* is a two-level name, where *libref* refers to the library, and *data-table* specifies the name of the input data table. For more information about this two-level name, see the section "Using CAS Sessions and CAS Engine Librefs" on page 1728.

You can use the previously fitted model to either score data or compute predictions at fixed values of the predictors. For each task, you use the DATA= option to specify the input data table to be used. To score the input data, you specify the OUTPUT statement to create a data table that contains the observationwise predictions. To compute predictions at fixed values of the predictors, you specify one or more MARGIN statements and specify the MARGINPRED option in the OUTPUT statement. You can also request observationwise predictions from each base learner model that you use to train the super learner model. To do so, specify the LEARNERPRED option in the OUTPUT statement.

Because the previously fitted model is stored and retrieved for the analysis, you cannot include statements and options that are specific to training a model. In particular, the BASELEARNER, CROSSVALIDATION, INPUT, STORE, and TARGET statements are not available when you specify the RESTORE= option.

SEED=*n*

specifies the initial seed to start the pseudorandom number generator for *k*-fold partitioning and base learner model building, where *n* is an integer. If you omit the SEED= option or if *n* is negative or 0, the procedure uses the time of day from the computer's clock to obtain the initial seed.

BASELEARNER Statement

BASELEARNER '*name*' *model-type* **<(***model-options***)>** < *input-options* > **;**

The BASELEARNER statement provides options for you to specify and customize a base learner model. Here *name* represents a label that you assign to the base learner model. It is enclosed in quotes and must be a valid variable name. Every base learner that you specify in a BASELEARNER statement must have a unique label. The labels for base learners are not case-sensitive.

You use the *model-type* to specify the type of a base learner model. You can specify the following *model-types*:

BART specifies a Bayesian additive regression trees model. For a list of available *model-options* and *input-options*, see the BART section.

BNET specifies a Bayesian network model. This model is available only if you specify a binary response variable. For a list of available *model-options* and *input-options*, see the BNET section.

FACTMAC specifies a factorization machine model. This model is available only if you specify a continuous response variable. For a list of available *model-options* and *input-options*, see the FACTMAC section.

FOREST specifies a forest model. For a list of available *model-options* and *input-options*, see the FOREST section.

GAMMOD specifies a generalized additive model based on low-rank regression splines. For a list of available *model-options* and *input-options*, see the GAMMOD section.

GAMSELECT specifies a generalized additive model with model selection. For a list of available *model-options* and *input-options*, see the GAMSELECT section.

GPCLASS specifies a Gaussian process classification model. This model is available only if you specify a binary response variable. For a list of available *model-options* and *input-options*, see the GPCLASS section.

GPREG specifies a Gaussian process regression model. This model is available only if you specify a continuous response variable. For a list of available *model-options* and *input-options*, see the GPREG section.

GRADBOOST specifies a gradient boosting model. For a list of available *model-options* and *input-options*, see the GRADBOOST section.

LIGHTGRADBOOST specifies a light gradient boosting machine model. For a list of available *model-options* and *input-options*, see the LIGHTGRADBOOST section.

LOGSELECT specifies a logistic regression model with model selection. This model is available only if you specify a binary response variable. For a list of available *model-options* and *input-options*, see the LOGSELECT section.

NNET specifies an artificial neural network model. For a list of available *model-options* and *input-options*, see the NNET section.

REGSELECT specifies an ordinary least squares regression model with model selection. This model is available only if you specify a continuous response variable. For a list of available *model-options* and *input-options*, see the REGSELECT section.

SVMACHINE specifies a support vector machine model. For a list of available *model-options* and *input-options*, see the SVMACHINE section.

TREESPLIT specifies a tree-based statistical model. For a list of available *model-options* and *input-options*, see the TREESPLIT section.

Depending on the SAS software product that you have, some of these model types might not be available. For more information, see the section "Supported Base Learner Model Types" on page 1762.

Depending on which model type you choose, different *model-options* and *input-options* might be available to you. The following sections describe the relevant *model-options* and *input-options* for each base learner model.

BART

The BART model fits a Bayesian additive regression trees model. For more information, see Chapter 4, "BART Procedure."

The *model-options* for the BART model cover only a subset of the options in PROC BART. You can specify any of the following BART-specific *model-options* by enclosing them in parentheses after the BART keyword:

MINLEAFSIZE=*number*
specifies the minimum number of observations that each child of a split must contain in the training data in order for the split to be considered. By default, MINLEAFSIZE=5.

NBI=*number*
specifies the number of burn-in iterations to perform before the procedure starts to save samples for prediction. By default, NBI=100.

NBINS=*number*
specifies the number of bins to use for binning the continuous input variables. By default, NBINS=50.

NMC=*number*
specifies the number of iterations in the main simulation loop. By default, NMC=200.

NTHIN=*number*
specifies the thinning rate of the simulation. By default, NTHIN=1.

NTREE=*number*
specifies the number of trees in a sample of the sum-of-trees ensemble. By default, NTREE=50.

For full descriptions of these *model-options* and the BART model, see the PROC BART documentation. Note that the default values for the NMC= and NTREE= options differ from those of the corresponding options in PROC BART.

You can specify *input-options* to customize the set of predictor variables for the BART model type. If you omit *input-options* for this base learner model, then the variables that you specify in the INPUT statements

are used as predictor variables for this base learner. If you specify at least one *input-option*, then the variables that you specify in the INPUT statements do not apply to this base learner. You can specify the following *input-options*:

INTINPUT=(*variables***)**
specifies the interval input *variables* for the base learner model.

NOMINPUT=(*variables***)**
specifies the nominal input *variables* for the base learner model.

If the response variable is nominal, then a binary response model is used. If the response variable is interval, then a Gaussian response model is used.

BNET

The BNET model creates a Bayesian network model. For more information, see Chapter 5, "BNET Procedure" (*SAS Viya: Machine Learning Procedures*).

The *model-options* for the BNET model cover only a subset of the options in PROC BNET. You can specify any of the following BNET-specific *model-options* by enclosing them in parentheses after the BNET keyword:

ALPHA=*number*
specifies the significance level for independence tests by using the chi-square or G-square statistics, where *number* is between 0 and 1, inclusive. By default, ALPHA=0.05.

INDEPTEST=*keyword*
specifies the method to use for independence tests. You can specify the following *keywords*:

ALL	uses the chi-square statistics, the G-square statistics, and the normalized mutual information for independence tests.
CHIGSQUARE	uses both the chi-square and G-square statistics for independence tests.
CHISQUARE	uses the chi-square statistics for independence tests.
GSQUARE	uses the G-square statistics for independence tests.
MI	uses the normalized mutual information for independence tests.

By default, INDEPTEST=CHIGSQUARE.

MAXPARENTS=*integer*
specifies the maximum number of parents that are allowed for each node in the network structure, where *integer* is a value between 1 and 16, inclusive. By default, MAXPARENTS=5.

MIALPHA=*number*
specifies the threshold for independence tests by using mutual information, where *number* is a value between 0 and 1, inclusive. By default, MIALPHA=0.05.

NUMBIN=*integer*
specifies the number of binning levels for all interval variables, where *integer* is a value between 2 and 1,024, inclusive. By default, NUMBIN=5.

PARENTING=*keyword*
specifies the algorithm to use for orienting the network structure. You can specify the following *keywords*:

BESTONE uses a greedy approach to determine the parents of each node.

BESTSET determines the best set of variables among possible candidate sets as the parents of each node.

By default, PARENTING=BESTSET.

PRESCREENING=*keyword*
specifies the initial screening for the input variables. You can specify the following *keywords*:

0 uses all the input variables.

1 uses only the input variables that are dependent on the target.

By default, PRESCREENING=1.

STRUCTURE=*keyword*
specifies the network structure. You can specify the following *keywords*:

GENERAL learns a general Bayesian network over the target and input variables.

MB learns the Markov blanket of the target variable.

NAIVE assumes a naive Bayesian network structure.

PC learns the parent-child Bayesian network structure.

TAN learns the tree-augmented naive Bayesian network structure.

By default, STRUCTURE=PC.

VARSELECT=*keyword*
specifies how input variables are selected beyond the prescreening. You can specify the following *keywords*:

0 uses all input variables that remain after the initial screening is performed as specified in the PRESCREENING= option.

1 tests each input variable for conditional independence of the target variable, given any other input variable.

2 tests each input variable further for conditional independence of the target variable, given any subset of other input variables.

3 determines the Markov blanket of the target variable and uses only the variables in the Markov blanket.

By default, VARSELECT=1.

For full descriptions of these *model-options* and the BNET model, see the PROC BNET documentation. Note that in contrast with PROC BNET, you can specify a single number only when you specify the ALPHA= or MAXPARENTS= option. Also in contrast with PROC BNET, you can specify a single keyword only when you specify the PARENTING=, PRESCREENING=, STRUCTURE=, or VARSELECT= option.

You can specify *input-options* to customize the set of predictor variables for the BNET model type. If you omit *input-options* for this base learner model, then the variables that you specify in the INPUT statements are used as predictor variables for this base learner. If you specify at least one *input-option*, then the variables that you specify in the INPUT statements do not apply to this base learner. You can specify the following *input-options*:

INTINPUT=(*variables***)**
specifies the interval input *variables* for the base learner model.

NOMINPUT=(*variables***)**
specifies the nominal input *variables* for the base learner model.

FACTMAC

The FACTMAC model fits a factorization machine model. For more information, see Chapter 8, "FACTMAC Procedure" (*SAS Viya: Machine Learning Procedures*).

The *model-options* for the FACTMAC model cover only a subset of the options in PROC FACTMAC. You can specify any of the following FACTMAC-specific *model-options* by enclosing them in parentheses after the FACTMAC keyword:

LEARNSTEP=*number*
specifies the learning step size for the stochastic gradient descent algorithm, where *number* is a positive real number. By default, LEARNSTEP=0.001.

MAXITER=*number*
specifies the maximum number of iterations for the algorithm to perform, where *number* is an integer greater than or equal to 1. By default, MAXITER=1.

NFACTORS=*number*
specifies the number of factors to estimate for the model, where *number* is an integer greater than or equal to 1. By default, NFACTORS=1.

For full descriptions of these *model-options* and the FACTMAC model, see the PROC FACTMAC documentation. Note that you must specify at least two nominal input variables. You can also specify any number of interval input variables.

You can specify *input-options* to customize the set of predictor variables for the FACTMAC model type. If you omit *input-options* for this base learner model, then the variables that you specify in the INPUT statements are used as predictor variables for this base learner. If you specify at least one *input-option*, then the variables that you specify in the INPUT statements do not apply to this base learner. You can specify the following *input-options*:

INTINPUT=(*variables***)**
specifies the interval input *variables* for the base learner model.

NOMINPUT=(*variables***)**
specifies the nominal input *variables* for the base learner model.

FOREST

The FOREST model creates a predictive model called a forest (which consists of several decision trees). For more information, see Chapter 11, "FOREST Procedure" (*SAS Viya: Machine Learning Procedures*).

The *model-options* for the FOREST model cover only a subset of the options in PROC FOREST. You can specify any of the following FOREST-specific *model-options* by enclosing them in parentheses after the FOREST keyword:

INBAGFRACTION=*number*
specifies the fraction of the random bootstrap sample of the training data to be used to grow each tree in the forest, where *number* is a value between 0 and 1, inclusive. By default, INBAGFRACTION=0.6.

MAXDEPTH=*number*
specifies the maximum depth of the tree to be grown. By default, MAXDEPTH=20.

MINLEAFSIZE=*number*
specifies the minimum number of observations that each child of a split must contain in the training data table in order for the split to be considered. By default, MINLEAFSIZE=5.

NTREES=*number*
specifies the number of trees to grow in the forest model. By default, NTREES=100.

NUMBIN=*number*
specifies the number of bins to use for binning the interval input variables. By default, NUMBIN=50.

VARS_TO_TRY=*m*
specifies the number of input variables to consider splitting on in a node, where *m* ranges from 1 to the number of input variables. By default, *m* is the square root of the number of input variables.

For full descriptions of these *model-options* and the FOREST model, see the PROC FOREST documentation.

You can specify *input-options* to customize the set of predictor variables for the FOREST model type. If you omit *input-options* for this base learner model, then the variables that you specify in the INPUT statements are used as predictor variables for this base learner. If you specify at least one *input-option*, then the variables that you specify in the INPUT statements do not apply to this base learner. You can specify the following *input-options*:

INTINPUT=(*variables***)**
specifies the interval input *variables* for the base learner model.

NOMINPUT=(*variables*)
specifies the nominal input *variables* for the base learner model.

GAMMOD

The GAMMOD model fits a generalized additive model that is based on low-rank regression splines. For more information, see Chapter 12, "GAMMOD Procedure."

There are no available *model-options* that you can specify when using a base learner of the GAMMOD model type.

You can specify *input-options* to customize the set of predictor variables for the GAMMOD model type. If you omit *input-options* for this base learner model, then the nominal variables that you specify in the INPUT statements enter the base learner as main parametric effects, and each interval variable that you specify in the INPUT statement enters the base learner as a univariate spline effect. If you specify at least one *input-option*, then the variables that you specify in the INPUT statements do not apply to this base learner. You can specify the following *input-options*:

CLASS=(*variables*)
names the classification variables to be used as explanatory variables in the analysis if they are also used in the PARAM(*effects*) option. You can include the response variable, but this is not required.

PARAM(*effects*)
specifies parametric effects that are constructed from variables in the input data; this option can appear multiple times. For information about constructing the model effects, see the section "Specification and Parameterization of Model Effects" on page 64 in Chapter 2, "Shared Concepts."

SPLINE(*variable*...<*variable*><*/spline-options*>)
specifies nonparametric spline effects that are constructed from variables in the input data. You can specify only continuous variables (not classification variables) in spline effects. You can specify this option any number of times. For information about constructing spline effects, see the section "MODEL Statement" on page 596 in Chapter 12, "GAMMOD Procedure."

You can specify the following *spline-options* after a slash:

DF=*n*
specifies a fixed degrees of freedom. When you specify this option, no smoothing parameter selection is performed on the spline term. If *n* is not an integer, then it is truncated to an integer.

M=*n*
specifies the order of the derivative in the penalty term, where *n* is a positive integer. The default is described in the section

"MODEL Statement" on page 596 in Chapter 12, "GAMMOD Procedure."

For a binary response variable, the response distribution for the GAMMOD model is the binary distribution function. For an interval response variable, the response distribution for the GAMMOD model is the normal distribution function.

For example, you can use the following syntax to specify a base learner by using the GAMMOD model type that includes a mixture of parametric and spline effects:

```
baselearner 'lrnr' GAMMOD class=(z) param(z x1) spline(x2) spline(x3);
```

This is equivalent to the following specification that uses PROC GAMMOD:

```
proc gammod data=mylib.data;
   class z;
   model y=param(z x1) spline(x2) spline(x3);
run;
```

GAMSELECT

The GAMSELECT model fits a generalized additive model and performs model selection. For more information, see Chapter 13, "GAMSELECT Procedure."

You can specify any of the following GAMSELECT-specific *model-options* by enclosing them in parentheses after the GAMSELECT keyword:

PARTBYFRAC(VALIDATE=*fraction***)**
randomly assigns specified proportions of the observations in the input data table to the validation role. You specify the proportions for validation by using the VALIDATE= suboption.

PARTBYVAR=*variable***(VALIDATE='***value***')**
names the variable in the input data table whose values are used to assign the validation role to each observation. You cannot use this variable as an analysis variable for this base learner model. The VALIDATE= suboption specifies the formatted value of this variable that is used to assign observations to the validation role. All observations that are not assigned to the validation role are assigned to the training role.

SELECTION=*method***< (***method-options***) >**
specifies the method to be used to select the model. You can specify the following *methods*. You can also specify *method-options* to apply to the specified method by enclosing them in parentheses after the *method*.

BOOSTING specifies the boosting method.

SHRINKAGE specifies the shrinkage method.

By default, SELECTION=BOOSTING.

Because of the intrinsic difference between the boosting method and the shrinkage method, the two selection methods have two different sets of options that enable you to control the selection process. You can specify the following *method-options* when SELECTION=BOOSTING:

CHOOSE=VALIDATE
specifies the criterion to use in selecting the final model.

You can specify the following value:

VALIDATE specifies the average square error for the validation data.

If you omit the CHOOSE= option, the average square error for the training data is used to select the final model.

MAXITER=*number*
specifies the maximum number of iterations for the boosting method. By default, MAXITER=500.

STEPSIZE=*number*
specifies the step size to use for the boosting algorithm, where *number* is a value between 0 and 1. By default, STEPSIZE=0.1.

You can specify the following *method-options* when SELECTION=SHRINKAGE:

LAMBDA1=*number*
sets a fixed nonnegative value to control the sparsity penalty.

LAMBDA2=*number*
sets a fixed nonnegative value to control the smoothness penalty.

MAXITER=*number*
specifies the maximum number of iterations that generalized additive model fitting can perform by solving reweighted additive models at each iteration. By default, MAXITER=500.

You can specify *input-options* to customize the set of predictor variables for the GAMSELECT model type. If you omit *input-options* for this base learner model, then the nominal variables that you specify in the INPUT statements enter the base learner as main parametric effects, and each interval variable that you specify in the INPUT statement enters the base learner as a univariate spline effect. If you specify at least one *input-option*, then the variables that you specify in the INPUT statements do not apply to this base learner. You can specify the following *input-options*:

CLASS=(*variables***)**
names the classification variables to be used as explanatory variables in the analysis if they are also used in the PARAM(*effects*) option. You can include the response variable, but this is not required.

PARAM(*effects***)**
specifies parametric effects that are constructed from variables in the input data; you can specify this option multiple times. For information about constructing the model effects, see the section "Specification and Parameterization of Model Effects" on page 64 in Chapter 2, "Shared Concepts."

SPLINE(*variable***<** *variable***><** */ *spline-options***>)**
specifies nonparametric spline effects that are constructed from variables in the input data. You can specify only continuous variables (not classification variables) in spline effects. You can specify one or two variables in this option when SELECTION=BOOSTING. You can specify only one variable when SELECTION=SHRINKAGE. You can specify this option any number of times. For information about constructing spline effects, see the section "MODEL Statement" on page 656 in Chapter 13, "GAMSELECT Procedure."

When SELECTION=BOOSTING, you can specify the following *spline-options* after a slash:

DEGREE=*n*
specifies the degree of the spline transformation, where *n* is a nonnegative integer. By default, DEGREE=3.

DF=*n*
specifies the fixed degrees of freedom for the spline at each boosting iteration. By default, DF=4 for univariate spline terms and DF=6 for bivariate spline terms.

For a binary response variable, the response distribution for the GAMSELECT model is the binary distribution function. For an interval response variable, the response distribution for this model is the normal distribution function.

For example, you can use the following syntax to specify a base learner model by using the GAMSELECT model type that includes a mixture of parametric and spline effects with boosting model selection based on partitioned data:

```
baselearner 'lrnr' gamselect(selection=boosting partbyfrac(validate=0.2))
   class=(z) param(z x1) spline(x2) spline(x3);
```

This is equivalent to the following specification that uses PROC GAMSELECT:

```
proc gamselect data=mylib.data;
   partition fraction(validate=0.2);
   selection method=boosting;
   class z;
   model y=param(z x1) spline(x2) spline(x3);
run;
```

GPCLASS

The GPCLASS model fits a Gaussian process classification model. For more information, see Chapter 15, "GPCLASS Procedure" (*SAS Viya: Machine Learning Procedures*).

The *model-options* for the GPCLASS model cover only a subset of the options in PROC GPCLASS. You can specify any of the following GPCLASS-specific *model-options* by enclosing them in parentheses after the GPCLASS keyword:

KERNEL=*keyword***<(***kernel-options***)>**
specifies the kernel-related parameters to use in the Gaussian process. You can specify the following *keywords*:

CHIGSQUARE uses a linear kernel.

GAUSSIAN uses a Gaussian kernel.

By default, KERNEL=GAUSSIAN.

You can also specify the following *kernel-options* by enclosing them in parentheses after the *keyword*:

CONSTANT=*n*
specifies the constant in the linear kernel function. By default, CONSTANT=0.

SIGMA=*n*
specifies the bandwidth in the Gaussian kernel function. By default, SIGMA=1.

MAXITER=*number*
specifies the maximum number of Newton iterations. By default, MAXITER=100.

MINLEAFSIZE=*number*
specifies the minimum change of the loss function between consecutive Newton iterations. By default, THRESHOLD=1.0E–10.

For full descriptions of these *model-options* and the GPCLASS model, see the PROC GPCLASS documentation.

You can specify *input-options* to customize the set of predictor variables for the GPCLASS model type. If you omit *input-options* for this base learner model, then the variables that you specify in the INPUT statements are used as predictor variables for this base learner. If you specify at least one *input-option*, then the variables that you specify in the INPUT statements do not apply to this base learner. You can specify the following *input-option*:

INTINPUT=(*variables***)**
specifies the interval input *variables* for the base learner model.

GPREG

The GPREG model performs Gaussian process regression. For more information, see Chapter 16, "GPREG Procedure" (*SAS Viya: Machine Learning Procedures*).

The *model-options* for the GPREG model cover only a subset of the options in PROC GPREG. You can specify any of the following GPREG-specific *model-options* by enclosing them in parentheses after the GPREG keyword:

ALGORITHM=*keyword*
specifies the optimization-algorithm to use during training. You can specify the following *keywords*:

ADAM specifies the adaptive moments algorithm.

SGD specifies the plain stochastic gradient descent algorithm.

By default, ALGORITHM=ADAM.

AUTORELEVANCEDETERMINATION
uses automatic relevance determination in the kernel function.

JITTERMAXITERS=*number*
specifies the maximum number of iterations for jitter Cholesky decomposition. By default, JITTERMAXITERS=10.

KERNEL=*keyword*
specifies the kernel-related parameters to be used in the Gaussian process. You can specify the following *keywords*:

LINEAR specifies a linear kernel.

MATERN32 specifies a Matern 3/2 kernel.

MATERN52 specifies a Matern 5/2 kernel.

PERIODIC specifies a periodic kernel.

RBF specifies a radial basis function kernel.

By default, KERNEL=RBF.

LEARNINGRATE=*number*
specifies the learning rate parameter for the adam or SGD algorithm. By default, LEARNINGRATE=0.001.

MINIBATCHSIZE=*number*
specifies the size of the minibatches to use in the adam or SGD algorithm. By default, MINIBATCHSIZE=10.

MOMENTUM=*number*
specifies the momentum, where *number* is a value between 0 and 1, inclusive. By default, MOMENTUM=0.

NINDUCINGPOINTS=*number*
specifies the number of inducing points to use in the sparse Gaussian process, where *number* is a positive integer. By default, NINDUCINGPOINTS=100.

For full descriptions of these *model-options* and the GPREG model, see the PROC GPREG documentation.

You can specify *input-options* to customize the set of predictor variables for the GPREG model type. If you omit *input-options* for this base learner model, then the variables that you specify in the INPUT statements are used as predictor variables for this base learner. If you specify at least one *input-option*, then the variables that you specify in the INPUT statements do not apply to this base learner. You can specify the following *input-option*:

INTINPUT=(*variables***)**
specifies the interval input *variables* for the base learner model.

GRADBOOST

The GRADBOOST model specifies a gradient boosting model. For more information, see Chapter 17, "GRADBOOST Procedure" (*SAS Viya: Machine Learning Procedures*).

The *model-options* for the GRADBOOST model cover only a subset of the options in PROC GRADBOOST. You can specify any of the following GRADBOOST-specific *model-options* by enclosing them in parentheses after the GRADBOOST keyword:

LASSO=*number*
specifies the L1 norm regularization parameter, where *number* must be nonnegative. By default, LASSO=0.

LEARNINGRATE=*number*
specifies the learning rate for the gradient boosting algorithm, where *number* is a value between 0 and 1, inclusive. By default, LEARNINGRATE=0.1.

MAXDEPTH=*number*
specifies the maximum depth of the tree to be grown. By default, MAXDEPTH=4.

MINLEAFSIZE=*number*
specifies the minimum number of observations that each child of a split must contain in the training data table in order for the split to be considered. By default, MINLEAFSIZE=5.

NTREES=*number*
specifies the *number* of trees to grow in the gradient boosting model. By default, NTREES=100.

NUMBIN=*number*
specifies the number of bins to use for binning the interval input variables. By default, NUMBIN=50.

RIDGE=*number*
specifies the L2 norm regularization parameter, where *number* must be nonnegative. By default, RIDGE=1.

SAMPLINGRATE=*number*
specifies the fraction of the training data to be used to grow each tree in the boosting model. By default, SAMPLINGRATE=0.5.

VARS_TO_TRY=*m*
specifies the number of input variables to consider splitting on in a node, where *m* ranges from 1 to the number of input variables. By default, *m* is the number of input variables

For full descriptions of these *model-options* and the GRADBOOST model, see the PROC GRADBOOST documentation.

You can specify *input-options* to customize the set of predictor variables for the GRADBOOST model type. If you omit *input-options* for this base learner model, then the variables that you specify in the INPUT statements are used as predictor variables for this base learner. If you specify at least one *input-option*, then the variables that you specify in the INPUT statements do not apply to this base learner. You can specify the following *input-options*:

INTINPUT=(*variables***)**
specifies the interval input *variables* for the base learner model.

NOMINPUT=(*variables***)**
specifies the nominal input *variables* for the base learner model.

LIGHTGRADBOOST

The LIGHTGRADBOOST model fits a light gradient boosting machine model. For more information, see Chapter 20, "LIGHTGRADBOOST Procedure" (*SAS Viya: Machine Learning Procedures*).

The *model-options* for the LIGHTGRADBOOST model cover only a subset of the options in PROC LIGHTGRADBOOST. You can specify any of the following LIGHTGRADBOOST-specific *model-options* by enclosing them in parentheses after the LIGHTGRADBOOST keyword:

BAGGINGFRACTION=*number*
randomly selects a portion of the observations without resampling. By default, BAGGINGFRACTION=1.0.

BAGGINGFREQUENCY=*number*
specifies the frequency of bagging. By default, BAGGINGFREQUENCY=0.

INPUTFRACTION=*number*
randomly selects a subset of features on each iteration or tree. By default, INPUTFRACTION=1.0.

LASSO=*number*
specifies the L1-norm regularization parameter, where *number* must be nonnegative. You cannot use this option for categorical responses. By default, LASSO=0.

LEAFSIZE=*number*
specifies the minimum number of observations that each child of a split must contain in the training data table in order for the split to be considered. By default, LEAFSIZE=20.

LEARNINGRATE=*number*
specifies the learning rate for each tree, where *number* must be greater than 0. By default, LEARNINGRATE=0.1.

MAXDEPTH=*number*
limits the maximum depth of the tree model. A *number* less than or equal to 0 means no limit. By default, MAXDEPTH=–1.

MAXITERS=*number*
specifies the maximum number of iterations for the boosting. By default, MAXITERS=100.

NUMBIN=*number*
specifies the number of bins to use for binning the interval input variables. By default, NUMBIN=255.

RIDGE=*number*
specifies the L2-norm regularization parameter, where *number* must be nonnegative. You cannot use this option for categorical responses. By default, RIDGE=1.

For full descriptions of these *model-options* and the LIGHTGRADBOOST model, see the PROC LIGHTGRADBOOST documentation.

You can specify *input-options* to customize the set of predictor variables for the LIGHTGRADBOOST model type. If you omit *input-options* for this base learner model, then the variables that you specify in the INPUT statements are used as predictor variables for this base learner. If you specify at least one *input-option*, then the variables that you specify in the INPUT statements do not apply to this base learner. You can specify the following *input-options*:

INTINPUT=(*variables***)**
specifies the interval input *variables* for the base learner model.

NOMINPUT=(*variables***)**
specifies the nominal input *variables* for the base learner model.

LOGSELECT

The LOGSELECT model specifies a logistic regression model with model selection. For more information, see Chapter 18, "LOGSELECT Procedure."

The *model-options* for the LOGSELECT model cover only a subset of the options in PROC LOGSELECT. You can specify any of the following LOGSELECT-specific *model-options* by enclosing them in parentheses after the LOGSELECT keyword:

LASSORHO=*r*
specifies the base regularization parameter for the LASSO model selection method. The regularization parameter for step i is r^i. By default, LASSORHO=0.8.

LASSOSTEPS=*n*
specifies the maximum number of steps for LASSO model selection. By default, LASSOSTEPS=20.

LASSOTOL=*r*
specifies the convergence tolerance for the optimization algorithm that solves for the LASSO parameter estimates at each step of LASSO model selection. By default, LASSOTOL=1E–6.

LINK=*keyword*
specifies the link function for the model. The default link is the logit. You can specify the following *keywords*:

LOGIT	specifies the logit link function.
PROBIT	specifies the probit link function.

PARTBYFRAC(VALIDATE=*fraction***)**
randomly assigns specified proportions of the observations in the input data table to the validation role. You specify the proportions for validation by using the VALIDATE= suboption.

PARTBYVAR=*variable***(VALIDATE='***value***')**
names the variable in the input data table whose values are used to assign the validation role to each observation. You cannot use this variable as an analysis variable for this base learner model. The VALIDATE= suboption specifies the formatted value of this variable that is used to assign observations to the validation role. All observations that are not assigned to the validation role are assigned to the training role.

SELECTION=*method***<(***method-options***)>**
specifies the method to be used to select the model. You can specify the following selection *methods*:

BACKWARD	performs backward elimination. This method starts with all effects in the model and deletes effects.
ELASTICNET	performs elastic net selection. This method selects effects by fitting a sequence of candidate models with a grid of regularization parameters.
FORWARD	performs forward selection. This method starts with no effects in the model and adds effects.

LASSO performs model selection by the group LASSO method. This method adds and removes effects by using a sequence of LASSO steps.

NONE specifies no model selection. This method fits the full model.

STEPWISE performs stepwise selection. This method is similar to the FORWARD selection method, except that effects already in the model do not necessarily stay there.

By default, SELECTION=NONE. You can also specify *method-options* to apply to the specified method by enclosing them in parentheses after the method. For more information about the available *method-options*, see the section "Supported Model Selection Suboptions" on page 1762.

You can specify *input-options* to customize the model effects for the LOGSELECT model type. If you omit *input-options* for this base learner model, then the variables that you specify in the INPUT statements are used as main model effects for this base learner. If you specify at least one *input-option*, then the variables that you specify in the INPUT statements do not apply to this base learner. You can specify the following *input-options*:

CLASS=(*variables***)**
names the classification variables to be used as explanatory variables in the analysis if you also specify them in the EFFECT= option. You can include the response variable, but this is not required.

EFFECT=(*effects***)**
specifies model effects that are constructed from variables in the input data table. For information about constructing the model effects, see the section "Specification and Parameterization of Model Effects" on page 64 in Chapter 2, "Shared Concepts."

For example, you can use the following syntax to specify a base learner model by using the LOGSELECT model type with LASSO model selection, where the regularization parameter is chosen on the basis of the partitioned data:

```
baselearner 'lrnr'
   logselect(selection=lasso(choose=validate) partbyfrac(validate=0.2))
   class=(z) effect=(z x1-x3);
```

This is equivalent to the following specification that uses PROC LOGSELECT:

```
proc logselect data=mylib.data;
   partition fraction(validate=0.2);
   selection method=lasso(choose=validate);
   class z;
   model y=z x1-x3;
run;
```

NNET

The NNET model trains an artificial neural network model. For more information, see Chapter 25, "NNET Procedure" (*SAS Viya: Machine Learning Procedures*).

The *model-options* for the NNET model cover only a subset of the options in PROC NNET. You can specify any of the following NNET-specific *model-options* by enclosing them in parentheses after the NNET keyword:

ALGORITHM=*keyword*

specifies the optimization algorithm to use during training. You can specify the following *keywords*:

ADAM	specifies the adaptive moments algorithm.
HF	specifies the Hessian-free algorithm.
LBFGS	specifies the limited-memory Broyden-Fletcher-Goldfarb-Shanno algorithm.
SGD	specifies the plain stochastic gradient descent algorithm.

By default, ALGORITHM=LBFGS.

ANNEALINGRATE=*number*

specifies the annealing parameter, where *number* is a nonnegative value. You can use this option when you specify the adaptive moments algorithm (ALGORITHM=ADAM) or the stochastic gradient descent algorithm (ALGORITHM=SGD). By default, ANNEALINGRATE=1.0E–6.

HIDDEN(*number* **<** */ layer-options* **>)**

specifies the number of neurons or units in a hidden layer. You must specify a *number*, which must be a positive integer. Multiple HIDDEN options are allowed. Each additional HIDDEN option adds a new hidden layer that connects sequentially to the previous layer. If you omit this option, the NNET base learner model has no hidden layer and is equivalent to a generalized linear model.

You can specify the following *layer-options* after a slash:

ACT=*keyword*

specifies the activation function for the neurons on each hidden layer. You can specify the following *keywords*:

EXP	specifies the exponential function.
IDENTITY	specifies the identity function.
LOGISTIC	specifies the logistic function.
RECTIFIER	specifies the rectifier activation function.
SIN	specifies the sine function.
TANH	specifies the hyperbolic tangent function.

By default, ACT=TANH.

COMB=*keyword*

specifies the combination function for the neurons in each hidden layer. You can specify the following *keywords*:

ADD	specifies the additive combination function.
LINEAR	specifies the linear combination function.

By default, COMB=LINEAR.

LEARNINGRATE=*number*
specifies the learning rate parameter, where *number* is a nonnegative value. You can use this option when you specify the adaptive moments algorithm (ALGORITHM=ADAM) or the stochastic gradient descent algorithm (ALGORITHM=SGD). By default, LEARNINGRATE=0.001.

MINIBATCHSIZE=*number*
specifies the size of the minibatches to use in the algorithm, where *number* is a positive value. You can use this option when you specify the adaptive moments algorithm (ALGORITHM=ADAM) or the stochastic gradient descent algorithm (ALGORITHM=SGD). By default, MINIBATCHSIZE=10.

MOMENTUM=*number*
specifies the momentum, where *number* is a value between 0 and 1, inclusive. You can use this option when you specify the adaptive moments algorithm (ALGORITHM=ADAM) or the stochastic gradient descent algorithm (ALGORITHM=SGD). By default, MOMENTUM=0.

REGL1=*number*
specifies the L1 regularization parameter for the model loss function, where *number* is a nonnegative value. By default, REGL1=0.

REGL2=*number*
specifies the L2 regularization parameter, where *number* is a nonnegative value. By default, REGL2=0.

For full descriptions of these *model-options* and the NNET model, see the PROC NNET documentation.

You can specify *input-options* to customize the set of predictor variables for the NNET model type. If you omit *input-options* for this base learner model, then the variables that you specify in the INPUT statements are used as predictor variables for this base learner. If you specify at least one *input-option*, then the variables that you specify in the INPUT statements do not apply to this base learner. You can specify the following *input-options*:

INTINPUT=(*variables***)**
specifies the interval input *variables* for the base learner model.

NOMINPUT=(*variables***)**
specifies the nominal input *variables* for the base learner model.

REGSELECT

The REGSELECT model specifies an ordinary least squares regression model with model selection. For more information, see Chapter 29, "REGSELECT Procedure."

The *model-options* for the REGSELECT model cover only a subset of the options in PROC REGSELECT. You can specify any of the following REGSELECT-specific *model-options* by enclosing them in parentheses after the REGSELECT keyword:

PARTBYFRAC(VALIDATE=*fraction***)**
randomly assigns specified proportions of the observations in the input data table to the validation role. You specify the proportions for validation by using the VALIDATE= suboption.

PARTBYVAR=*variable***(VALIDATE='***value***')**
names the variable in the input data table whose values are used to assign the validation role to each observation. You cannot use this variable as an analysis variable for this base learner model. The VALIDATE= suboption specifies the formatted value of this variable that is used to assign observations to the validation role. All observations that are not assigned to the validation role are assigned to the training role.

SELECTION=*method***< (***method-options***) >**
specifies the method to be used to select the model. You can specify the following *methods*:

BACKWARD performs backward elimination. This method starts with all effects in the model and deletes effects.

ELASTICNET performs elastic net selection. This method selects effects by fitting a sequence of candidate models with a grid of regularization parameters.

FORWARD performs forward selection. This method starts with no effects in the model and adds effects.

FORWARDSWAP specifies forward-swap selection, which is an extension of the forward selection method.

LAR specifies the least angle regression method. Like forward selection, this method starts with no effects in the model and adds effects. The parameter estimates at any step are "shrunk" when compared to the corresponding least squares estimates. If the model contains classification variables, then these classification variables are split.

LASSO specifies the LASSO method, which adds and deletes parameters on the basis of a version of ordinary least squares in which the sum of the absolute regression coefficients is constrained. If the model contains classification variables, then these classification variables are split.

MCP specifies the MCP method, which uses the minimax concave penalty as the penalty function in penalized least squares estimation.

NONE specifies no model selection.

SCAD specifies the SCAD method, which uses smoothly clipped absolute deviation as the penalty function in penalized least squares estimation.

STEPWISE performs stepwise selection. This method is similar to the forward selection method, except that effects already in the model do not necessarily stay there.

By default, SELECTION=NONE. You can also specify *method-options* to apply to the specified method by enclosing them in parentheses after the *method*. For more information about the available *method-options*, see the section "Supported Model Selection Suboptions" on page 1762.

You can specify *input-options* to customize the model effects for the REGSELECT model type. If you omit *input-options* for this base learner model, then the variables that you specify in the INPUT statements are used as main model effects for this base learner. If you specify at least one *input-option*, then the variables that you specify in the INPUT statements do not apply to this base learner. You can specify the following *input-options*:

CLASS=(*variables***)**
names the classification variables to be used as explanatory variables in the analysis if you also specify them in the EFFECT= option. You can include the response variable, but this is not required.

EFFECT=(*effects***)**
specifies model effects that are constructed from variables in the input data table. For information about constructing the model effects, see the section "Specification and Parameterization of Model Effects" on page 64 in Chapter 2, "Shared Concepts."

For example, you can use the following syntax to specify a base learner model by using the REGSELECT model type with LASSO model selection, where the regularization parameter is chosen on the basis of the partitioned data:

```
baselearner 'lrnr'
   regselect(selection=lasso(choose=validate) partbyfrac(validate=0.2))
   class=(z) effect=(z x1-x3);
```

This is equivalent to the following specification that uses PROC REGSELECT:

```
proc regselect data=mylib.data;
   partition fraction(validate=0.2);
   selection method=lasso(choose=validate);
   class z;
   model y=z x1-x3;
run;
```

SVMACHINE

The SVMACHINE model fits a support vector machine model. For more information, see Chapter 39, "SVMACHINE Procedure" (*SAS Viya: Machine Learning Procedures*).

The *model-options* for the SVMACHINE model cover only a subset of the options in PROC SVMACHINE. You can specify any of the following SVMACHINE-specific *model-options* by enclosing them in parentheses after the SVMACHINE keyword:

C=*number*
specifies the penalty value, where *number* is a real number greater than 0. By default, C=1.0.

DEGREE=*number*
specifies the degree that is used in a polynomial kernel. By default, DEGREE=2.

EPSILON=*number*
specifies the insensitive loss value, where *number* is a nonnegative real number. By default, EPSILON=0.01.

KERNEL=*keyword*
specifies the type of kernel. You can specify the following *keywords*:

LINEAR	uses a linear kernel.
POLYNOMIAL	uses a polynomial kernel.
RBF	uses a radial basis function kernel.
SIGMOID	uses a sigmoid kernel.

By default, KERNEL=LINEAR.

METHOD=*keyword*
specifies the optimization method. You can specify the following *keywords*:

ACTIVESET	uses the active-set method.
CD	uses the coordinate descent method.
IPOINT	uses the interior point method.

By default, METHOD=IPOINT.

RBFPARAMETER=*number*
specifies the K_PAR parameter that is used in an RBF kernel, where *number* is greater than or equal to 0.0001. By default, the *number* is the square root of the number of features.

REGL2=*number*
specifies the L2 penalty value when METHOD=CD, where *number* is a real number greater than 0.

SCALE
scales the input variables to between 0 and 1, inclusive, during training when the response variable is continuous. By default, the input variables are not scaled when the response variable is continuous.

SIGMOIDPARAMETER1=*number*
specifies the K_PAR1 parameter that is used in a sigmoid kernel, where *number* is a positive number. By default, SIGMOIDPARAMETER1=1.

SIGMOIDPARAMETER2=*number*
specifies the K_PAR2 parameter that is used in a sigmoid kernel, where *number* is a real number. By default, SIGMOIDPARAMETER2=–1.

For full descriptions of these *model-options* and the SVMACHINE model, see the PROC SVMACHINE documentation.

You can specify *input-options* to customize the set of predictor variables for the SVMACHINE model type. If you omit *input-options* for this base learner model, then the variables that you specify in the INPUT statements are used as predictor variables for this base learner. If you specify at least one *input-option*, then the variables that you specify in the INPUT statements do not apply to this base learner. You can specify the following *input-options*:

INTINPUT=(*variables***)**
specifies the interval input *variables* for the base learner model.

NOMINPUT=(*variables***)**
specifies the nominal input *variables* for the base learner model.

TREESPLIT

The TREESPLIT model fits a decision tree model. For more information, see Chapter 34, "TREESPLIT Procedure."

The *model-options* for the TREESPLIT model cover only a subset of the options in PROC TREESPLIT. You can specify any of the following TREESPLIT-specific *model-options* by enclosing them in parentheses after the TREESPLIT keyword:

CRITERION=*keyword*
specifies the criterion by which to split a parent node into child nodes. You can specify the following *keywords*:

CHAID uses the chi-square automatic interaction detector (CHAID) criterion.

CHISQUARE uses a chi-square statistic to split each variable, and then uses the p-values that correspond to the resulting splits to determine the splitting variable. This *keyword* is available only for categorical responses.

ENTROPY uses the gain in information to split each variable, and then to determine the split. This *keyword* is available only for categorical responses.

GINI uses the decrease in the Gini index to split each variable and then to determine the split. This *keyword* is available only for categorical responses.

IGR uses the entropy metric to split each variable, and then uses the information gain ratio to determine the split. This *keyword* is available only for categorical responses.

FTEST uses an F statistic to split each variable, and then uses the resulting p-value to determine the split variable. This *keyword* is available only for continuous responses.

RSS uses the change in response variance to split each variable and then to determine the split. This *keyword* is available only for continuous responses.

By default, CRITERION=IGR for categorical responses and CRITERION=RSS for continuous responses.

DEGREE=*number*
specifies the maximum depth of the tree to be grown. For the default, see Chapter 34, "TREESPLIT Procedure."

MINLEAFSIZE=*number*
specifies the minimum number of observations per child node. By default, MINLEAFSIZE=5.

NUMBIN=*number*
specifies the number of bins to use for binning interval predictor variables. By default, NUMBIN=50.

For full descriptions of these *model-options* and the TREESPLIT model, see the PROC TREESPLIT documentation. Note that in contrast with PROC TREESPLIT, no pruning is done when you use the TREESPLIT base learner model.

You can specify *input-options* to customize the set of predictor variables for the TREESPLIT model type. If you omit *input-options* for this base learner model, then the variables that you specify in the INPUT statements are used as predictor variables for this base learner. If you specify at least one *input-option*, then the variables that you specify in the INPUT statements do not apply to this base learner. You can specify the following *input-options*:

INTINPUT=(*variables***)**
specifies the interval input *variables* for the base learner model.

NOMINPUT=(*variables***)**
specifies the nominal input *variables* for the base learner model.

CROSSVALIDATION Statement

CROSSVALIDATION < *options* > **;**

The CROSSVALIDATION statement controls details about the *k*-fold cross-validation process. If you omit this statement, PROC SUPERLEARNER uses an internal algorithm to determine the number of folds. For more information about the internal algorithm, see the section "*K*-Fold Cross-Validation" on page 1761.

You can specify the following *options*:

KFOLD=*n*
specifies the number of partition folds in cross-validation, where *n* is an integer between 2 and the number of observations that are used for training, inclusive. If you omit this option, the value of *n* is determined by the process described in the section "*K*-Fold Cross-Validation" on page 1761.

STRATVAR=*variable*
specifies a categorical *variable* for stratified cross-validation. This option is valid only if the number of observations in each class of the *variable* is larger than the number of folds. For more information about stratified cross-validation, see the section "*K*-Fold Cross-Validation" on page 1761.

INPUT Statement

INPUT *variables* < */option* > **;**

The INPUT statement specifies one or more predictor variables for the super learner model that share a common variable type. If your analysis has predictor variables of different types (for example, interval variables and nominal variables), you must specify multiple INPUT statements, one for each type. If you specify multiple INPUT statements, the union of variables that you specify in all the INPUT statements is used as predictor variables for each base learner model. However, if you specify any *input-options* in a BASELEARNER statement, then that base learner no longer uses variables that you specify in the INPUT statements as predictors.

You can specify the following *option* after a slash (/):

LEVEL=*value*
specifies the variable type. You can specify the following *values*:

INTERVAL specifies that the variables are interval, which must be numeric.

NOMINAL specifies that the variables are nominal, also known as classification variables, which can be numeric or character.

By default, LEVEL=INTERVAL for numeric variables and LEVEL=NOMINAL for categorical character variables.

MARGIN Statement

MARGIN '*name*' *variable*='*value*' < . . . *variable*='*value*' > **;**

The MARGIN statement computes a predictive margin (Lane and Nelder 1982). A predictive margin is obtained by fixing the *value* of one or more input *variables* and averaging the predicted response values over the distribution of predictor variables in an input data table.

The MARGIN statement has the following elements:

name
identifies the predictive margin. The *name* is used to name a variable in an output data table that is created by a OUTPUT statement. It must be a valid variable name.

variable=**'***value***'**
identifies an input *variable* and the *value* that it is set to when the predictive margin is computed. The *variable* can be either continuous or categorical. For a categorical variable, you must specify the *value* in quotes. You must specify a raw (not formatted) value. You can specify multiple input *variables* and the *values* that they are set to when computing the predictive margin.

OUTPUT Statement

OUTPUT OUT=*libref.data-table* < *options* > **;**

The OUTPUT statement creates a data table that contains observationwise predicted values that PROC SUPERLEARNER computes after fitting the model. To avoid data duplication when you have large data tables, the variables in the input data table are not included in the output data table unless you specify them in the COPYVARS= option.

You must specify the following option:

OUT=*libref.data-table*
names the output data table for PROC SUPERLEARNER to use. You must specify this option before any other options. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to where the data table is to be stored, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 1728.

data-table specifies the name of the output data table.

You can also specify the following *options*:

COPYVAR=*variable*

COPYVARS=(*variables***)**
transfers one or more *variables* from the input data table to the output data table.

LEARNERPRED
computes predicted response values by using each trained base learner model. The default name of the computed variable in the output data table is the name of the base learner that you specify in the BASELEARNER statement. You can use this option to compare base learner and super learner model predictions.

MARGINPRED
outputs the predicted response values under each intervening scenario that you specify in a MARGIN statement. The default name of the computed variable in the output data table is the name that you specify in the MARGIN statement.

STORE Statement

STORE < **OUT=** >*libref.data-table* **;**

The STORE statement saves the context and results of the statistical analysis. The resulting item store has a binary file format that you cannot modify. *libref.data-table* is a two-level name, where *libref* refers to the library, and *data-table* specifies the name of the output data table. For more information about this two-level name, see the section "Using CAS Sessions and CAS Engine Librefs" on page 1728.

An example of item store technology is to perform a time-consuming analysis and to store its results by using the STORE statement. Later you can perform specific statistical analysis tasks that are based on the saved results of the previous analysis, without having to fit the model again. You can process the contents of the item store by using the RESTORE= option or the **`slScore`** action in the **`superLearner`** action set. Currently, processing the contents of the item store by using the ASTORE procedure or the **`score`** action in the **`aStore`** action set is not supported.

TARGET Statement

TARGET *variable* < */option* > **;**

The TARGET statement specifies the response variable for the super learner model.

You can specify the following *option* after a slash (/):

LEVEL=*value*
specifies the variable type. You can specify the following *values*:

INTERVAL	specifies that the response variable is interval, which must be numeric.
NOMINAL	specifies that the response variable is nominal, also known as a classification variable, which can be numeric or character.

By default, LEVEL=INTERVAL for numeric variables and LEVEL=NOMINAL for categorical character variables.

Details: SUPERLEARNER Procedure

The super learner model is a cross-validation-based approach that combines predictive algorithms into an ensemble model. Asymptotically, it generates predictions that are at least as good as those of the best-performing algorithm.

Super Learner Algorithm

Given an input data set with a vector of predictors $\mathbf{x}$ and a response variable y, the super learner model consists of two layers of models that are used to describe the relationship between $\mathbf{x}$ and y.

The first layer includes a library of B individual base learner models:

$$f_b\left(\mathbf{x} \mid \boldsymbol{\beta}_b\right) \text{ for } b \in \{1, \ldots, B\}$$

where $\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \ldots, \boldsymbol{\beta}_B$ are the model parameters for base learners *1* through B.

The second layer contains the meta-learner model m, which is a function of the individual base learners in the first layer:

$$m\left(f_1\left(\mathbf{x} \mid \boldsymbol{\beta}_1\right), f_2\left(\mathbf{x} \mid \boldsymbol{\beta}_2\right), \ldots, f_B\left(\mathbf{x} \mid \boldsymbol{\beta}_B\right) \mid \boldsymbol{\alpha}\right)$$

where $\boldsymbol{\alpha}$ is the vector of meta-learner model parameters, also known as the super learner model coefficients.

Training a super learner amounts to estimating the model parameters in the base learner models and the meta-learner model. You estimate the base learner model parameters $\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \ldots, \boldsymbol{\beta}_B$ by fitting each base learner to the entire input data set. You estimate the super learner model coefficients $\boldsymbol{\alpha}$ by following these general steps:

1. Randomly split all observations in the input data set into K disjoint blocks.
2. Train each base learner by using only data in blocks *2* through K, excluding block *1*.
3. For each trained base learner, generate predictions for block *1*.
4. Repeat steps 2 and 3 $(K-1)$ times, each time leaving out block $k = 2, 3, \ldots, K$ for predictions. This step is known as k-fold cross-validation.
5. Across all blocks, fit a regression of the observed responses on the out-of-sample predictions from step 3. This is known as the meta-learning step.

For more information about k-fold cross-validation, see the section "*K*-Fold Cross-Validation" on page 1761. For more information about estimating the super learner model coefficients $\boldsymbol{\alpha}$, see the section "Meta-learning Methods" on page 1760.

Meta-learning Methods

Let $K_i \in \{1, \ldots, K\}$ denote the block of input data that contains observation i; let n_k denote the number of observations in block k; and let $\hat{\boldsymbol{\beta}}_b^{\sim k}$ denote the parameter estimates of base learner model b by training it on the input data, excluding block k. Let $\hat{f}_{b,i}^{\sim k} = f_b\left(\mathbf{x}_i \mid \hat{\boldsymbol{\beta}}_b^{\sim k}\right)$ denote the out-of-sample predicted response that is made by base learner b, given the inputs $\mathbf{x}_i$.

The meta-learner model m is defined as a linear combination of the base learner models:

$$m\left(f_1, \ldots, f_B \mid \boldsymbol{\alpha}\right) = \boldsymbol{\alpha}_1 f_1 + \boldsymbol{\alpha}_2 f_2 + \cdots + \boldsymbol{\alpha}_B f_B$$

Continuous Response

For a continuous response, you estimate the super learner coefficients $\boldsymbol{\alpha}$ by minimizing the following cross-validated empirical risk that is based on the squared loss function:

$$\sum_{k=1}^{K} \sum_{i:K_i=k} \left\{ y_i - m\left(\hat{f}_{1,i}^{\sim k}, \ldots, \hat{f}_{B,i}^{\sim k} \mid \boldsymbol{\alpha}\right)\right\}^2$$

Solving this minimization problem is equivalent to fitting a linear regression model (without the intercept) that regresses the observed responses on the base learner out-of-sample predictions. In addition, it is theoretically advantageous to restrict the search for the coefficients $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_B$ to the convex combination (Van der Laan, Polley, and Hubbard 2007); that is, $\boldsymbol{\alpha}_1 + \cdots + \boldsymbol{\alpha}_B = 1$ and $\boldsymbol{\alpha}_b \geq 0$ for all $b \in \{1, \ldots, B\}$. Solving the cross-validated empirical risk minimization with this convex constraint is the default meta-learning method for a continuous response.

Binary Response

For a binary response $y_i \in \{0, 1\}$, now $\hat{f}_{b,i}^{\sim k}$ denotes the out-of-sample predicted probability of $y_i = 1$ that is made by base learner b, given the inputs $\mathbf{x}_i$. Three meta-learning approaches are available for a binary response.

The first approach is to estimate the super learner coefficients $\boldsymbol{\alpha}$ by minimizing the cross-validated empirical risk that is based on the squared loss function.

The second approach is to estimate $\boldsymbol{\alpha}$ by maximizing the following binomial log likelihood:

$$\sum_{k=1}^{K} \sum_{i:K_i=k} \left\{ y_i \log\left(m\left(\hat{f}_{1,i}^{\sim k}, \ldots, \hat{f}_{B,i}^{\sim k} \mid \boldsymbol{\alpha}\right)\right) + (1 - y_i) \log\left(1 - m\left(\hat{f}_{1,i}^{\sim k}, \ldots, \hat{f}_{B,i}^{\sim k} \mid \boldsymbol{\alpha}\right)\right)\right\}$$

As in the continuous response case, the coefficients $\boldsymbol{\alpha}$ are restricted so that $\boldsymbol{\alpha}_1 + \cdots + \boldsymbol{\alpha}_B = 1$ and $\boldsymbol{\alpha}_b \geq 0$ for all $b \in \{1, \ldots, B\}$.

The third approach is to maximize the area under the ROC curve (AUC). This approach is effective when the response variable is highly imbalanced or when classification performance is the primary objective (LeDell, van der Laan, and Petersen 2016). To maximize the AUC in the meta-learning step, you estimate the super learner coefficients $\boldsymbol{\alpha}$ by maximizing the following:

$$\sum_{i:y_i=1} \sum_{j:y_j=0} I\left(m\left(\hat{f}_{1,i}^{\sim K_i}, \ldots, \hat{f}_{B,i}^{\sim K_i} \mid \boldsymbol{\alpha}\right) > m\left(\hat{f}_{1,j}^{\sim K_j}, \ldots, \hat{f}_{B,j}^{\sim K_j} \mid \boldsymbol{\alpha}\right)\right)$$

where $I(\cdot)$ is the indicator function. As in the continuous response case, the search for coefficients $\boldsymbol{\alpha}$ is restricted to the convex combination. The AUC optimization problem is solved by using the black-box solver. For more information about the black-box solver, see the discussion in the section "Black-Box Optimization Solver" in *SAS Optimization: Mathematical Optimization Procedures*. Because the objective function is nonconvex and piecewise constant, the black-box solver converges to a local optimum. The availability of AUC maximization is limited by the SAS software product that you are using. PROC SUPERLEARNER displays an error message and stops the analysis if METHOD=CCAUC is not available with your product license.

Discrete Super Learner

The discrete super learner model (also known as a cross-validated selector) is a variation of the super learner that selects a single base learner model that has the smallest cross-validated risk instead of searching for the optimal combination of all base learners. The base learner that has the smallest cross-validated risk receives a super learner coefficient of *1*, and the remaining base learners receive coefficients of *0*. You can train a discrete super learner by specifying the METHOD=CVSELECTOR option in the PROC SUPERLEARNER statement.

K-Fold Cross-Validation

K-fold cross-validation is a required step for training a super learner model. To strike a balance between computation time and predictive performance, you compute the default number of folds by using the recommendation of Phillips et al. (2023). In their approach, the number of folds is determined on the basis of the effective sample size. If the response is binary and $\bar{p}$ is its prevalence in the entire input data set, the effective sample size is computed as $n_{\text{eff}} = \min(n, 5n_{\text{rare}})$, where $n_{\text{rare}} = n \times \min(\bar{p}, 1 - \bar{p})$ is the number of rare events in the response. For a continuous response, the effective sample size is equal to the number of observations in the input data. The default number of folds K is computed as follows:

$$K = \begin{cases} n_{\text{eff}} & \text{if } n_{\text{eff}} < 30 \\ 20 & \text{if } 30 \le n_{\text{eff}} < 500 \\ 10 & \text{if } 500 \le n_{\text{eff}} < 5{,}000 \\ 5 & \text{if } 5{,}000 \le n_{\text{eff}} < 10{,}000 \\ 2 & \text{if } n_{\text{eff}} \ge 10{,}000 \end{cases}$$

You can also set the number of folds by specifying the KFOLD= option in the CROSSVALIDATION statement.

When the response variable or an important predictor is categorical, it can be beneficial to preserve its prevalence in each partitioned block of the input data, so that each partitioned block contains the same number of cases in each class of the variable. You can achieve this through stratified cross-validation. To do so, you specify the variable in the STRATVAR= option of the CROSSVALIDATION statement.

Cross-Validated Risk

The cross-validated (CV) risk of a base learner model is an estimate of its true out-of-sample predictive performance that is based on its performance in cross-validation. For a continuous response, the CV risk of base learner b is computed on the basis of the squared loss function:

$$\text{CVrisk}_b = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{y}_{i,b} - y_i\right)^2$$

where n is the number of observations in the input data set, $\hat{y}_{i,b}$ is the out-of-sample predicted value of observation i from base learner b, and y_i is the observed response value of observation i.

For a binary response $y_i \in \{0, 1\}$, the CV risk of base learner b can be computed on the basis of the squared loss, the log loss function, or the AUC loss function. For the log loss function, the CV risk is

$$\text{CVrisk}_b = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log\left(\hat{p}_{i,b}(y_i = 1)\right) + (1 - y_i)\log\left(\hat{p}_{i,b}(y_i = 0)\right)\right]$$

where n is the number of observations and $\hat{p}_{i,b}(y_i = 1)$ is base learner b's out-of-sample predicted probability that $y_i = 1$. The CV risk can be undefined if $\hat{p}_{i,b}(y_i = 1)$ is exactly 1 or 0.

For the AUC loss function, the CV risk is

$$\text{CVrisk}_b = 1 - \frac{1}{n_1 n_0}\sum_{i:y_i=1}\sum_{j:y_j=0} I(\hat{p}_{i,b}(y_i = 1) > \hat{p}_{j,b}(y_j = 1))$$

where n_1 and n_0 denote the number of observations for $y = 1$ and $y = 0$, respectively, and $I(\cdot)$ denotes the indicator function.

Supported Base Learner Model Types

The available base learner model types are limited by the SAS software product that you are using. If you attempt to use a base learner model type that is not available with your product license, PROC SUPERLEARNER displays an error message and stops the analysis.

Supported Model Selection Suboptions

This section provides more details about the model selection methods and suboptions that are available for the LOGSELECT and REGSELECT base learner model types. For these base learner model types, you can use the SELECTION= option in the BASELEARNER statement to specify a model selection method.

The SELECTION= option in the BASELEARNER statement is similar to the METHOD= option in the SELECTION statement of the LOGSELECT and REGSELECT procedures. When you use these procedures, you can choose to specify suboptions for each model selection method to further customize the model selection process. PROC SUPERLEARNER supports a subset of the suboptions in these procedures. Table 33.2 lists the applicable suboptions for each selection method.

Table 33.2 Applicable Suboptions by Selection Method

Suboption	BACKWARD	ELASTICNET	FORWARD	LAR	LASSO	MCP/SCAD	STEPWISE
ADAPTIVE		x			x		
ALPHA=						x	
BIGM=						x	
CHOOSE=	x	x	x	x	x	x	x
COMPETITIVE							x
CRITERION=	x		x				x
ENSCALE		x					
FAST	x						
L2=		x					
LAMBDA=		x				x	
LSCOEFFS				x	x		
MAXEFFECTS=		x	x	x	x		x
MINEFFECTS=	x						
MIXING=		x					
NLAMBDA=		x					
RELAXED					x		
SELECT=	x		x				x
SLENTRY=		x	x	x	x		x
SLSTAY=	x	x			x		x
SOLVER=		x				x	
STOP=	x	x	x	x	x		x

The following list includes a short description of each supported suboption:

ADAPTIVE
applies adaptive weights to each of the coefficients when SELECTION=LASSO or ELASTICNET.

ALPHA=*value*
specifies the regularization parameter α to use when you specify SELECTION=MCP or SCAD.

BIGM=*value*
specifies the constant $\mathcal{M}$ when you specify SOLVER=MILP.

CHOOSE=*criterion*
chooses from the list of models (at each step of the selection process) the model that yields the best value of the specified criterion. For the supported criteria, see the chapters for the relevant procedures.

COMPETITIVE
If you specify this option, then the SELECT= criterion is evaluated for all models in which an effect currently in the model is removed or an effect not yet in the model is added. The effect whose removal from or addition to the model yields the maximum improvement to the SELECT= criterion is removed or added.

CRITERION=*criterion*
is an alias for the SELECT= option.

ENSCALE
scales the results of the elastic net method to offset bias from the double shrinkage inherent in this method (Zou and Hastie 2005).

FAST
implements the computational algorithm of Lawless and Singhal (1978) to compute a first-order approximation to the remaining slope estimates for each subsequent elimination of a variable from the model.

L2=*value*
specifies the ridge regularization parameter to use when you specify SELECTION=ELASTICNET.

LAMBDA=*value*
specifies the regularization parameter λ to use when you specify SELECTION=ELASTICNET, MCP, or SCAD.

LSCOEFFS
requests a hybrid version of the LAR and LASSO methods, in which the sequence of models is determined by the LAR or LASSO algorithm but the coefficients of the parameters for the model at any step are determined by using ordinary least squares.

MAXEFFECTS=*n*
specifies the maximum number of effects in any model that is considered during the selection process.

MINEFFECTS=*n*
specifies the minimum number of effects in any model that is considered during backward selection.

MIXING=*value*
specifies the mixing parameter that controls the balance between the L_1 penalty and the L_2 penalty in elastic net selection. The *value* must be between 0 and 1, inclusive. If you specify 0, elastic net selection fits ridge regression models. If you specify 1, elastic net selection fits LASSO regression models. By default, MIXING=0.5.

NLAMBDA=*n*
specifies the number of regularization parameters to use in fitting candidate models. By default, NLAMBDA=1.

RELAXED
specifies relaxed LASSO selection, in which each candidate is determined by the LASSO algorithm but the coefficients of the parameters for each candidate at any step are determined by using a linear combination of LASSO coefficients and least squares estimates.

SELECT=*criterion*
specifies the criterion that the procedure uses to determine the order in which effects enter or leave at each step of the selection method. For supported criteria, see the chapter for the relevant procedure.

SLENTRY=*value*
specifies the significance level for entry when STOP=SL or SELECT=SL. By default, SLENTRY=0.05.

SLSTAY=*value*
specifies the significance level for staying in the model when STOP=SL or SELECT=SL. By default, SLSTAY=0.05.

SOLVER=*solver-value*
specifies the optimization solver when you specify SELECTION=ELASTICNET, MCP, or SCAD. You can specify the following *solver-values*:

ADMM	specifies the alternating direction methods of multipliers (ADMM) as the solver.
BFGS	specifies the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method as the solver.
LBFGS	specifies the limited-memory BFGS method as the solver.
MILP	specifies the mixed integer linear programming (MILP) solver.
NLP	specifies the nonlinear programming (NLP) solver.

For the MCP and SCAD selection methods, by default SOLVER=MILP.

STOP=SL | NONE | *criterion*
specifies a criterion that is used to stop the selection process. The criteria that are supported depend on the type of model that you are fitting. For information about the supported criteria, see the chapter for the relevant procedure.

You can specify the following values:

NONE	stops the selection process if no suitable candidates for addition or removal are found or if a size-based limit is reached.
SL	stops the selection process at the step where the significance level of the candidate for entry is greater than the SLENTRY= option value for addition steps and where the significance level of the candidate for removal is greater than the SLSTAY= option value.
criterion	stops the selection process if the selection process produces a local extremum of this *criterion* or if a size-based limit is reached.

For more information about the supported model selection methods, see the section "Model Selection Methods" on page 75 and the LOGSELECT and REGSELECT procedure documentation. For more information about the supported suboptions, see the section "SELECTION Statement" on page 42.

For example, the following code specifies a linear regression model with adaptive LASSO model selection:

```
baselearner 'adaptivelasso' regselect(selection=lasso(adaptive));
```

Missing Values

When you are training a super learner model, any observation that has missing values for an input variable, the response variable, or a variable that is used for stratified cross-validation is excluded from the analysis. For the purpose of deciding which observations are used, the input variables are the union of the set of variables that you specify in each INPUT statement and the set of variables that you specify in each *input-option* of a BASELEARNER statement. If you use the APPLYROWORDER option, any observation that has missing values for a partition or orderBy variable is not used for training. When you are scoring or computing predictive margins, any observation that has missing values for an input variable that is used for training is excluded from the analysis.

Displayed Output

The following sections describe the output that PROC SUPERLEARNER produces. The output is organized into various tables, which are discussed in order of their appearance.

Model Information

The "Model Information" table displays basic information about the super learner model, such as the input data, response variable, type of response variables, number of folds, number of base learner models, and meta-learning method.

Number of Observations

The "Number of Observations" table displays the number of observations that are read from the input data table and the number of observations that are used in the analysis. If you specify a BASELEARNER statement, the table displays the number of observations that are used for training. If you specify a MARGIN statement, the table displays the number of observations that are used for computing predictive margins. If you specify an OUTPUT statement, the table displays the number of observations that are used for scoring.

Super Learner Coefficients

The "Super Learner Coefficients" table displays the name, model type, super learner coefficient estimate, and cross-validated risk of each base learner model. Each row of the table corresponds to a base learner that you define in a BASELEARNER statement.

Scoring Information

The "Scoring Information" table displays the name of the data table that is used for scoring, the name of the item store, and the response variable.

Margins

The "Margins" table displays predictive margin estimates. Each row of the table corresponds to a predictive margin that you define in a MARGIN statement.

OutputCasTables Table

The OutputCasTables table is a special table that has information about each CAS table that is created during a CAS action execution. The information for each CAS table consists of the CAS table name, the caslib in which the table resides, and the number of columns and rows in the CAS table.

ODS Table Names

Each table that PROC SUPERLEARNER creates has a name that is associated with it. You must use this name to refer to the table when you use ODS statements. Table 33.3 lists the name of each table and a short description of its contents.

Table 33.3 ODS Tables Produced by PROC SUPERLEARNER

Table Name	Description	Statements	Option
Margins	Predictive margins	MARGIN	
ModelInfo	Information about the modeling environment	Default output	
NObs	Number of observations read and used	Default output	
OutputCasTables	See the section "OutputCasTables Table" on page 1767	OUTPUT STORE	OUT=
ScoreInfo	Information about the stored model used for scoring	PROC SUPERLEARNER	RESTORE=
SLCoefficients	Information about the super learner coefficients	Default output	

Examples: SUPERLEARNER Procedure

The examples in this chapter are available in the GitHub repository located at https://github.com/sascommunities/doc-supplement-casstat.

Example 33.1: Storing and Scoring

(View the complete code for this example (superlearnerex1.sas) in the example repository.)

When you use the SUPERLEARNER procedure to train a predictive model, you can save the model by using the STORE statement. You can then use the saved model to score new observations or compute predictive margins by fixing the values of one or more predictor variables. To do so, you use the RESTORE= option in the PROC SUPERLEARNER statement for subsequent scoring.

The following statements generate the data table inputData, which consists of 5,000 observations on a continuous response variable (y), five continuous predictor variables (x1–x5), and two binary predictor variables (z1,z2), in your CAS session:

```
data mylib.dataForTrain / single=yes;
   drop i;
   array x{5};
   array z{2};
   call streaminit(6520);
   pi=constant("pi");

   do i=1 to 5000;
     x1=rand("Normal");
     x2=rand("Uniform");
     z1=rand("Bernoulli",0.7);
     x3=rand("Normal",x1*z1,0.6*z1);
     z2=rand("Bernoulli",logistic(z1*x3));
     x4=rand("Normal",0,0.8*z2);
     if (x4 - z1) > 0 then x5 = x4 - z1;
     else x5 = (x4 - z1)**2;

     y=2+x1+x1*z1+sin(pi/2*x2)+x3**2+z2+x4**3+x5+rand("Normal");
   output;

   end;
run;
```

Note that the response variable y is generated on the basis of x1–x5, z1, and z2.

The following program trains a super learner ensemble model on the simulated data:

```
proc superlearner data=mylib.dataForTrain seed=2324;
   target y / level=interval;
   input z1 z2 / level=nominal;
   input x1-x5 / level=interval;
   baselearner 'lm' regselect;
```

```
    baselearner 'lasso_2way' regselect(selection=elasticnet(lambda=5 mixing=1))
      class=(z1 z2) effect=(z1|z2|x1|x2|x3|x4|x5 @2);
    baselearner 'gam' gammod class=(z1 z2) param(z1 z2 x1 x2)
      spline(x3) spline(x4) spline(x5);
    baselearner 'bart' bart(nTree=10 nMC=100);
    baselearner 'forest' forest;
    baselearner 'svm' svmachine;
    baselearner 'factmac' factmac(nfactors=4 learnstep=0.15);
    store out=mylib.slmodel;
run;
```

When you use PROC SUPERLEARNER for training, you must specify the TARGET statement, one or more INPUT statements, and two or more BASELEARNER statements.

You use the TARGET statement to specify y as the response variable. You indicate that it should be treated as a continuous variable by using the LEVEL= option. You specify two INPUT statements: the first one contains two variables, which are treated as categorical predictor variables; the second one contains five variables, which are labeled as continuous predictor variables.

When you use the BASELEARNER statement to specify a base learner model, you must first provide a name for the base learner in single quotes and then a model type. This statement supports various model types and options that you can choose for building a base learner. For more information, see the BASELEARNER statement.

In this example, you specify seven base learners by using seven BASELEARNER statements. For the first base learner, 'lm', you use the REGSELECT model type to specify a linear regression model.

You specify the second base learner, 'lasso_2way', again by using the REGSELECT model type. For this base learner, you use the SELECTION= option to specify the elastic net model selection method. You use the LAMDBA= and MIXING= suboptions to specify the lambda and mixing parameters, respectively. You use the EFFECT= option to specify that all the main effects and two-way interactions among the seven predictor variables are to be used as constructed effects for this base learner. When you specify the EFFECT= option, nominal predictor variables that you specify in the INPUT statement are no longer automatically considered as classification variables for this base learner. You include z1 and z2 in the CLASS= option so that the procedure treats them as classification variables for this base learner.

For the third base learner, you use the GAMMOD model type to specify a generalized additive model. You use the CLASS=, PARAM, and SPLINE input options to specify the model effects of the base learner. You require that z1 and z2 be treated as classification variables by including them in the CLASS= option. You include the main effects of z1, z2, x1, and x2 in the base learner by specifying the PARAM option. You include univariate spline effects of x3, x4, and x5 in the base learner by using the SPLINE option.

The remaining four base learners are defined by using the BART, FOREST, SVMACHINE, and FACTMAC model types, which correspond to specifying a Bayesian additive regression tree model, a forest model, a support vector machine model, and a factorization machine model, respectively. For the base learner 'bart', you customize the number of trees and the number of MCMC iterations by using the NTREE= and NMC= options, respectively. For the base learner 'factmac', you customize the number of factors and learning steps by using the NFACTORS= and LEARNSTEP= options, respectively.

You use the STORE statement to save the trained super learner ensemble model as an item store. You can use the saved item store to score new observations without having to retrain the model again. This is illustrated later in this example.

Output 33.1.1 displays basic information about the super learner ensemble specification, and Output 33.1.2 displays the estimated coefficient for each base learner.

Output 33.1.1 Model Information and Number of Observations

The SUPERLEARNER Procedure

Model Information	
Data Source	DATAFORTRAIN
Target Variable	y
Target Variable Type	INTERVAL
Number of Folds	5
Number of Base Learners	7
Loss Function	Quadratic Loss
Meta-learning Method	Convex-constrained Least Squares
Random Number Seed	2324

Number of Observations Read	5000
Number of Observations Used	5000

Output 33.1.2 Super Learner Coefficients

Super Learner Model Coefficients

Name	Model Type	Coefficient	Cross-Validated Risk
lm	Ordinary Linear Least Squares	0	4.35073
lasso_2way	Ordinary Linear Least Squares	0.21575	1.40477
gam	Generalized Additive Model	0.71463	1.15176
bart	Bayesian Additive Regression Trees	0.06962	1.69381
forest	Forest	0	1.78765
svm	Support Vector Machine	0	4.52188
factmac	Factorization Machine	0	2.60129

By default, based on the input data sample size, the input data are divided into five folds for cross-validation. Because the target variable is continuous, by default the convex-constrained least squares meta-learning method is used to estimate the weight of each base learner in the super learner ensemble, and the quadratic loss is used to compute the cross-validated risk of each base learner.

The following DATA step code uses the same data-generating process to create new observations to be used for scoring:

```
data mylib.dataForScore / single=yes;
   drop i;
   array x{5};
   array z{2};
   call streaminit(2456);
   pi=constant("pi");

   do i=1 to 1000;
     x1=rand("Normal");
```

```
      x2=rand("Uniform");
      z1=rand("Bernoulli",0.7);
      x3=rand("Normal",x1*z1,0.6*z1);
      z2=rand("Bernoulli",logistic(z1*x3));
      x4=rand("Normal",0,0.8*z2);
      if (x4 - z1) > 0 then x5 = x4 - z1;
      else x5 = (x4 - z1)**2;

      y=2+x1+x1*z1+sin(pi/2*x2)+x3**2+z2+x4**3+x5+rand("Normal");
   output;

   end;
run;
```

The following statements show how to use PROC SUPERLEARNER to score the new data by using the previously fitted model:

```
proc superlearner data=mylib.dataForScore restore=mylib.slmodel;
   output out=mylib.scoredData1 learnerpred;
run;
```

You specify the fitted model in the RESTORE= option. You use the DATA= option to specify the data table of observations to score, and you use the OUTPUT statement to designate the output data table to contain the predicted responses. The LEARNERPRED option includes the predicted values from each base learner in the output data table, in addition to the predicted responses from the super learner ensemble. The RESTORE= option and OUTPUT statement are both required when you use PROC SUPERLEARNER to score data by using a previously saved model.

The following statements print the first 10 observations of the scored data table. The results are shown in Output 33.1.3.

```
proc print data=mylib.scoredData1(obs=10);
run;
```

Output 33.1.3 First 10 Observations in the Output Data Table

Obs	P_y	lm	lasso_2way	gam	bart	forest	svm	factmac
1	6.65683	6.54213	6.50964	6.71963	6.46835	6.50313	6.86751	4.30350
2	1.98392	1.85226	2.49392	1.79935	2.29800	2.18953	1.91218	2.03633
3	2.61899	3.58495	2.75508	2.57583	2.64024	2.56014	3.21489	3.15332
4	4.42276	5.64113	4.70199	4.32795	4.53066	4.32606	5.15384	4.35422
5	2.23673	2.88074	2.44980	2.22783	1.66777	2.56502	2.54407	3.13678
6	5.53240	7.17779	6.03486	5.29490	6.41315	5.92846	6.63275	6.20529
7	2.86278	3.19713	3.02958	2.86950	2.27690	2.76490	2.82833	3.28876
8	8.03762	7.80369	8.38578	7.90433	8.32684	8.12643	7.62153	7.72098
9	3.34112	4.96500	3.41055	3.36905	2.83928	3.39109	4.51414	5.07560
10	4.26333	3.58532	3.94204	4.35639	4.30383	4.30255	3.20928	3.13718

The P_y column contains super learner predictions, and the other columns contain predictions from each individual base learner.

You can also use the saved model to compute predictive margins. To compute a predictive margin, you use the MARGIN statement to specify a fixed value for one or more variables. Then the procedure scores each

observation in an input data table that you specify in the DATA= option, but the values of the variable(s) that you specify in the MARGIN statement are replaced by the specified value(s). By default, PROC SUPERLEARNER prints the means of the predicted responses. You can also directly access the predicted response values for each set of interventions by specifying the MARGINPRED option in the OUTPUT statement to include them in the output data table.

The following code shows how to compute predictive margins and the predicted response values under interventions by using PROC SUPERLEARNER and the trained model that you saved earlier in this example:

```
proc superlearner data=mylib.dataForScore restore=mylib.slmodel;
   margin "Scenario1" z1=0 x1=0.25;
   margin "Scenario2" x1=0.25;
   margin "Scenario3" x1=0.5;
   margin "Scenario4" x1=0.75;
   output out=mylib.scoredData2 marginpred;
run;
```

You specify the saved model mylib.slmodel in the RESTORE= option, and you specify the data table that contains new observations to be scored in the DATA= option. Four predictive margins are specified using the MARGIN statement. The first scenario fixes the values of z1 and x1 to 0 and 0.25, respectively. The remaining three predictive margins fix the value of x1 to 0.25, 0.5, and 0.75, respectively.

The "Scoring Information" table in Output 33.1.4 displays the names of the saved model, the data table that is used for training, the data table of scored observations, and the target variable. The "Number of Observations" table displays the number of observations that are used to score and compute the predictive margins.

Output 33.1.4 Scoring Information and Number of Observations

The SUPERLEARNER Procedure

Scoring Information	
Store	SLMODEL
Scoring Table	DATAFORSCORE
Training Table	DATAFORTRAIN
Target Variable	y

Number of Observations Read	1000
Number of Observations Used for Computing Predictive Margins	1000
Number of Observations Used for Scoring	1000

The "Predictive Margins" table in Output 33.1.5 displays the means of the predicted margin estimates for the interventions that are specified by a MARGIN statement.

Output 33.1.5 Predictive Margins

Predictive Margins	
Name	**Estimate**
Scenario1	4.99108
Scenario2	5.27512
Scenario3	5.62181
Scenario4	6.00051

The following statements print the first 10 observations of the scored data table that contain predicted responses without intervention and predicted responses under each intervening scenario. The results are shown in Output 33.1.6.

```
proc print data=mylib.scoredData2(obs=10);
run;
```

Output 33.1.6 First 10 Observations in the Output Data Table

Obs	P_y	Scenario1	Scenario2	Scenario3	Scenario4
1	6.65683	5.55252	7.11563	7.41870	7.75641
2	1.98392	3.85930	3.85930	4.15335	4.48204
3	2.61899	3.56465	3.89671	4.24107	4.62008
4	4.42276	4.22765	4.57249	4.93537	5.33288
5	2.23673	3.57691	3.92740	4.25761	4.62246
6	5.53240	4.30642	4.33119	4.69125	5.08595
7	2.86278	4.11930	4.46090	4.77948	5.13271
8	8.03762	6.60072	7.11505	7.51966	7.94006
9	3.34112	4.40659	4.38477	4.71462	5.07911
10	4.26333	4.92668	5.26729	5.56103	5.88941

The P_y column again contains super learner predictions, and the other columns contain predictions from each intervening scenario.

Example 33.2: Causal Effect Estimation

(View the complete code for this example (superlearnerex2.sas) in the example repository.)

This example illustrates how you can use the SUPERLEARNER procedure for causal effect estimation.

This example uses the SmokingWeight data set available in the examples for the CAUSALTRT procedure in *SAS/STAT User's Guide*. These data are a subset of data from the NHANES I Epidemiologic Follow-Up Study (NHEFS) in Hernán and Robins (2020). For the study, medical and behavioral information was collected during an initial physical examination and again at follow-up interviews about 10 years later. As in the PROC CAUSALTRT examples, the question of interest here is what effect does quitting smoking, as recorded in the variable Quit, have on an individual's change in weight (measured in kilograms), as recorded in the variable Change.

To create the data table SmokingWeight in your CAS session, you can modify the DATA step that is used to create the data set in PROC CAUSALTRT to create the data table in your CAS engine library . For this example, it is assumed that your CAS engine libref is named mylib, but you can substitute any appropriately defined CAS engine libref.

In addition to the treatment variable Quit and outcome variable Change, the data include the following variables:

- Activity: level of daily activity, with values 0, 1, and 2
- Age: age in 1971

- BaseWeight: weight in kilograms in 1971
- Education: level of education, with values 0, 1, 2, 3, and 4
- Exercise: amount of regular recreational exercise, with values 0, 1, and 2
- PerDay: number of cigarettes smoked per day in 1971
- Race: 0 for white; 1 otherwise
- Sex: 0 for male; 1 for female
- Weight: weight in kilograms at the follow-up interview
- YearsSmoke: number of years an individual has smoked

For this example, it is assumed that the variables Activity, Age, BaseWeight, Education, Exercise, PerDay, Race, Sex, and YearsSmoke make up a valid adjustment set that you can use to obtain unbiased estimates of the potential outcome means.

Targeted maximum likelihood estimation (TMLE) is a doubly robust method that combines modeling of both the treatment and outcome variable to estimate the potential outcome means and causal effects. This example uses the super learner to model the treatment and outcome variables, because it has been proven effective when combined with TMLE for causal effect estimation.

The following statements use PROC SUPERLEARNER to fit a super learner model for the treatment variable Quit and create an output data table that contains the predictions from the fitted model:

```
proc superlearner data=mylib.SmokingWeight seed=2324;
   target Quit / level=nominal;
   input Sex Race Education Exercise Activity / level=nominal;
   input Age YearsSmoke PerDay / level=interval;
   baselearner 'logistic' logselect;
   baselearner 'logistic_2way' logselect
      class=(Sex Race Education Exercise Activity)
      effect=(Age|YearsSmoke|PerDay|Sex|Race|Education|Exercise|Activity @ 2);
   baselearner 'logistic_ridge' logselect(selection=ELASTICNET(lambda=5 mixing=0))
      class=(Sex Race Education Exercise Activity)
      effect=(Age|YearsSmoke|PerDay|Sex|Race|Education|Exercise|Activity @ 2);
   baselearner 'gam' gammod;
   baselearner 'bart' bart(nTree=10 nMC=100);
   output out=mylib.swTrtEstData copyvars=(_ALL_);
run;
```

When you use PROC SUPERLEARNER, you must specify the TARGET statement, at least one INPUT statement, and at least two BASELEARNER statements. You specify the input data table to use for training the super learner model in the DATA= option. The TARGET statement specifies information about the target variable. Each INPUT statement specifies predictor variables of the same type. Each BASELEARNER statement specifies a base learner model to be included in the super learner model.

In the TARGET statement, you use the LEVEL= option to indicate that you want to treat the treatment variable Quit as a categorical variable. You specify the categorical predictor variables in one INPUT statement and the continuous predictor variables in another.

In this example, five base learners are specified. For the first base learner, 'logistic', you specify a logistic regression model by using the LOGSELECT model type. The second base learner, 'logistic_2way', also specifies a logistic regression model but includes different model effects. You use the CLASS= option to specify the classification variables for this base learner. You use the EFFECT= option to construct main and two-way interaction effects that are based on the eight input variables and to include them in this base learner. The third base learner, 'logistic_ridge', adds a ridge penalty to the second base learner specification. You do this by using the SELECTION= option to apply the elastic net model selection method to this base learner. You set the lambda parameter to 5 by using the LAMDBA= option, and you set the mixing parameter to 0 by using the MIXING= option. For the fourth base learner, 'gam', you use the GAMMOD model type to specify a nonparametric logistic regression. You specify a Bayesian additive regression trees model by using the BART model type. You set the number of trees to 10 and the number of MCMC iterations to 100 by using the NTREES= and NMC= options, respectively.

In the OUTPUT statement, you use the COPYVARS= option to copy all variables from the input data table to the output data table.

The "Model Information" table in Output 33.2.1 summarizes important information, including the input data source, response (target) variable and its type, number of folds that are used in cross-validation, number of base learners, loss function, meta-learning method, and random number seed.

Output 33.2.1 Model Information

The SUPERLEARNER Procedure

Model Information	
Data Source	SMOKINGWEIGHT
Target Variable	Quit
Target Variable Type	NOMINAL
Number of Folds	10
Number of Base Learners	5
Loss Function	Log Loss
Meta-learning Method	Convex-constrained Binomial Likelihood Maximization
Random Number Seed	2324

By default, based on the sample size used for training, the input data are divided into 10 folds for cross-validation. Because the response variable is categorical, the CCLOGLIK meta-learning method is used by default to compute the super learner coefficients, and the log loss function is used to compute the cross-validated risks of base learners.

Output 33.2.2 displays the "Number of Observations" table. In this analysis, a number of observations are not used for training, because there are missing values in the predictor variables and the response variable. A number of observations are not used for scoring, because there are missing predicted treatment assignment probabilities from the base learners. These probabilities are missing because there are missing values in the predictor variables.

Output 33.2.2 Number of Observations

Number of Observations Read	1746
Number of Observations Used for Training	1629
Number of Observations Used for Scoring	1629

Output 33.2.3 displays the super learner coefficient estimates and the cross-validated risks of the base learners. The base learner 'logistic' has the smallest cross-validated risk and the largest coefficient in the ensemble.

Output 33.2.3 Super Learner Coefficients

Super Learner Model Coefficients

Name	Model Type	Coefficient	Cross-Validated Risk
logistic	Logistic Regression	0.54565	0.55602
logistic_2way	Logistic Regression	0.01448	0.59092
logistic_ridge	Logistic Regression	0.05880	0.56960
gam	Generalized Additive Model	0	0.56010
bart	Bayesian Additive Regression Trees	0.38106	0.55771

By default, PROC SUPERLEARNER models the probability that the variable Quit takes the value 0. The name of the variable in the output data table that contains the predicted probability is P_Quit0. The following DATA step creates the variable P_Quit1, which contains the predicted probability of being assigned to the treatment condition, Quit=1:

```
data mylib.swTrtEstData;
  set mylib.swTrtEstData;
  P_Quit1 = 1 - P_Quit0;
run;
```

The following statements use PROC SUPERLEARNER to fit a super learner model for the outcome variable Change and create an output data table that contains the predictions from the fitted model:

```
proc superlearner data=mylib.swTrtEstData seed=2324;
   baselearner 'linear' regselect;
   baselearner 'linear_2way' regselect
      class=(Sex Race Education Exercise Activity Quit)
      effect=(Age|YearsSmoke|PerDay|Sex|Race|Education|Exercise|Activity|Quit @ 2);
   baselearner 'linear_ridge' regselect(selection=ELASTICNET(lambda=5 mixing=0))
      class=(Sex Race Education Exercise Activity Quit)
      effect=(Age|YearsSmoke|PerDay|Sex|Race|Education|Exercise|Activity|Quit @ 2);
   baselearner 'gam' gammod;
   baselearner 'bart' bart(nTree=10 nMC=100);
   margin 'quitsmoking' Quit=1;
   margin 'noquitsmoking' Quit=0;
   target Change / level=interval;
   input Sex Race Education Exercise Activity Quit / level=nominal;
   input Age YearsSmoke PerDay / level=interval;
   output out=mylib.swDREstData marginpred copyvars=(Quit Change P_Quit1 P_Quit0);
run;
```

The syntax is almost identical to the syntax that is used to model the treatment probability. The first three base learners are specified using the REGSELECT model instead of the LOGSELECT model for modeling a continuous response variable. Also, the treatment variable Quit is added as a predictor variable. You use two MARGIN statements to intervene and set the Quit variable to two different values. You use the MARGINPRED option in the OUTPUT statement to include predictions that are based on the intervened values of the Quit variable in the output data table.

The "Model Information" table in Output 33.2.4 again summarizes important information about the fitted outcome mean model.

Output 33.2.4 Model Information

The SUPERLEARNER Procedure

Model Information	
Data Source	SWTRTESTDATA
Target Variable	Change
Target Variable Type	INTERVAL
Number of Folds	10
Number of Base Learners	5
Loss Function	Quadratic Loss
Meta-learning Method	Convex-constrained Least Squares
Random Number Seed	2324

By default, based on the sample size used for training, the input data are divided into 10 folds for cross-validation. Because the response variable is continuous, CCLS is used as the default meta-learning method to compute the super learner coefficients.

Output 33.2.5 displays the "Number of Observations" table. In this analysis, a number of observations are not used for training, because there are missing values in the predictor variables and the response variable. A number of observations are not used for scoring and computing the predictive margins, because there are missing predicted responses from the base learners. These predicted responses are missing because there are missing values in the predictor variables.

Output 33.2.5 Number of Observations

Number of Observations Read	1746
Number of Observations Used for Training	1566
Number of Observations Used for Computing Predictive Margins	1629
Number of Observations Used for Scoring	1629

Output 33.2.6 displays the super learner coefficient estimates and the cross-validated risks of the base learners. The base learner 'gam' has the smallest cross-validated risk and the largest coefficient in the ensemble.

Output 33.2.6 Super Learner Coefficients

Super Learner Model Coefficients			
Name	**Model Type**	**Coefficient**	**Cross-Validated Risk**
linear	Ordinary Linear Least Squares	0	57.31210
linear_2way	Ordinary Linear Least Squares	0.18160	59.30354
linear_ridge	Ordinary Linear Least Squares	0.15837	57.85823
gam	Generalized Additive Model	0.40464	56.70110
bart	Bayesian Additive Regression Trees	0.25539	56.93869

The following statements use the CAEFFECT procedure to estimate potential outcome means and their difference by using the doubly robust TMLE method:

```
proc caeffect data=mylib.swDREstData inference method=TMLE;
   treatvar Quit;
   outcomevar Change;
   pom treatLev=1 treatProb=P_Quit1 predout=quitsmoking;
   pom treatLev=0 treatProb=P_Quit0 predout=noquitsmoking;
   difference evtLev=1;
run;
```

For this example, you use the TREATVAR and OUTCOMEVAR statements to specify the treatment and outcome variables, respectively. You use two POM statements to specify the potential outcomes of interest under the two treatment levels. In each POM statement, you supply the variables in the input data table that contains the predicted treatment assignment probabilities and outcome means that the TMLE analysis requires. You do this by specifying the TREATPROB= and PREDOUT= options, respectively. The variable names of the predicted outcome means correspond to the names that you specify in the MARGIN statements earlier. You use the DIFFERENCE statement to request a difference between the potential outcome mean estimates.

Output 33.2.7 displays the potential outcome mean estimates and their difference. The "POM Estimates" table indicates the level of treatment that defines the potential outcome and the estimate of its mean.

Output 33.2.7 Potential Outcome Mean and Difference Estimates

The CAEFFECT Procedure

POM Estimates

Treatment Level	Estimate	Standard Error	95% Confidence Limits		Z	Pr > |Z|
1	5.06841	0.43041	4.22482	5.91201	11.78	<.0001
0	1.81793	0.21516	1.39622	2.23964	8.45	<.0001

POM Differences

Treatment Levels							
Event	Reference	Difference	Standard Error	95% Confidence Limits		Z	Pr > |Z|
1	0	3.2505	0.47196	2.32545	4.17552	6.89	<.0001

The "POM Differences" table shows the requested difference in potential outcome mean estimates. With the proper causal assumptions, the difference in potential outcome mean estimates corresponds to an estimate of the average treatment effect (ATE) of a weight change that is approximately 3.2 kilograms greater because the subjects quit smoking.

References

Breiman, L. (1996). "Stacked Regressions." *Machine Learning* 24:49–64.

Dudoit, S., and van der Laan, M. J. (2005). "Asymptotics of Cross-Validated Risk Estimation in Estimator Selection and Performance Assessment." *Statistical Methodology* 2:131–154.

Hernán, M. A., and Robins, J. M. (2020). *Causal Inference: What If*. Boca Raton, FL: Chapman & Hall/CRC.

Lane, P. W., and Nelder, J. A. (1982). "Analysis of Covariance and Standardization as Instances of Prediction." *Biometrics* 38:613–621.

Lawless, J. F., and Singhal, K. (1978). "Efficient Screening of Nonnormal Regression Models." *Biometrics* 34:318–327.

LeDell, E., van der Laan, M. J., and Petersen, M. (2016). "AUC-Maximizing Ensembles through Metalearning." *International Journal of Biostatistics* 12:203–218.

Naimi, A. I., and Balzer, L. B. (2018). "Stacked Generalization: An Introduction to Super Learning." *European Journal of Epidemiology* 33:459–464.

Phillips, R. V., van der Laan, M. J., Lee, H., and Gruber, S. (2023). "Practical Considerations for Specifying a Super Learner." *International Journal of Epidemiology* 52:1276–1285.

Van der Laan, M. J., and Dudoit, S. (2003). *Unified Cross-Validation Methodology for Selection among Estimators and a General Cross-Validated Adaptive Epsilon-Net Estimator: Finite Sample Oracle Inequalities and Examples*. UC Berkeley Division of Biostatistics Working Paper Series, No. 130, University of California Berkeley.

Van der Laan, M. J., Polley, E. C., and Hubbard, A. E. (2007). "Super Learner." *Statistical Applications in Genetics and Molecular Biology* 6:25.

Van der Laan, M. J., and Rose, S. (2011). *Targeted Learning: Causal Inference for Observational and Experimental Data*. New York: Springer.

Van der Vaart, A. W., Dudoit, S., and van der Laan, M. J. (2006). "Oracle Inequalities for Multi-fold Cross Validation." *Statistics and Decisions* 24:351–371.

Wolpert, D. H. (1992). "Stacked Generalization." *Neural Networks* 5:241–259.

Zou, H., and Hastie, T. (2005). "Regularization and Variable Selection via the Elastic Net." *Journal of the Royal Statistical Society, Series B* 67:301–320.

Chapter 34
TREESPLIT Procedure

Contents

Overview: TREESPLIT Procedure **1782**
PROC TREESPLIT Features 1783
Using CAS Sessions and CAS Engine Librefs 1783
Getting Started: TREESPLIT Procedure **1784**
Syntax: TREESPLIT Procedure **1791**
PROC TREESPLIT Statement 1791
AUTOTUNE Statement 1800
CLASS Statement 1815
CODE Statement 1815
CROSSVALIDATION Statement 1816
FREQ Statement 1817
GROW Statement 1817
MITIGATEBIAS Statement 1819
MODEL Statement 1821
OUTPUT Statement 1821
PARTITION Statement 1822
PRUNE Statement 1823
SAVESTATE Statement 1824
VIICODE Statement 1825
WEIGHT Statement 1825
Details: TREESPLIT Procedure **1826**
Building a Decision Tree 1826
Splitting Criteria 1829
Splitting Nominal Values 1831
Pruning 1833
Scoring 1837
Measures of Model Fit 1838
Variable Importance 1839
Hyperparameter Tuning 1842
k-fold Cross Validation 1846
Bias Mitigation 1847
Displayed Output 1849
ODS Table Names 1852
ODS Graphics 1854
INPUT and TARGET Statement Syntax 1854
Examples: TREESPLIT Procedure **1856**

Example 34.1: Creating a Binary Classification Tree with Validation Data 1856
Example 34.2: Creating a Regression Tree 1861
Example 34.3: Assessing Variable Importance 1865
Example 34.4: Mitigating Bias of a Classification Tree 1866
Example 34.5: Tuning the Hyperparameters of a Decision Tree 1870
References **1873**

Overview: TREESPLIT Procedure

The TREESPLIT procedure builds tree-based statistical models for classification and regression in SAS Viya. The procedure produces a classification tree, which models a categorical response, or a regression tree, which models a continuous response. Both types of trees are called decision trees, because the model is expressed as a series of if-then statements. For each type of tree, you specify a response variable (also called a target variable), whose values you want PROC TREESPLIT to predict, and one or more input variables (called predictor variables), whose values the procedure uses to predict the values of the target variable.

The predictor variables for tree models can be categorical or continuous. The set of all possible combinations of the predictor variables is called the predictor space. The model is based on partitioning the predictor space into nonoverlapping segments, which correspond to the terminal nodes (called leaves) of the tree. Partitioning is done repeatedly, starting with the root node, which contains all the data, and continuing until a stopping criterion is met. At each step, the parent node is split into child nodes by selecting a predictor variable and a split value for that variable that minimize the variability, according to a specified measure (or the default measure), in the response variable across the child nodes. Various measures, such as the Gini index, entropy, and residual sum of squares, can be used to assess candidate splits for each node. The selected predictor variable and its split value are called the splitting rule.

Tree models are built from training data for which the response values are known, and these models are subsequently used to score (classify or predict) response values for new data. For classification trees, the most frequent response level of the training observations in a leaf is used to classify observations in that leaf. For regression trees, the average response of the training observations in a leaf is used to predict the response for observations in that leaf. The splitting rules that define the leaves provide the information that is needed to score new data; these rules consist of the splitting rules and default rules for each node.

The process of building a decision tree begins with growing a large, full tree. The full tree can overfit the training data, resulting in a model that does not adequately generalize to new data. To prevent overfitting, the full tree is often pruned back to a smaller subtree that balances the goals of fitting training data and predicting new data. Two commonly applied approaches for finding the best subtree are cost-complexity pruning (Breiman et al. 1984) and C4.5 pruning (Quinlan 1993). For more information, see the section "Building a Decision Tree" on page 1826.

Compared with other regression and classification methods, tree models have the advantage that they are easy to interpret and visualize, especially when the tree is small. Tree-based methods scale well to large data, and they offer various methods of handling missing values.

However, tree models have limitations. Regression tree models fit response surfaces that are constant over rectangular regions of the predictor space, so they often lack the flexibility needed to capture smooth relationships between the predictor variables and the response. Another limitation of tree models is that small

changes in the data can lead to very different splits, and this undermines the interpretability of the model (Hastie, Tibshirani, and Friedman 2009; Kuhn and Johnson 2013).

PROC TREESPLIT Features

The main features of the TREESPLIT procedure are as follows:

- provides a computationally efficient strategy for generating candidate splits
- provides the cost-complexity, C4.5, and reduced-error methods of pruning trees
- supports the use of validation data for selecting the best subtree during pruning
- supports the use of *k*-fold cross validation for cost-complexity pruning when validation data are not present
- provides various methods of handling missing values
- creates tree diagrams and plots for pruning analysis
- computes statistics for assessing model fit
- computes measures of variable importance
- produces a file that contains SAS DATA step code for scoring new data
- provides an output data table that contains leaf assignments and predicted values for observations

The TREESPLIT procedure uses ODS Graphics to create plots as part of its output. For information about the statistical graphics available with the TREESPLIT procedure, see the PLOTS options in the PROC TREESPLIT statement and the section "ODS Graphics" on page 1854.

Using CAS Sessions and CAS Engine Librefs

SAS Cloud Analytic Services (CAS) is the analytic server and associated cloud services in SAS Viya. This section describes how to create a CAS session and set up a CAS engine libref that you can use to connect to the CAS session. It assumes that you have a CAS server already available; contact your system administrator if you need help starting and terminating a server. This CAS server is identified by specifying the host on which it runs and the port on which it listens for communications. To simplify your interactions with this CAS server, the host information and port information for the server are stored as SAS option values that are retrieved automatically whenever this CAS server needs to be accessed. You can examine the host and port values for the server at your site by using the following statements:

```
proc options option=(CASHOST CASPORT);
run;
```

In addition to starting a CAS server, your system administrator might also have created a CAS session and a CAS engine libref for your use. You can define your own sessions and CAS engine librefs that connect to the CAS server as shown in the following statements:

```
cas mysess;
libname mylib cas sessref=mysess;
```

The CAS statement creates the CAS session named mysess, and the LIBNAME statement creates the mylib CAS engine libref that you use to connect to this session. It is not necessary to explicitly name the CASHOST and CASPORT of the CAS server in the CAS statement, because these values are retrieved from the corresponding SAS option values.

If you have created the mysess session, you can terminate it by using the TERMINATE option in the CAS statement as follows:

```
cas mysess terminate;
```

For more information about the CAS and LIBNAME statements, see the section "Introduction to Shared Concepts" on page 6 in Chapter 2, "Shared Concepts."

Getting Started: TREESPLIT Procedure

(View the complete code for this example at https://github.com/sascommunities/doc-supplement-casstat/tree/main/Examples/treesplgs1.sas.)

NOTE: Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections "Using CAS Sessions and CAS Engine Librefs" on page 7 and "Loading a SAS Data Set onto a CAS Server" on page 7 in Chapter 2, "Shared Concepts."

This example explains basic features of the TREESPLIT procedure for building a classification tree. The data contain measurements of 13 chemical attributes for 178 samples of wine. Each wine is derived from one of three cultivars that are grown in the same area of Italy, and the goal of the analysis is a model that classifies samples into cultivar groups. The data are available from the UCI Irvine Machine Learning Repository (Lichman 2013). [1]

The following statements create a data set named Wine that contains the measurements:

```
data Wine;
   %let url = http://archive.ics.uci.edu/ml/machine-learning-databases;
   infile "&url/wine/wine.data" url delimiter=',';
   input Cultivar Alcohol Malic Ash Alkan Mg TotPhen
         Flav NFPhen Cyanins Color Hue ODRatio Proline;
   label Cultivar = "Cultivar"
         Alcohol  = "Alcohol"
         Malic    = "Malic Acid"
         Ash      = "Ash"
```

[1] Disclaimer: SAS may reference other websites or content or resources for use at Customer's sole discretion. SAS has no control over any websites or resources that are provided by companies or persons other than SAS. Customer acknowledges and agrees that SAS is not responsible for the availability or use of any such external sites or resources, and does not endorse any advertising, products, or other materials on or available from such websites or resources. Customer acknowledges and agrees that SAS is not liable for any loss or damage that may be incurred by Customer or its end users as a result of the availability or use of those external sites or resources, or as a result of any reliance placed by Customer or its end users on the completeness, accuracy, or existence of any advertising, products, or other materials on, or available from, such websites or resources.

```
            Alkan    = "Alkalinity of Ash"
            Mg       = "Magnesium"
            TotPhen  = "Total Phenols"
            Flav     = "Flavonoids"
            NFPhen   = "Nonflavonoid Phenols"
            Cyanins  = "Proanthocyanins"
            Color    = "Color Intensity"
            Hue      = "Hue"
            ODRatio  = "OD280/OD315 of Diluted Wines"
            Proline  = "Proline";
run;
```

The following DATA step loads the mylib.Wine data into your CAS session. This DATA step assumes that your CAS engine libref is named mylib, but you can substitute any appropriately defined CAS engine libref.

```
data mylib.Wine;
   set Wine;
run;
```

The following statements print the first 10 observations of Wine, as shown in Figure 34.1:

```
proc print data=Wine(obs=10); run;
```

Figure 34.1 Partial Listing of mylib.Wine

Obs	Cultivar	Alcohol	Malic	Ash	Alkan	Mg	TotPhen	Flav	NFPhen	Cyanins	Color	Hue	ODRatio	Proline
1	1	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	2.29	5.64	1.04	3.92	1065
2	1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	3.40	1050
3	1	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81	5.68	1.03	3.17	1185
4	1	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2.18	7.80	0.86	3.45	1480
5	1	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1.82	4.32	1.04	2.93	735
6	1	14.20	1.76	2.45	15.2	112	3.27	3.39	0.34	1.97	6.75	1.05	2.85	1450
7	1	14.39	1.87	2.45	14.6	96	2.50	2.52	0.30	1.98	5.25	1.02	3.58	1290
8	1	14.06	2.15	2.61	17.6	121	2.60	2.51	0.31	1.25	5.05	1.06	3.58	1295
9	1	14.83	1.64	2.17	14.0	97	2.80	2.98	0.29	1.98	5.20	1.08	2.85	1045
10	1	13.86	1.35	2.27	16.0	98	2.98	3.15	0.22	1.85	7.22	1.01	3.55	1045

The variable Cultivar is a nominal categorical variable that has levels 1, 2, and 3, and the 13 attribute variables are continuous.

The following statements use the TREESPLIT procedure to create a classification tree:

```
ods graphics on;

proc treesplit data=mylib.Wine seed=54321;
   class Cultivar;
   model Cultivar = Alcohol Malic Ash Alkan Mg TotPhen Flav
                            NFPhen Cyanins Color Hue ODRatio Proline;
   grow entropy;
   prune costcomplexity(leaves=SE);
run;
```

The MODEL statement specifies Cultivar as the response variable and the variables to the right of the equals sign as the predictor variables. The inclusion of Cultivar in the CLASS statement designates it as a categorical

response variable and requests a classification tree. All the predictor variables are treated as continuous variables because none are included in the CLASS statement.

The GROW and PRUNE statements control two fundamental aspects of building classification and regression trees: growing and pruning. You use the GROW statement to specify the criterion for recursively splitting parent nodes into child nodes as the tree is grown. For classification trees, the default criterion is entropy; for more information, see the section "Splitting Criteria" on page 1829.

By default, the growth process continues until the tree reaches a maximum depth of 10 (you can specify a different limit by using the MAXDEPTH= option). The result is often a large tree that overfits the data and is likely to perform poorly in predicting future data. A recommended strategy for avoiding this problem is to prune the tree to a smaller subtree that minimizes prediction error. You can use the PRUNE statement to specify the method of pruning. The default method is cost complexity.

The default output includes several informational tables, which are shown in Figure 34.2 through Figure 34.7. The "Model Information" table in Figure 34.2 provides information about the model and the methods that are used to grow and prune the tree.

Figure 34.2 Model Information

The TREESPLIT Procedure

Model Information	
Split Criterion	Entropy
Pruning Method	Cost Complexity
Max Branches per Node	2
Max Tree Depth	10
Tree Depth Before Pruning	3
Tree Depth After Pruning	2
Number of Leaves Before Pruning	7
Number of Leaves After Pruning	3

The "Observation Information" table in Figure 34.3 provides the numbers of observations that are read and used. These numbers are the same in this example because there are no missing values in the predictor variables and the ASSIGNMISSING= option is not set to NONE.

Figure 34.3 Observation Information

	Training
Number of Observations Read	178
Number of Observations Used	178

The plot in Figure 34.4 is a tool for selecting the tuning parameter for cost-complexity pruning. The parameter (indicated on the lower horizontal axis) indexes a sequence of progressively smaller subtrees that are nested within the large tree. The parameter value 0 corresponds to the fully grown tree, and positive values control the trade-off between complexity (number of leaves) and fit to the training data, as measured by average misclassification rate.

Figure 34.4 Misclassification Rate as a Function of Cost Complexity

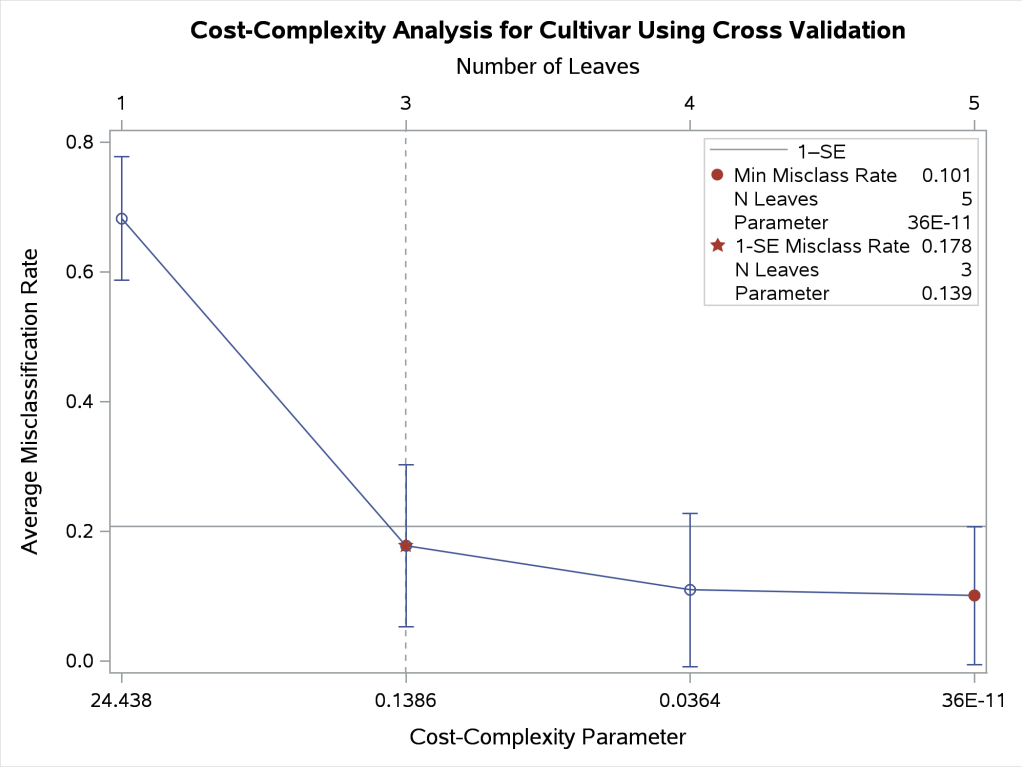
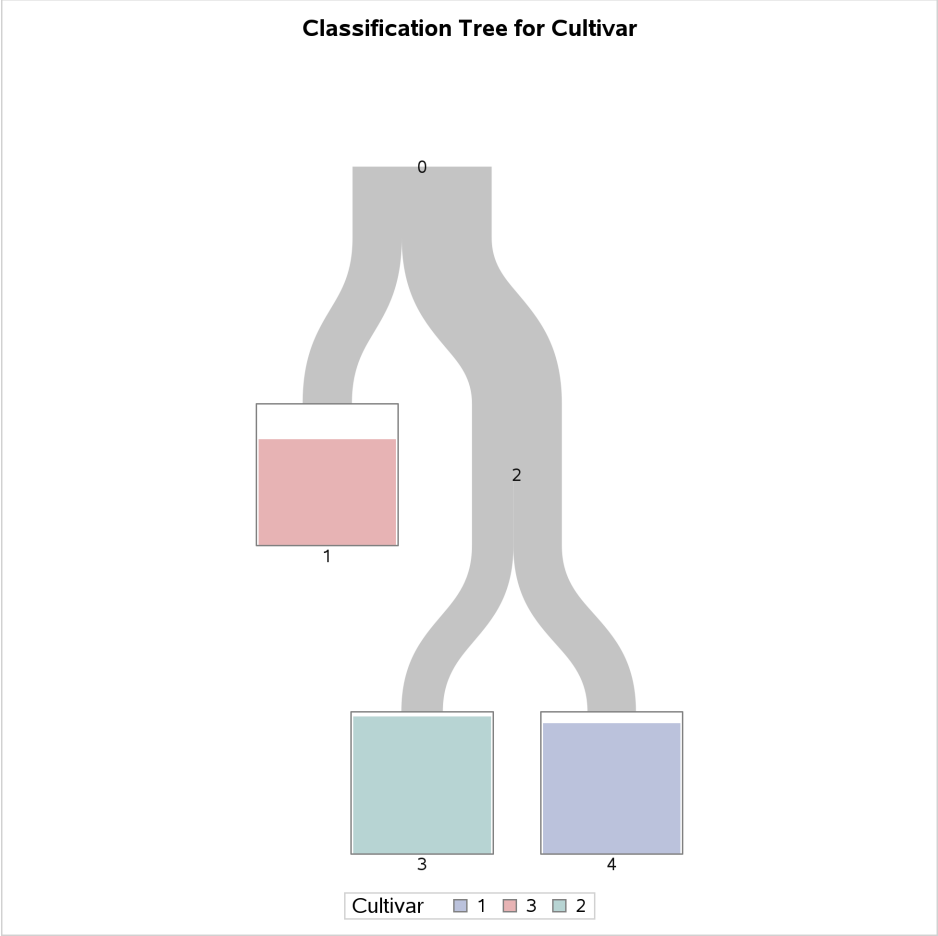


Figure 34.4 shows the minimum average misclassification rate, which is obtained by 10-fold cross validation. In the plot, this value is indicated by a filled-in circle. Information about the 1-SE misclassification rate is also included because the LEAVES=SE suboption was specified.

Breiman's 1-SE rule chooses the parameter that corresponds to the smallest subtree for which the misclassification rate is less than one standard error above the minimum misclassification rate (Breiman et al. 1984). The parameter value that corresponds to the 1-SE rule is indicated by a star. The dotted vertical line in Figure 34.4 indicates the chosen tree.

The tree diagram in Figure 34.5, which is produced by default when ODS Graphics is enabled, provides an overview of the tree as a classifier.

Figure 34.5 Overview Diagram of Final Tree

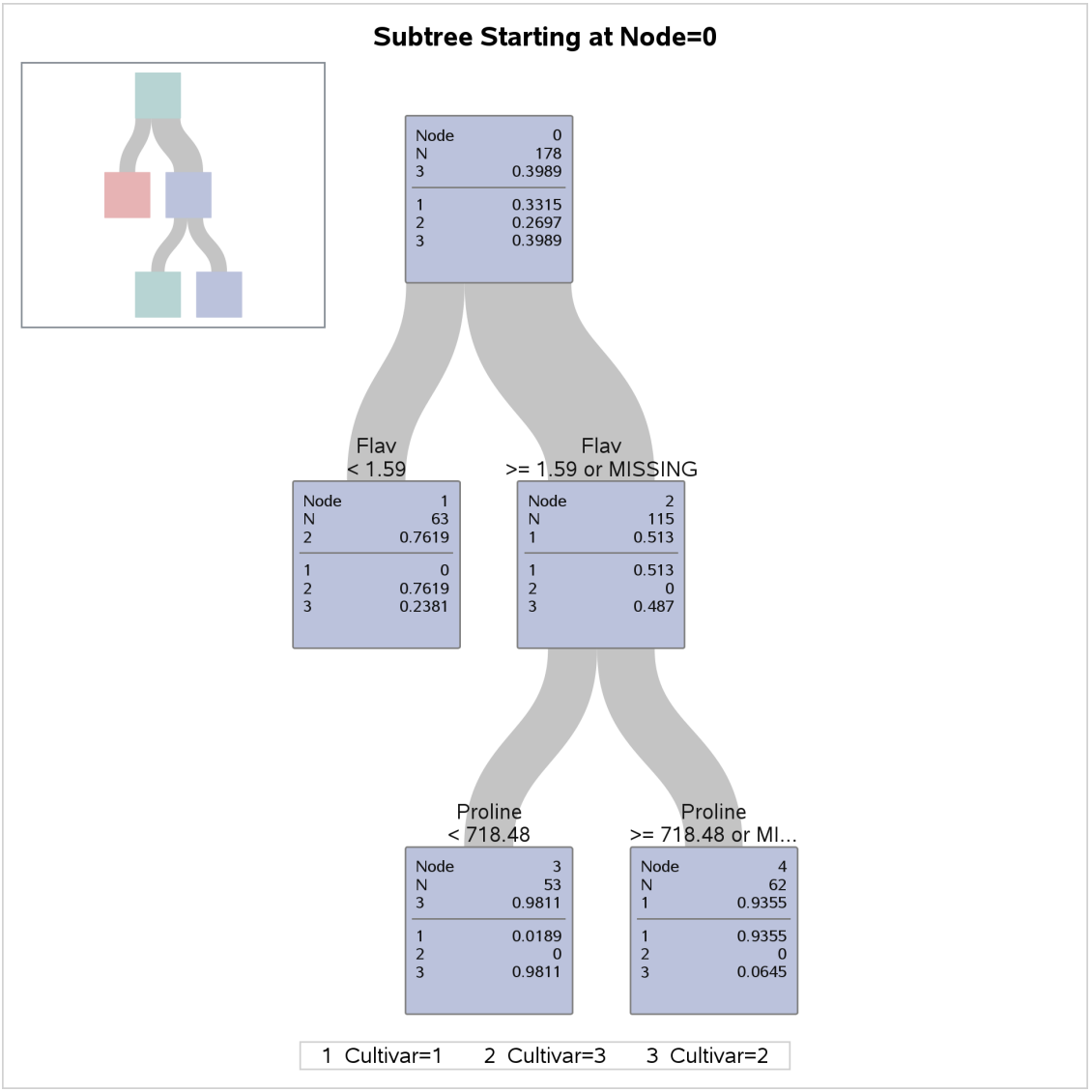


The tree is constructed by starting with all the observations in the root node (labeled 0). This node is split into two internal nodes (1 and 2, respectively). Node 1 is further split into leaf nodes (3 and 4), and node 2 is further split into leaf nodes (5 and 6).

The color of the bar for each leaf node indicates the most frequent level of Cultivar among the observations in that node; this is also the classification level that is assigned to all observations in that node. The height of the bar indicates the proportion of observations in the node that have the most frequent level. The width of the link between parent and child nodes is proportional to the number of observations in the child node.

The diagram in Figure 34.6 provides more detail about the nodes and splits.

Figure 34.6 Detailed Tree Diagram



The detailed tree diagram displays a box for each node; the box contains six lines of information, separated by a horizontal line. The proportion of each level of the predictor variable is shown below the horizontal line, and the level that has the highest proportion is also displayed above the horizontal line. Also displayed above the horizontal line are the node identifier and the number of observations that are assigned to the node.

The root node (node 0) contains 178 samples. Because the MAXBRANCH= option is not specified in the preceding statements, PROC TREESPLIT divides each node into two child nodes (MAXBRANCH=2 by default). At node 0, PROC TREESPLIT determines that the impurity of the root node is maximally decreased (as measured by the entropy criterion, which is the default) by splitting the 178 observations such that all samples for which Flav $\geq$ 1.59 are assigned to node 2 and all samples for which Flav < 1.59 are assigned to node 1. This is the splitting rule for node 0. In this training phase, 63 samples are assigned to node 1 and 115 samples are assigned to node 2.

Figure 34.6 also indicates that node 1 contains 0 samples with level 1. The legend shows that level 1 corresponds to a Cultivar value of 1, so node 1 contains no samples of the first cultivar. Similarly, the diagram

indicates that node 2 contains 0 samples with level 2, and the legend shows that level 2 corresponds to a Cultivar value of 3, so node 2 contains no samples of the third cultivar.

The splitting rule for node 1 consists of the variable Color and the split value 3.8032. All samples for which Color $\geq$ 3.8032 are assigned to node 4 (which then contains 49 samples), and all samples for which Color $<$ 3.8032 are assigned to node 3 (which then contains 14 samples).

The resulting classification tree yields simple rules for predicting the cultivar. For example, a sample for which Flav $<$ 1.59 and Color $<$ 3.8032 is predicted to be from the second cultivar (node 3 indicates that level 3 has the highest proportion of observations, and the legend shows that level 3 corresponds to a Cultivar value of 2).

Figure 34.6 displays the entire tree that begins with the root node and has a depth of three levels. You can use the PLOTS=ZOOMEDTREE option in the PROC TREESPLIT statement to request diagrams that begin with other nodes and have specified depths.

The table in Figure 34.7 displays fit statistics for the tree model.

Figure 34.7 Fit Statistics

The TREESPLIT Procedure

Fit Statistics for Selected Tree

	Number of Leaves	Misclassification Rate
Training	3	0.1124

The misclassification rate is the total proportion of the 178 wine samples that were misclassified. The following numbers of samples were misclassified in the terminal nodes:

- node 3: 0
- node 4: 1
- node 5: 1
- node 6: 4

So the total misclassification rate is $(0 + 1 + 1 + 4)/178 \approx 0.0337$.

Syntax: TREESPLIT Procedure

The following statements and options are available in the TREESPLIT procedure:

PROC TREESPLIT < *options* > ;
  AUTOTUNE < *options* > ;
  CLASS *variables* ;
  CODE < *options* > ;
  CROSSVALIDATION < *options* > ;
  FREQ *variable* ;
  GROW *criterion* < *options* > ;
  MITIGATEBIAS *mitigate-options* ;
  MODEL *response* **=** *variable. . .* ;
  OUTPUT OUT=*libref.data-table output-options* ;
  PARTITION < *partition-options* > ;
  PRUNE *prune-method* < **(***prune-options***)** > ;
  SAVESTATE RSTORE=*libref.data-table* ;
  VIICODE < *options* > ;
  WEIGHT *variable* ;

The PROC TREESPLIT statement and the MODEL statement are required. If any variables are character or are to be treated as categorical, at least one CLASS statement is required. Variables that appear after the equals sign (=) in the MODEL statement are predictor variables that model the *response* variable. By default, all *variables* that appear in the MODEL statement are treated as continuous variables. A CLASS statement causes a *variable* to be treated as categorical. Specifying a *variable* in a CLASS statement but not in a MODEL statement causes the *variable* to be implicitly added to the MODEL statement.

The following sections describe the PROC TREESPLIT statement and then describe the other statements in alphabetical order.

PROC TREESPLIT Statement

PROC TREESPLIT < *options* > ;

The PROC TREESPLIT statement invokes the procedure. Table 34.1 summarizes the options in this statement.

Table 34.1 PROC TREESPLIT Statement Options

Option	Description
Basic Options	
APPLYROWORDER	Specifies that the procedure use a prespecified data distribution
BINMETHOD=	Specifies how to bin interval inputs prior to training
CVCC	Requests a table of the results of the cost-complexity pruning based on cross validation
INMODEL=	Specifies a saved tree model to use to score a new table
NOPRINT	Suppresses ODS output
NTHREADS=	Specifies the number of threads to use in the computation

Table 34.1 *continued*

Option	Description
NUMBIN=	Specifies the number of bins to use for continuous variables
OUTMODEL=	Specifies the data table to which the decision tree model is to be saved
PLOTS=	Specifies options for plots
PRINTTARGET	Create tables that contain information about columns in the output
PRUNINGTABLE	Creates a table that contains the pruning information
RBAIMP	Creates a variable importance table by using random branch assignment
SEED=	Sets the seed for pseudorandom number generation
TREEPLOT	Creates a table that contains the information used to render the tree graphics
VII=	Calculates the importance of the specified types of variable interactions
Splitting Options	
ASSIGNMISSING=	Specifies how to handle missing values in a predictor variable
CLUSTERSPLIT	Requests that an alternative method of determining splits be used
MAXBRANCH=	Specifies the maximum number of child nodes per parent node
MAXDEPTH=	Specifies the maximum tree depth
MINLEAFSIZE=	Specifies the minimum number of observations per child node
MINUSEINSEARCH=	Specifies the minimum number of observations to use with the USEINSEARCH policy for handling missing values
NOMSEARCH=	Specifies the method to use to find a split on a nominal variable
SPLITONCE	Specifies that a variable can be used to split only once per branch

You can specify the following *options*:

APPLYROWORDER
uses a data distribution and row order as determined by a previous **`partition`** action call. For more information, see the section "The APPLYROWORDER Option" on page 107 in Chapter 2, "Shared Concepts."

ASSIGNMISSING=BRANCH | NONE | MACSMALL | POPULAR | SIMILAR | USEINSEARCH
specifies how to handle missing values of predictor variables during training and how to handle missing values and unknown levels of predictor variables. An unknown level of a categorical predictor variable is a level that does not exist in the training data but is encountered during scoring. During scoring, unknown levels are treated as missing values.

You can specify one of the following values:

BRANCH during the training phase, assigns any observation that has a missing value for the predictor variable to a specially created child node (branch). If all observations have nonmissing values for a predictor variable, then no branch is created to contain observations with missing values. In the scoring phase, if an observation has a missing value for a predictor variable and no special branch was created to contain observations with missing values, or if an observation has an unknown level for a

predictor variable, then the observation is assigned to the child node that contains the most training observations.

NONE during the training phase, excludes any observation that has a missing value for any predictor variable. In the scoring phase, if an observation has a missing value or an unknown level for a categorical predictor variable, then the observation is assigned to the child node that contains the most training observations. If an observation has a missing value for a continuous predictor, then the observation is assigned to the child node with the smallest numerical values of the predictor.

MACSMALL during the training phase, treats a missing value in a categorical predictor variable as a separate, legitimate value. If all observations have nonmissing values for a categorical predictor variable, then no branch is selected to contain observations with missing values. In the scoring phase, if an observation has a missing value for a categorical predictor variable and no branch is selected to contain observations with missing values, or if an observation has an unknown level for a predictor variable, then the observation is assigned to the child node that contains the most training observations.

In both the training and the scoring phases, missing values in continuous predictor variables are treated as the smallest possible value.

POPULAR during the training phase, assigns any observation that has a missing value in the predictor variable to the child node that has the most training observations. In the scoring phase, if an observation has a missing value or an unknown level for a predictor variable, then the observation is assigned to the child node that contains the most training observations.

SIMILAR during the training phase, assigns any observation that has a missing value in the predictor variable to the child node whose observations are most similar to it. This similarity is determined using the chi-square criterion for categorical responses or the F-test criterion for continuous responses. If all observations have nonmissing values for a predictor variable, then no branch is selected to contain observations with missing values. In the scoring phase, if an observation has a missing value for a predictor variable and no branch is selected to contain observations with missing values, or if an observation has an unknown level for a predictor variable, then the observation is assigned to the child node that contains the most training observations.

USEINSEARCH during the training phase, treats a missing value in a predictor variable as a separate, legitimate value. If all observations have nonmissing values for a predictor variable, then no branch is selected to contain observations with missing values. In the scoring phase, if an observation has a missing value for a predictor variable and no branch is selected to contain observations with missing values, or if an observation has an unknown level for a predictor variable, then the observation is assigned to the child node that contains the most training observations.

By default, ASSIGNMISSING=USEINSEARCH.

BINMETHOD=BUCKET | QUANTILE

specifies how to bin interval input variables prior to growing the decision tree. The number of bins that are created is determined by the NUMBIN= option.

You can specify one of the following values:

BUCKET bins interval input variables into fixed-width bins. The width of each bin for a particular variable is calculated by subtracting the smallest value among all observations from the largest value among all observations, and then dividing that result by the number of bins.

QUANTILE bins interval input variables into bins according to their quantile. The width of the bins for a particular variable is not fixed, but the number of observations in each bin is approximately equal.

By default, BINMETHOD=QUANTILE.

CLUSTERSPLIT

determines the splits at each node using clustering on each input variable, and then chooses the splitting variable on the basis of which variable and split optimize the criterion that is specified in the GROW statement.

CVCC

CVCOSTCOMPLEXITY

requests a table of the results of cost-complexity pruning based on cross validation. For each penalty parameter in the cross validation, the table provides the penalty parameter, the minimum, the maximum, and the average error. The error is the misclassification rate when the response variable is categorical and is the average square error (ASE) when the response variable is continuous. You can use the PLOTS=CVCC option to request a plot of the information in this table.

DATA=*libref.data-table*

names the input data table for PROC TREESPLIT to use. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to the data, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 1783.

data-table specifies the name of the input data table.

INMODEL=< *libref.***>***data-table*

specifies the data table that you have previously saved as a tree model by using the OUTMODEL= option in a previous run of PROC TREESPLIT. *libref.data-table* is a two-level name, where *libref* refers to the library, and *data-table* specifies the name of the input data table. For more information about this two-level name, see the DATA= option and the section "Using CAS Sessions and CAS Engine Librefs" on page 1783.

When you use the INMODEL= option, the OUTPUT statement is required and any other options in the PROC TREESPLIT statement, except for NOPRINT and PLOTS, are ignored.

The data table used in the INMODEL= option must include the attributes that are associated with the table produced by the OUTMODEL= option, or PROC TREESPLIT issues an error.

MAXBRANCH=*b*

specifies the maximum number of child nodes per parent node in the tree. PROC TREESPLIT tries to create this number of children unless it is impossible (for example, if a split variable does not have enough levels).

By default, MAXBRANCH=2.

MAXDEPTH=*number*

specifies the maximum depth of the tree to be grown. The number of levels in a tree is equal to the depth plus 1. The default is calculated by the following equation, where b is the value of the MAXBRANCH= option in the PROC TREESPLIT statement:

$$\text{MaxDepth} = \left\lceil \frac{10}{\log_2(b)} \right\rceil$$

This value can be tuned with the AUTOTUNE statement. In the ODS output that corresponds to tables that are generated using the AUTOTUNE statement, the term "MAXLEVEL" is used. The value of MAXLEVEL is equal to the value of the MAXDEPTH= option plus 1.

MINLEAFSIZE=*number*

LEAFSIZE=*number*

specifies the minimum *number* of observations in the training data that each child of a split must contain in order for the split to be considered.

By default, MINLEAFSIZE=5. This value can be tuned with the AUTOTUNE statement.

MINUSEINSEARCH=*number*

specifies a threshold for using missing values in the split search when ASSIGNMISSING=USEINSEARCH as the missing value policy. If the number of observations that have missing values for the splitting variable is greater than or equal to *number*, then PROC TREESPLIT uses the USEINSEARCH policy for missing values.

By default, MINUSERINSEARCH=1.

NOMSEARCH(*suboption* **...)**

specifies search methods for splitting on a nominal variable. For more information, see the section "Splitting Nominal Values" on page 1831.

If you do not specify this option, the default values of the *suboptions* determine the search method. You can specify one or more of the following *suboptions*:

MAXCATEGORIES=*number*

specifies the maximum number of categories to use in a splitting rule. The input is eligible for splitting nodes in which no more than *number* categories appear. The value of *number* must be greater than 1.

By default, MAXCATEGORIES=128.

SHRINKAGE=*number*
specifies how much weight to give the average gradient when you combine it with the average gradient within a category. You use this option only when sorting the categories. The value of *number* must be nonnegative.

By default, SHRINKAGE=10.

SORT=*number*
specifies the minimum cardinality in the node of the nominal variable for using the sort method. The sort method sorts the categories and then examines binary splits that preserve the ordering. The value of *number* must be a nonnegative integer. If *number* is greater than the value of the MAXCATEGORIES=*number* option, then the variable is not used in the node.

By default, SORT=10.

NOPRINT
suppresses the generation of ODS output.

NTHREADS=*number-of-threads*
specifies the number of threads to use in the computation. The default value is the number of CPUs available on the machine.

NUMBIN=*number*
NBINS=*number*
specifies the *number* of bins to use for binning interval predictor variables. PROC TREESPLIT bins continuous predictors to a fixed bin size. This option controls the number of bins and thereby also the size of the bins.

By default, NUMBIN=50. This value can be tuned with the AUTOTUNE statement.

OUTMODEL=*libref.data-table*
specifies the data table to which to save the decision tree model. *libref.data-table* is a two-level name, where *libref* refers to the library, and *data-table* specifies the name of the output data table. For more information about this two-level name, see the DATA= option and the section "Using CAS Sessions and CAS Engine Librefs" on page 1783.

The data table that results from this option contains information about each node in the decision tree model, including the splitting variables, the child nodes, the number of observations at each node, and the predicted response at each node. This table also has attributes attached that contain information about the input variables, the target variable, and the decision tree model.

PLOTS <(*global-plot-option***)> <= *plot-request* <(***options***)>>**
PLOTS <(*global-plot-option***)> <= (***plot-request* <(***options***)> <... *plot-request* <(***options***)>>)>**
controls the plots that are produced through ODS Graphics. When you specify only one *plot-request*, you can omit the parentheses around it.

You can specify the following *global-plot-option*:

ONLY
suppresses the default plots. Only plots that you specifically request are displayed.

You can specify the following *plot-requests*:

ALL
produces all appropriate plots.

CVCC
produces a plot that is used to determine the tuning parameter for cost-complexity pruning when *k*-fold cross validation is used. This plot displays the average misclassification rate when the response variable is a classification variable and displays the average square error (ASE) when the response variable is a continuous variable.

NONE
suppresses the default plots. Only plots that you specifically request are displayed.

PRUNEUNTIL
plots the metric that is used to select the final subtree.

This option is enabled by default except when you specify the OFF option in the PRUNE statement (which turns off pruning completely). Specifying the PRUNEUNTIL option has no effect if you also specify the OFF option in the prune statement.

WHOLETREE < (*whole-tree-options*) >
produces a plot to visualize the entire finished (grown and pruned) tree. This option is enabled by default.

You can specify the following values:

LINKSTYLE=CURVED | ORTHOGONAL | STRAIGHT
specifies the style of links between nodes and leaves in the tree. You can specify the following *link-styles*:

CURVED requests curved links between the nodes and their children.

ORTHOGONAL requests that links go straight down partway from a node to its children, create a horizontal line at the base of the vertical line, and then go straight down from that line to each child.

STRAIGHT requests that links go straight from the nodes to their children.

By default, LINKSTYLE=CURVED.

LINKWIDTH=CONSTANT | PROPORTIONAL
specifies the width of links between nodes and leaves in the tree. You can specify the following values:

CONSTANT requests that all links have the same thickness.

PROPORTIONAL requests that links have a thickness proportional to the total number of observations that go between the node and each child.

By default, LINKWIDTH=PROPORTIONAL.

NOLEGEND
turns off the legend.

ZOOMEDTREE < (*zoomed-tree-options*) >
produces a plot to visualize a portion of the finished (grown and pruned) tree. This option is enabled by default.

You can specify the following values:

DEPTH=*depth*
creates a plot down to *depth* for each *node-id* specified in the NODES= option.

By default, DEPTH=2.

LINKSTYLE=CURVED | ORTHOGONAL | STRAIGHT
specifies the style of links between nodes and leaves in the tree. You can specify the following values:

CURVED requests curved links between the nodes and their children.

ORTHOGONAL requests that links go straight down partway from a node to its children, create a horizontal line at the base of the vertical line, and then go straight down from that line to each child.

STRAIGHT requests that links go straight from the nodes to their children.

By default, LINKSTYLE=CURVED.

LINKWIDTH=CONSTANT | PROPORTIONAL
specifies the width of links between nodes and leaves in the tree. You can specify the following values:

CONSTANT requests that all links have the same thickness.

PROPORTIONAL requests that links have a thickness proportional to the total number of observations that go between the node and each child.

By default, LINKWIDTH=PROPORTIONAL.

NODES=(*node-id* **<** *node-id* **< . . . > >)** requests a plot for a subtree that is rooted at each node specified by a *node-id*. The values of *node-id* are alphanumeric strings that are displayed within the nodes in the plot that is created by the WHOLETREE option. PROC TREESPLIT creates one plot for each *node-id* that you specify. The default node ID is "0," the root of the entire tree.

NOLEGEND
suppresses the display of the legend

PRINTTARGET
outputs tables that indicate generated columns in the OUT= table from the OUTPUT statement. For a continuous response, PROC TREESPLIT generates an output table named PredName, which indicates the name of the predicted value column. For a categorical response, PROC TREESPLIT generates an output table named PredIntoName, which indicates the name of the predicted value column, and also an output table named PredProbName, which indicates the names of the predicted probability columns.

PRUNINGTABLE
outputs a table of the pruning results.

RBAIMP
creates a variable importance table by using random branch assignment (RBA). This table is created in addition to the normal variable importance table that is calculated using the residual sum of squares (RSS) error. For more information about RBA and RSS variable importance, see the section "Variable Importance" on page 1839.

SEED=*number*
specifies the initial seed for random number generation. The seed is used in the AUTOTUNE statement, in the CROSSVALIDATION statement, and for *k*-fold cross validation pruning. The value of *number* must be an integer. By default, the seed is generated by reading the time of day from the computer's clock.

SPLITONCE
splits predictor variables only once on a branch. When you specify this option, a variable cannot be split more than once on the path from the root to any particular terminal node. However, a variable can be used more than once across branches.

TREEPLOT
creates a table that contains the information used to create the WholeTree and ZoomedTree ODS graphics.

VII=*n*

INTERACTIONIMP=*n*
calculates the variable interaction importance, which is described in the section "Variable Interaction Importance" on page 1840. *n* can be one of the following values:

0 suppresses the calculation of variable importance.

1 calculates the importance of individual variables without interactions.

2 calculates the importance of two-way interactions and of individual variables.

3 calculates the importance of three-way and two-way interactions and of individual variables.

If you do not specify this option, then only the importance of individual variables without interactions is calculated.

AUTOTUNE Statement

AUTOTUNE < *options* > **;**

The AUTOTUNE statement searches for the best combination of values of the MAXDEPTH=, MINLEAFSIZE=, and NUMBIN= options in the PROC TREESPLIT statement and the criterion in the GROW statement. If you specify the AUTOTUNE statement, then you cannot also specify *k*-fold cross validation with cost-complexity pruning. You cannot specify both the AUTOTUNE statement and the CROSSVALIDATION statement in the same run of PROC TREESPLIT.

Table 34.2 summarizes the *options* that you can specify in the AUTOTUNE statement.

Table 34.2 AUTOTUNE Statement Options

Option	Description
APPENDLOOKUP	Specifies that the table specified in the HISTORYTABLE= option contain the rows from the table specified in the LOOKUPTABLE= option
CONSTRAINTS=	Specifies the constraints on the output metrics
EVALHISTORY=	Specifies how to report the evaluation history of the tuner
FOLDCOLUMN=	Specifies the column in the data table in which the cross validation fold value is indicated
FRACTION=	Specifies the fraction of observations to use for validation
HISTORYTABLE=	Specifies the CAS table that contains the evaluation history
KFOLD=	Specifies the number of folds for *k*-fold cross validation
LIVEUPDATE	Specifies that the table specified in the HISTORYTABLE= option be updated at every evaluation
LOOKUPTABLE=	Specifies the CAS table to use for evaluation lookup
MAXBAYES=	Specifies the maximum number of points in the kriging model
MAXEVALS=	Specifies the maximum number of evaluations
MAXITER=	Specifies the maximum number of iterations when SEARCHMETHOD=GA or SEARCHMETHOD=BAYESIAN
MAXTIME=	Specifies the maximum time for all iterations
MAXTRAINTIME=	Specifies the maximum time for a model training
NCONVITER=	Specifies the number of convergence iterations
NOGRIDSHUFFLE	Requests that the grid points not be shuffled
NOLOCALSEARCH	Disables local search optimization
NPARALLEL=	Specifies the number of parallel sessions
NSUBSESSIONWORKERS=	Specifies the number of workers in parallel sessions
OBJECTIVE=	Specifies the objective function
POPSIZE=	Specifies the population size when SEARCHMETHOD=GA or SEARCHMETHOD=BAYESIAN
SAMPLESIZE=	Specifies the sample size when SEARCHMETHOD=LHS or SEARCHMETHOD=RANDOM
SEARCHMETHOD=	Specifies the search method that the optimizer uses
SECONDOBJECTIVE=	Specifies the second objective to use for tuning
SELECTINITPOINT	Specifies that the tuner select the best evaluation from the lookup table

Table 34.2 *continued*

Option	Description
TARGETEVENT=	Specifies the target event for ROC-based calculations
TRAINFRACTION=	Specifies the fraction of observations to use for training
TUNINGPARAMETERS=	Specifies the custom tuning parameters
USEPARAMETERS=	Specifies how to handle the TUNINGPARAMETERS= option

APPENDLOOKUP
specifies that the rows from the table specified in the LOOKUPTABLE= option be appended to the output table that is created using the HISTORYTABLE= option.

This option is ignored if you do not specify both the HISTORYTABLE= and LOOKUPTABLE= options.

CONSTRAINTS=(*suboption* **| . . . | <** *suboption* **>)**
specifies constraints on output metrics that the tuner will use during tuning.

You can specify one or more of the following *suboptions*:

ASE (LB=*number* **UB=***number***)**
specifies a constraint on the average square error to be used while tuning the model.

You must specify one or both of the following suboptions:

LB=*number*
specifies the lower bound for the average square error.

UB=*number*
specifies the upper bound for the average square error.

F05 (LB=*number* **UB=***number***)**
specifies a constraint on the F0.5 coefficient to be used while tuning the model.

You must specify one or both of the following suboptions:

LB=*number*
specifies the lower bound for the F0.5 coefficient.

UB=*number*
specifies the upper bound for the F0.5 coefficient.

F1 (LB=*number* **UB=***number***)**
specifies a constraint on the F1 coefficient to be used while tuning the model.

You must specify one or both of the following suboptions:

LB=*number*
specifies the lower bound for the F1 coefficient.

UB=*number*
specifies the upper bound for the F1 coefficient.

FNR (LB=*number* **UB=***number***)**
specifies a constraint on the false negative rate to be used while tuning the model.

You must specify one or both of the following suboptions:

LB=*number*
specifies the lower bound for the false negative rate.

UB=*number*
specifies the upper bound for the false negative rate.

FPR (LB=*number* **UB=***number***)**
specifies a constraint on the false positive rate to be used while tuning the model.

You must specify one or both of the following suboptions:

LB=*number*
specifies the lower bound for the false positive rate.

UB=*number*
specifies the upper bound for the false positive rate.

GAMMA (LB=*number* **UB=***number***)**
specifies a constraint on the gamma coefficient to be used while tuning the model.

You must specify one or both of the following suboptions:

LB=*number*
specifies the lower bound for the gamma coefficient.

UB=*number*
specifies the upper bound for the gamma coefficient.

GINI (LB=*number* **UB=***number***)**
specifies a constraint on the Gini coefficient to be used while tuning the model.

You must specify one or both of the following suboptions:

LB=*number*
specifies the lower bound for the Gini coefficient.

UB=*number*
specifies the upper bound for the Gini coefficient.

KS (LB=*number* **UB=***number***)**
specifies a constraint on the Kolmogorov-Smirnov coefficient to be used while tuning the model.

You must specify one or both of the following suboptions:

LB=*number*
specifies the lower bound for the Kolmogorov-Smirnov coefficient.

UB=*number*
specifies the upper bound for the Kolmogorov-Smirnov coefficient.

MAE (LB=*number* **UB=***number***)**
specifies a constraint on the mean absolute error to be used while tuning the model.

You must specify one or both of the following suboptions:

LB=*number*
specifies the lower bound for the mean absolute error.

UB=*number*
specifies the upper bound for the mean absolute error.

MCE (LB=*number* **UB=***number***)**
specifies a constraint on the misclassification rate to be used while tuning the model.

You must specify one or both of the following suboptions:

LB=*number*
specifies the lower bound for the misclassification rate.

UB=*number*
specifies the upper bound for the misclassification rate.

MCLL (LB=*number* **UB=***number***)**
specifies a constraint on the multiclass log loss to be used while tuning the model.

You must specify one or both of the following suboptions:

LB=*number*
specifies the lower bound for the multiclass log loss.

UB=*number*
specifies the upper bound for the multiclass log loss.

MISC (LB=*number* **UB=***number***)**
specifies a constraint on the misclassification error percentage to be used while tuning the model.

You must specify one or both of the following suboptions:

LB=*number*
specifies the lower bound for the misclassification error percentage.

UB=*number*
specifies the upper bound for the misclassification error percentage.

MSE (LB=*number* **UB=***number***)**
specifies a constraint on the mean square error to be used while tuning the model.

You must specify one or both of the following suboptions:

LB=*number*
specifies the lower bound for the mean square error.

UB=*number*
specifies the upper bound for the mean square error.

MSLE (LB=*number* **UB=***number***)**
specifies a constraint on the mean square logarithmic error to be used while tuning the model.

You must specify one or both of the following suboptions:

LB=*number*
specifies the lower bound for the mean square logarithmic error.

UB=*number*
specifies the upper bound for the mean square logarithmic error.

RASE (LB=*number* **UB=***number***)**
specifies a constraint on the root average square error to be used while tuning the model.

You must specify one or both of the following suboptions:

LB=*number*
specifies the lower bound for the root average square error.

UB=*number*
specifies the upper bound for the root average square error.

RMAE (LB=*number* **UB=***number***)**
specifies a constraint on the root mean absolute error to be used while tuning the model.

You must specify one or both of the following suboptions:

LB=*number*
specifies the lower bound for the root mean absolute error.

UB=*number*
specifies the upper bound for the root mean absolute error.

RMSLE (LB=*number* **UB=***number***)**
specifies a constraint on the root mean square logarithmic error to be used while tuning the model.

You must specify one or both of the following suboptions:

LB=*number*
specifies the lower bound for the root mean square logarithmic error.

UB=*number*
specifies the upper bound for the root mean square logarithmic error.

SCORETIME (LB=*number* **UB=***number***)**

SCORINGTIME (LB=*number* **UB=***number***)**
specifies a constraint on the model scoring time to be used while tuning the model.

You must specify one or both of the following suboptions:

LB=*number*
specifies the lower bound for the model scoring time.

UB=*number*
specifies the upper bound for the model scoring time.

TAU (LB=*number* **UB=***number***)**
specifies a constraint on the tau coefficient to be used while tuning the model.

You must specify one or both of the following suboptions:

LB=*number*
specifies the lower bound for the tau coefficient.

UB=*number*
specifies the upper bound for the tau coefficient.

TNR (LB=*number* **UB=***number***)**
specifies a constraint on the true negative rate to be used while tuning the model.

You must specify one or both of the following suboptions:

LB=*number*
specifies the lower bound for the true negative rate.

UB=*number*
specifies the upper bound for the true negative rate.

TPR (LB=*number* **UB=***number***)**
specifies a constraint on the true positive rate to be used while tuning the model.

You must specify one or both of the following suboptions:

LB=*number*
specifies the lower bound for the true positive rate.

UB=*number*
specifies the upper bound for the true positive rate.

TRAINTIME (LB=*number* **UB=***number***)**
TRAININGTIME (LB=*number* **UB=***number***)**
specifies a constraint on the model training time to be used while tuning the model.

You must specify one or both of the following suboptions:

LB=*number*
specifies the lower bound for the model training time.

UB=*number*
specifies the upper bound for the model training time.

EVALHISTORY=ALL | LOG | NONE | TABLE
specifies how to report the evaluation history of the tuner.

You can specify one of the following values:

ALL	reports each evaluation in the log and creates the EvaluationHistory ODS table.
LOG	prints the following information to the log for each evaluation: evaluation number, objective value, best objective value up to that point, evaluation time, and elapsed time since the beginning of the tuning process.
NONE	suppresses reporting of evaluations in the log and does not create the Evaluation-History ODS table.
TABLE	creates the EvaluationHistory ODS table, which contains all evaluated points. The table contains columns for the evaluation number, all tuning parameters, and the objective function value.

By default, EVALHISTORY=TABLE.

FOLDCOLUMN=*string*
specifies the name of the column in the data table in which the cross validation fold value is indicated for all observations. This column must contain integers between 1 and *number*, where *number* is the desired number of folds. This option has an effect similar to that of the KFOLD=*number* option. If you specify this option, the tuner uses the cross validation process to estimate the accuracy of the models. But instead of dividing the data into multiple folds (partitions), as the KFOLD=*number* option does, the tuner reads the fold numbers from the specified column in the data table and assigns each observation to the specified fold. In cross validation, each model evaluation requires *number* of training executions (on *number*–1 data folds) and *number* of scoring executions (on one holdout fold). Thus, the evaluation time is increased by approximately a factor of *number*. For small to medium data tables or for unbalanced data tables, cross validation typically provides a better representation of error across the entire data table (a better generalization error).

If you specify both this option and the KFOLD= option, the value of the KFOLD= option must match the number of folds in the fold column. If you omit this option, the KFOLD= option, and the

FRACTION= option, then the default value of FRACTION=0.3 is used. If you specify a PARTITION statement, the validation partition defined in that statement is used, and the FOLDCOLUMN= option is ignored. You cannot specify the FOLDCOLUMN= option in combination with the FRACTION= option.

FRACTION=*number*

specifies the fraction of all data to use for validation, where *number* must be between 0.01 and 0.99, inclusive. If you specify this option, the tuner uses a single-partition validation for finding the objective value (validation error estimate). Using this option might not be advisable for small or unbalanced data tables, where the random assignment of the validation subset might not provide a good estimate of error. For large, balanced data tables, a single-partition validation is usually sufficient for estimating error; a single partition is more efficient than cross validation in terms of the total execution time.

If you specify a PARTITION statement, the validation partition defined in that statement is used, and this option is ignored. You cannot specify this option in combination with the KFOLD= option.

If you specify the TRAINFRACTION= option, then by default the value of the FRACTION= option is one minus the value of the TRAINFRACTION= option. Otherwise, by default, FRACTION=0.3.

HISTORYTABLE=*libref.data-table* **<(***data-set-options***)>**

specifies the data table created by the tuning process that contains the evaluation history. *libref.data-table* is a two-level name, where *libref* refers to the library, and *data-table* specifies the name of the output data table. For more information about this two-level name, see the section "Using CAS Sessions and CAS Engine Librefs" on page 1783.

The *data-set-options* are various options that enable you to control how the data table is created. For more information about data set options, see *SAS Data Set Options: Reference*. In the case of the HISTORYTABLE= option, the default value of the REPLACE= data set option is NO. The REPLACE= option specifies whether a new SAS data set can overwrite an existing data set that has the same name. You can override the default value by specifying REPLACE=YES.

KFOLD=*number*

specifies the number of folds (partitions) in the cross validation process, where *number* must be between 2 and 20, inclusive. If you specify this option, the tuner uses cross validation to find the objective value. In cross validation, each model evaluation requires *number* of training executions (on *number*–1 data folds) and *number* of scoring executions (on one holdout fold). Thus, the evaluation time is increased by approximately a factor of *number*. For small to medium data tables or for unbalanced data tables, cross validation typically provides a better representation of error across the entire data table (a better generalization error).

Another option that triggers cross validation is the FOLDCOLUMN= option. If you specify both the KFOLD= option and the FOLDCOLUMN= option, the value of the KFOLD= option must match the number of folds in the fold column.

If you omit the KFOLD= option, the FOLDCOLUMN= option, and the FRACTION= option, then the default value of FRACTION=0.3 is used. If you specify a PARTITION statement, the validation partition defined in that statement is used, and the KFOLD= option is ignored. You cannot specify the KFOLD= option in combination with the FRACTION= option.

LIVEUPDATE
specifies that the table specified in the HISTORYTABLE= option be updated at every evaluation. This option forces the table to be a promoted CAS table so that it can be seen by other sessions.

This option is ignored if you do not specify the HISTORYTABLE= option.

LOOKUPTABLE=*libref.data-table*
specifies the data table previously created by the tuning process that you want to use as the lookup table. *libref.data-table* is a two-level name, where *libref* refers to the library, and *data-table* specifies the name of the input data table. For more information about this two-level name, see the section "Using CAS Sessions and CAS Engine Librefs" on page 1783.

MAXBAYES=*number*
specifies the maximum number of points in the Kriging model. This option is honored only when SEARCHMETHOD=BAYESIAN and when *number* has a minimum value of 10.

By default, MAXBAYES=100.

MAXEVALS=*number*
specifies the maximum number of configuration evaluations allowed for the tuner, where *number* must be an integer greater than or equal to 3. When the *number* of evaluations is reached, the tuner terminates the search and returns the results. To produce a single objective function value (validation error estimate), each configuration evaluation requires either a single model training and scoring execution on a validation partition, or a number of training and scoring executions equal to the value of the KFOLD= option for cross validation. The MAXEVALS= option might lead to termination before the value of the MAXITER= option or the MAXTIME= option is reached.

By default, MAXEVALS=50.

MAXITER=*number*
specifies the maximum number of iterations of the optimization tuner, where *number* must be greater than or equal to 1. Each iteration normally involves a number of objective evaluations up to the value of the POPSIZE= option. The MAXITER= option is used only when SEARCHMETHOD=GA or SEARCHMETHOD=BAYESIAN; it is ignored when SEARCHMETHOD=RANDOM or SEARCHMETHOD=LHS. The MAXITER= option might lead to termination before the value of the MAXEVALS= option or the MAXTIME= option is reached.

By default, MAXITER=5.

MAXTIME=*number*
specifies the maximum time (in seconds) allowed for the tuner, where *number* must be greater than or equal to 1. When this value is reached, the tuner terminates the search and returns results. The actual run time for optimization might be longer because it includes the remaining time needed to finish the current evaluation. For long-running model training (large data tables), the actual run time might significantly exceed *number*. The MAXTIME= option might lead to termination before the value of the MAXEVALS= option or the MAXITER= option is reached.

By default, MAXTIME=36000.

MAXTRAINTIME=*number*
specifies the maximum time allowed for a single model train. The model train is terminated if it exceeds this time, and the objective value is set to missing.

By default, there is no maximum time allowed for a single train.

NCONVITER=*number*
specifies the *number* of convergence iterations after which tuning is terminated.

By default, NCONVITER=4.

NOGRIDSHUFFLE
requests that the points that are generated by the grid search method not be shuffled before execution.

NOLOCALSEARCH
disables local search optimization for the tuning process.

By default, local search optimization is enabled.

NPARALLEL=*number*
specifies the number of evaluations to be performed in parallel, where *number* must be greater than or equal to 0.

By default, NPARALLEL=0, which indicates that the value of *number* is determined as follows:

- If SEARCHMETHOD=GA or SEARCHMETHOD=BAYESIAN, then the number of parallel evaluations is equal to the value of the POPSIZE= option minus 1.
- If SEARCHMETHOD=LHS or SEARCHMETHOD=RANDOM, then the number of parallel evaluations is equal to the value of the SAMPLESIZE= option, with a maximum value of 32.

NSUBSESSIONWORKERS=*number*
specifies the number of workers to use in parallel subsessions. When alternative configurations are evaluated in parallel, a number of subsessions are created by the tuner, and each subsession uses multiple workers. By default, the number of workers that are used in a parallel subsession is determined as described in the section "Determining the Number of Parallel Evaluations" on page 1844.

OBJECTIVE=*function*
specifies which measure of model performance the tuner uses as the objective function.

You can specify one of the following values for *function*. Some values can be specified only when the target variable is of a particular type, as shown in parentheses.

ASE specifies average square error as the objective function.

AUC specifies area under the curve as the objective function (nominal type only).

F05 specifies the F0.5 coefficient as the objective function (nominal type only).

F1 specifies the F1 coefficient as the objective function (nominal type only).

FNR specifies the false negative rate as the objective function (nominal type only).

FPR specifies the false positive rate as the objective function (nominal type only).

GAMMA specifies the gamma coefficient as the objective function (nominal type only).

GINI specifies the Gini coefficient as the objective function (nominal type only).

KS specifies the Kolmogorov-Smirnov coefficient as the objective function (nominal type only).

MAE specifies the mean absolute error as the objective function (interval type only).

MCE specifies the misclassification rate as the objective function (nominal type only).

MCLL specifies the multiclass log loss as the objective function (nominal type only).

MISC specifies the misclassification error percentage as the objective function (nominal type only).

MSE specifies the mean square error as the objective function (interval type only).

MSLE specifies the mean square logarithmic error as the objective function (interval type only).

RASE specifies the root average square error as the objective function.

RMAE specifies the root mean absolute error as the objective function (interval type only).

RMSLE specifies the root mean square logarithmic error as the objective function (interval type only).

SCORETIME specifies the model scoring time as the objective function.

TAU specifies the tau coefficient as the objective function (nominal type only).

TNR specifies the true negative rate as the objective function (nominal type only).

TPR specifies the true positive rate as the objective function (nominal type only).

TRAINTIME specifies the model training time as the objective function.

By default, OBJECTIVE=MISC for nominal targets, and OBJECTIVE=MSE for interval targets.

POPSIZE=*number*

specifies the maximum population size, where population is the number of configuration evaluations in one iteration (population). The *number* must be greater than or equal to 1. In some cases, the tuner algorithm might generate a number of new configurations that is smaller than *number*. The POPSIZE= option is used only when SEARCHMETHOD=GA or SEARCHMETHOD=BAYESIAN; it is ignored when SEARCHMETHOD=RANDOM or SEARCHMETHOD=LHS.

By default, POPSIZE=10.

SAMPLESIZE=*number*

specifies the total number of evaluations, where *number* must be greater than or equal to 1. This option is only used when SEARCHMETHOD=RANDOM or SEARCHMETHOD=LHS; it is ignored when SEARCHMETHOD=GA or SEARCHMETHOD=BAYESIAN.

By default, SAMPLESIZE=50.

SEARCHMETHOD=BAYESIAN | GA | GRID | LHS | RANDOM

specifies the search method to use for tuning. You can specify the following values:

BAYESIAN builds a kriging surrogate model to approximate the objective value and uses this surrogate model to generate new alternative configurations at each iteration. The kriging model is continuously updated during the search process.

GA uses an initial Latin hypercube sample that seeds a genetic algorithm to generate a new population of alternative configurations at each iteration.

GRID uses all combinations of selected values of the hyperparameters.

LHS uses a Latin hypercube to generate a single sample of configurations that is uniform in each tuning parameter but random in combinations.

RANDOM generates a single sample of purely random configurations.

By default, SEARCHMETHOD=GA.

SECONDOBJECTIVE=*function*

specifies the type of the second objective to use for tuning.

You can specify one of the following values for *function*. Some values can be specified only when the target variable is of a particular type, as shown in parentheses.

NONE does not use a second objective for tuning.

ASE specifies average square error as the second objective.

AUC specifies area under the curve as the second objective (nominal type only).

F05 specifies the F0.5 coefficient as the second objective (nominal type only).

F1 specifies the F1 coefficient as the second objective (nominal type only).

FNR specifies the false negative rate as the second objective (nominal type only).

FPR specifies the false positive rate as the second objective (nominal type only).

GAMMA specifies the gamma coefficient as the second objective (nominal type only).

GINI specifies the Gini coefficient as the second objective (nominal type only).

KS specifies the Kolmogorov-Smirnov coefficient as the second objective (nominal type only).

MAE specifies the mean absolute error as the second objective (interval type only).

MCE specifies the misclassification rate as the second objective (nominal type only).

MCLL specifies the multiclass log loss as the second objective (nominal type only).

MISC specifies the misclassification error percentage as the second objective (nominal type only).

MSE specifies the mean square error as the second objective (interval type only).

MSLE specifies the mean square logarithmic error as the second objective (interval type only).

RASE specifies the root average square error as the second objective.

RMAE specifies the root mean absolute error as the second objective (interval type only).

RMSLE specifies the root mean square logarithmic error as the second objective (interval type only).

SCORETIME specifies the model scoring time as the second objective.

TAU specifies the tau coefficient as the second objective (nominal type only).

TNR specifies the true negative rate as the second objective (nominal type only).

TPR specifies the true positive rate as the second objective (nominal type only).

TRAINTIME specifies the model training time as the second objective.

By default, SECONDOBJECTIVE=NONE.

SELECTINITPOINT
specifies that the tuner select the best evaluation from the table specified in the LOOKUPTABLE= option as the initial point for the tuning process.

This option is ignored if you do not specify the LOOKUPTABLE= option.

TARGETEVENT=*string*
specifies the target event to use for calculating the selected objective function. This option is ignored when the value of the OBJECTIVE= option is not AUC, F1, F05, GINI, GAMMA, TAU, or KS.

If you omit the TARGETEVENT= option, the tuner selects one of the target levels and uses it to calculate the specified objective function.

TRAINFRACTION=*number*
specifies the fraction of all data to be used for training, where *number* must be between 0.01 and 0.99, inclusive. If you specify this option, the tuner uses a single-partition validation to find the objective value (validation error estimate). Using this option might not be advisable for small or unbalanced data tables, where the random assignment of the validation subset might not provide a good estimate of error. For large, balanced data tables, a single-partition validation is usually sufficient for estimating error; a single partition is more efficient than cross validation in terms of the total execution time.

If you specify a PARTITION statement, the validation partition defined in that statement is used, and this option is ignored. You cannot specify this option in combination with the KFOLD= option.

If you specify the FRACTION= option, then by default the value of the TRAINFRACTION= option is one minus the value of the FRACTION= option. Otherwise, by default, TRAINFRACTION=0.7.

TUNINGPARAMETERS=(*suboption* **| . . . | <** *suboption* **>)**
TUNEPARMS=(*suboption* **| . . . | <** *suboption* **>)**
specifies which parameters to tune and which ranges to tune over. If USEPARAMETERS=STANDARD, this option is ignored.

You can specify one or more of the following suboptions:

CRITERION (VALUES=*value-list* **INIT=***value* **EXCLUDE)**
specifies information about the splitting criteria to use for tuning the decision tree. For more information about the criteria, see the GROW statement.

You can specify the following additional suboptions:

VALUES=*value-list*
specifies a list of splitting criteria to consider during tuning, where *value-list* is a space separated list that can include one or more of the following values: CHAID, CHISQUARE, FTEST, GAIN, GINI, and VARIANCE.

INIT=*value*
specifies the initial splitting criterion for the tuner to use. You can specify the following *values*: CHAID, CHISQUARE, FTEST, GAIN, GINI, IGR, RSS, or VARIANCE.

By default, INIT=IGR for nominal targets and INIT=VARIANCE for interval targets.

EXCLUDE
excludes the splitting criterion from the tuning process.

MAXDEPTH (LB=*number* **UB=***number* **VALUES=***value-list* **INIT=***number* **EXCLUDE)**
specifies information about the maximum depth to grow the decision tree to use for tuning the decision tree. For more information, see the MAXDEPTH= option.

You can specify the following additional suboptions:

LB=*number*
specifies the minimum depth to consider during tuning. If you specify this suboption, you cannot specify the VALUES= suboption.

By default, LB=1.

UB=*number*
specifies the maximum depth to consider during tuning. If you specify this suboption, you cannot specify the VALUES= suboption.

By default, UB=19.

VALUES=*value-list*
specifies a list of depth values to consider during tuning, where *value-list* is a space-separated list of positive integers. If you specify this suboption, you cannot specify either the LB= or UB= suboption.

INIT=*number*
specifies the initial depth for the tuner to use.

By default, INIT=10.

EXCLUDE
excludes depth from the tuning process.

MINLEAFSIZE (LB=*number* **UB=***number* **VALUES=***value-list* **INIT=***number* **EXCLUDE)**
specifies information about tuning the leaf size option for training the decision tree. This option is not tuned by default. If it is specified without any suboptions, then tuning uses the following values: 1, 5, 10, 20, 40, 80, 160, 320. For more information, see the MINLEAFSIZE= option in the PROC TREESPLIT statement.

You can specify the following additional suboptions:

LB=*number*
specifies the minimum leaf size value to consider during tuning. If you specify this suboption, you cannot specify the VALUES= suboption.

By default, LB=1.

UB=*number*
specifies the maximum leaf size value to consider during tuning. If you specify this suboption, you cannot specify the VALUES= suboption.

By default, UB=320.

VALUES=*value-list*
specifies a list of leaf size values to consider during tuning, where *value-list* is a space-separated list of positive integers. If you specify this suboption, you cannot specify either the LB= or UB= suboption.

INIT=*number*
specifies the initial leaf size for the tuner to use.

By default, INIT=5.

EXCLUDE
excludes the leaf size from the tuning process.

NUMBIN (LB=*number* **UB=***number* **VALUES=***value-list* **INIT=***number* **EXCLUDE)**
specifies information about the number of bins in which to bin the interval inputs while tuning the decision tree. For more information, see the NUMBIN= option.

You can specify the following additional suboptions:

LB=*number*
specifies the minimum number of bins to consider during tuning. If you specify this suboption, you cannot specify the VALUES= suboption.

By default, LB=20.

UB=*number*
specifies the maximum number of bins to consider during tuning. If you specify this suboption, you cannot specify the VALUES= suboption.

By default, UB=200.

VALUES=*value-list*
specifies a list of numbers of bins to consider during tuning, where *value-list* is a space-separated list of positive integers. If you specify this suboption, you cannot specify either the LB= or UB= suboption.

INIT=*number*
specifies the initial number of bins for the tuner to use.

By default, INIT=20.

EXCLUDE
excludes the number of bins from the tuning process.

USEPARAMETERS=*tuning-parameter-option*
specifies which set of parameters to tune.

You can specify the following *tuning-parameter-options*:

STANDARD tunes using the default bounds and initial values for all parameters.

CUSTOM tunes only the parameters that are specified in the TUNINGPARAMETERS= option.

COMBINED tunes the parameters that are specified in the TUNINGPARAMETERS= option and uses default bounds and initial values to tune all other parameters.

CLASS Statement

CLASS *variables* **;**

The CLASS statement causes the specified *variables* to be treated as categorical variables in the analysis. These variables enter the analysis not through their values but through levels to which the unique values are mapped. For more information about these mappings, see the section "Levelization of Classification Variables" on page 62.

You can specify only one CLASS statement.

NOTE: All classification levels are padded or truncated to 32 characters.

CODE Statement

CODE < *options* > **;**

The CODE statement writes SAS DATA step code for computing predicted values of the fitted model to a file or to a table. To score new data, you can then include the file in a DATA step, or you can specify the table in the **runCodeTable** action in the **dataStep** action set (for more information, see *SAS Viya: System Programming Guide*).

Table 34.3 summarizes the *options* available in the CODE statement.

Table 34.3 CODE Statement Options

Option	Description
COMMENT	Adds comments to the generated code
FILE=	Names the file in which to save the generated code
FORMATWIDTH=	Specifies the numeric format width for the regression coefficients
INDENTSIZE=	Specifies the number of spaces to indent the generated code
LABELID=	Specifies a number used to construct names and labels
LINESIZE=	Specifies the line size for the generated code
NOTRIM	Compares formatted values, including blank padding
OUT=	Names an output table in which to save the generated code

For more information about the syntax of the CODE statement, see the section "CODE Statement" on page 13 in Chapter 2, "Shared Concepts."

CROSSVALIDATION Statement

CROSSVALIDATION < *options* > **;**

The CROSSVALIDATION statement performs *k*-fold cross validation to find the average estimated validation error. You cannot specify this statement if you specify either the AUTOTUNE statement or the PARTITION statement. **NOTE:** The CROSSVALIDATION statement does not support any pruning method. If you specify this statement, you must also disable pruning by specifying the OFF option in the PRUNE statement.

You can specify the following *options*:

KFOLD=*number*
specifies the number of partition folds in the cross validation process, where *number* is between 2 and 20, inclusive. By default, KFOLD=5.

NOPARALLEL
specifies that *k*-fold cross validation not be run in parallel. By default, the process runs in parallel.

NSUBSESSIONWORKERS=*number*
specifies the number of worker nodes to use in parallel subsessions. If you omit this option or if the *number* is 0, the number of worker nodes is determined automatically.

FREQ Statement

FREQ *variable* **;**

The *variable* in the FREQ statement identifies a numeric variable in the input data table that contains the frequency of occurrence of each observation. PROC TREESPLIT treats each observation as if it appears f times, where f is the value of the FREQ *variable* for the observation. If f is not an integer, it is truncated to an integer. If f is less than 1 or missing, the observation is not used in the analysis. When you omit the FREQ statement, each observation is assigned a frequency of 1.

GROW Statement

GROW *criterion* **<(***options***)> ;**

The GROW statement specifies the *criterion* by which to split a parent node into child nodes. As it grows the tree, PROC TREESPLIT calculates the specified *criterion* for each predictor variable and then splits on the predictor variable that optimizes the specified *criterion*.

For categorical responses, the available *criteria* are CHAID, CHISQUARE, ENTROPY, GINI, and IGR; the default is IGR. For continuous responses, the available *criteria* are CHAID, FTEST, and RSS; the default is RSS.

For either categorical or continuous responses, you can specify the following *criterion*:

CHAID <(*options***)>**
for categorical predictor variables, CHAID uses the value (as specified in the ALPHA= option) of a chi-square statistic (for a classification tree) or an F statistic (for a regression tree) to merge similar levels of the predictor variable until the number of children in the proposed split reaches the number that you specify in the MAXBRANCH= option. The p-values for the final split determine the variable on which to split.

For continuous predictor variables, CHAID chooses the best single split until the number of children in the proposed split reaches the value that you specify in the MAXBRANCH= option.

You can specify the following *options*:

ALPHA=*value*
specifies the maximum p-value for a split to be considered.

By default, ALPHA=0.3.

BONFERRONI
requests a Bonferroni adjustment to the p-value for a variable after the split has been determined.

By default, no adjustment is made.

For categorical responses only, you can specify the following *criteria*:

CHISQUARE < (*options*) >
uses a chi-square statistic to split each variable and then uses the *p*-values that correspond to the resulting splits to determine the splitting variable.

You can specify the following *options*:

ALPHA=*value*
specifies the maximum *p*-value for a split to be considered.

By default, ALPHA=0.3.

BONFERRONI
requests a Bonferroni adjustment to the *p*-value for a variable after the split has been determined.

By default, no adjustment is made.

ENTROPY < *option* >

GAIN < *option* >
uses the gain in information (decrease in entropy) to split each variable and then to determine the split. You can specify the following *option*:

MINENTROPY=*number*

MINGAIN=*number*
specifies the minimum gain value to validate a split.

GINI
uses the decrease in the Gini index to split each variable and then to determine the split.

IGR
uses the entropy metric to split each variable and then uses the information gain ratio to determine the split.

The default *criterion* for categorical responses is IGR.

For continuous responses only, you can specify the following *criteria*:

FTEST < (*options*) >
uses an *F* statistic to split each variable and then uses the resulting *p*-value to determine the split variable.

You can specify the following *options*:

ALPHA=*value*
specifies the maximum *p*-value for a split to be considered.

By default, ALPHA=0.3.

BONFERRONI
requests a Bonferroni adjustment to the *p*-value for a variable after the split has been determined.

By default, no adjustment is made.

RSS

VARIANCE

uses the change in response variance to split each variable and then to determine the split.

The default *criterion* for continuous responses is RSS.

You can tune the *criterion* by using the AUTOTUNE statement.

MITIGATEBIAS Statement

MITIGATEBIAS SENSITIVEVAR=*variable* **PREDICTEDVARS=(***variable-names***) PREDICTEDEVENTS=***"event-list"* < *options* > **;**

The MITIGATEBIAS statement enables PROC TREESPLIT to iteratively train a model while minimizing the bias metric that you specify. To use this statement, you must also specify the SAVESTATE statement and a categorical response (target) variable. You cannot also specify the AUTOTUNE statement, the CROSSVALIDATION statement, or the KFOLD= suboption in the PRUNE statement.

You must specify the following options:

PREDICTEDEVENTS="*event-list***"**

PEVENTS="*event-list***"**

specifies the events that correspond to each variable in the PREDICTEDVARS= option. You must specify the PREDICTEDEVENTS= and PREDICTEDVARS= options with the same number of items in the same order. Items in the *event-list* must be separated by the delimiter that you specify in the DELIMETER= option.

PREDICTEDVARS=(*variable-names***)**

PVARS=(*variable-names***)**

specifies the names of the variables that contain the posterior probability for each level in model prediction that corresponds to the response (target) variable. You must specify the PREDICTEDVARS= and PREDICTEDEVENTS= options with the same number of items in the same order. Duplicate variables are not allowed.

SENSITIVEVAR=*variable*

specifies the sensitive variable to use in order to reduce the value of the bias measurement that you specify in the BIASMETRIC= option.

You can also specify the following *options*:

BIASMETRIC=*bias-metric*

specifies the type of bias measurement. You can specify the following *bias-metrics*:

DEMOGRAPHICPARITY specifies demographic parity as the bias measurement.

EQUALIZEDODDS specifies equalized odds as the bias measurement.

EQUALOPPORTUNITY specifies equal opportunity as the bias measurement.

PREDICTIVEPARITY specifies predictive parity as the bias measurement.

By default, BIASMETRIC=DEMOGRAPHICPARITY.

BOUND=*number*
specifies the bound value for the exponentiated gradient reduction algorithm.

By default, BOUND=100.

DELIMITER="*character***"**
DLM="*character***"**
specifies the delimiter to be used to separate events that you specify in the PREDICTEDEVENTS= option. A delimiter is used when event names contain embedded spaces (such as "Fuel Oil") or special characters (such as ";" or "*"). You must use quotation marks around *character*. You can specify only one delimiter; combinations of delimiters are not supported. The valid values of the DELIMITER= option are " " (space), ";" (semicolon), "*" (asterisk), "." (period), and "," (comma).

By default, DELIMITER=" " (space).

If you specify a delimiter other than " " (space), PROC TREESPLIT treats any spaces (including empty spaces) as valid characters. You are responsible for mapping events correctly in order to obtain correct results.

LEARNINGRATE=*number*
specifies the step size to use in updating the exponentiated gradient reduction algorithm.

By default, LEARNINGRATE=0.01.

LOGLEVEL=0 | 1 | 2
specifies the level of log information to print.

0 prints only warnings and errors.

1 prints some notes.

2 prints notes for each iteration.

By default, LOGLEVEL=1.

MAXITER=*number*
specifies the maximum number of iterations to run the exponentiated gradient reduction algorithm.

By default, MAXITER=10.

SEED=*number*
specifies the seed for the pseudorandom number generator. By default, the random number stream is based on the time of day from the computer's clock. A 0 or negative seed value also results in random number streams that are based on the time of day from the computer's clock. If you want a reproducible random number sequence between runs, specify a value greater than 0.

By default, SEED=0.

TARGETEVENT="*event***"**
specifies the formatted value of the response (target) variable that represents the event of interest.

By default, *event* is the first level that you specify in the PREDICTEDEVENTS= option.

TOLERANCE=*number*
specifies the parity constraint violation tolerance. If you set *number* to 0, PROC TREESPLIT trains for the maximum number of iterations, which you specify in the MAXITER= option.

By default, TOLERANCE=0.005.

TUNEBOUND
specifies that the bound value must be tuned.

MODEL Statement

MODEL *response* **=** *variable* **;**

The MODEL statement causes PROC TREESPLIT to create a tree model by using *response* as the response variable and one or more *variables* as predictors. By default, *variables* are treated as a continuous predictors if they are numeric variables, or as categorical variables if they also appear in a CLASS statement.

NOTE: Specifying a character variable in a MODEL statement without previously declaring it in a CLASS statement results in an error.

OUTPUT Statement

OUTPUT OUT=*libref.data-table* < *option* > **;**

The OUTPUT statement creates an output data table that contains the results of PROC TREESPLIT.

You must specify the following argument:

OUT=*libref.data-table*
names the output data table for PROC TREESPLIT to use. You must specify this option before any other options. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to where the data table is to be stored, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 1783.

data-table specifies the name of the output data table.

You can also specify the following *options*:

COPYVAR=*variable*
COPYVARS=(*variables***)**
lists one or more variables from the input data table to be transferred to the output data table.

ROLE<=*name*>
generates a numeric variable that indicates the role played by each observation in fitting the model. By default, the variable is named _ROLE_. You can add an optional *=name* to change the name of this generated variable. For each observation, the interpretation of this variable is shown in Table 34.4.

Table 34.4 Role Interpretation

Value	Observation Role
0	Not used
1	Training
2	Validation
3	Testing

If you do not partition the input data by using a PARTITION statement, then the value of the role variable is 1 for all observations.

PARTITION Statement

PARTITION *partition-option* **;**

The PARTITION statement specifies how observations in the input data set are logically partitioned into disjoint subsets for model training, validation, and testing. For more information, see the section "Using Validation and Test Data" on page 101 in Chapter 2, "Shared Concepts." Either you can designate a variable in the input data table and a set of formatted values of that variable to determine the role of each observation, or you can specify proportions to use for randomly assigning observations to each role.

You must specify exactly one of the following *partition-options*:

FRACTION(<TEST=*fraction***> <VALIDATE=***fraction***> <SEED=***number***>)**
randomly assigns specified proportions of the observations in the input data table to the roles. You specify the proportions for testing and validation by using the TEST= and VALIDATE= suboptions. If you specify both the TEST= and VALIDATE= suboptions, then the sum of the specified fractions must be less than 1 and the remaining fraction is the proportion of the observations assigned to the training role. The SEED= option specifies an integer that is used to start the pseudorandom number generator for random partitioning of data for training, testing, and validation. If you do not specify SEED=*number* or if *number* is less than or equal to 0, the seed is generated by reading the time of day from the computer's clock.

ROLE=*variable* **(<TEST='***value***'> <TRAIN='***value***'> <VALIDATE='***value***'>)**
ROLEVAR=*variable* **(<TEST='***value***'> <TRAIN='***value***'> <VALIDATE='***value***'>)**
names the *variable* in the input data table whose values are used to assign roles to each observation. This *variable* cannot also appear as an analysis variable in other statements or options. The TEST=, TRAIN=, and VALIDATE= suboptions specify the formatted values of this variable that are used to assign observation roles. If you do not specify the TRAIN= suboption, then all observations whose role is not determined by the TEST= or VALIDATE= suboption are assigned to the training role.

PRUNE Statement

PRUNE < *prune-method* >< (*prune-options*) > **;**

The PRUNE statement specifies the pruning method and related options.

You can specify the following *prune-methods*. The default *prune-method* is COSTCOMPLEXITY.

C45 < (CONFIDENCE=*confidence-level***) >**
requests C4.5 pruning (Quinlan 1993), which is based on the upper confidence limit for the error rate. For more information, see the section "Pruning" on page 1833. This pruning method is available only for classification trees (which have a categorical response). PROC TREESPLIT uses the error rate from the training data only.

You can specify the following *prune-option*:

CONFIDENCE=*confidence-level*
specifies the pruning confidence level, which must be a positive number in the range of [0, 1]. The default confidence level is 0.25.

COSTCOMPLEXITY < (*prune-options***) >**
CC < (*prune-options***) >**
requests cost-complexity pruning (Breiman et al. 1984; Quinlan 1987; Zhang and Singer 2010). You can specify this pruning method both for classification trees (which have a categorical response) and for regression trees (which have a continuous response).

By default, cost-complexity pruning uses k-fold cross validation. If you specify a validation data set by using the PARTITION statement, PROC TREESPLIT uses the validation data for subtree selection and does not perform k-fold cross validation.

You can specify one the following *prune-options*:

ALPHA=*number*
selects the subtree whose cost-complexity value α is equal to *number* instead of using k-fold cross validation. You cannot specify this option with either the KFOLD= or the LEAVES= options.

KFOLD=*number*
specifies the *number* of folds for k-fold cross validation. This option is ignored if you specify the PARTITION statement. You can specify this option and the LEAVES=SE value at the same time.

By default, KFOLD=10.

LEAVES=*number* | **ALL** | **SE**
selects the subtree according to one of the following values:

number	selects the subtree that has the *number* of leaves, or the subtree whose number of leaves is less than and closest to *number* if no subtree with exactly *number* of leaves exists, instead of using k-fold cross validation.
ALL	selects the largest tree after removing splits that do not improve the error instead of using k-fold cross validation.

SE selects the subtree by performing *k*-fold cross validation and using Breiman's 1-SE rule instead of minimum error rate. You can specify this value and the KFOLD= option at the same time.

OFF

NONE

turns off pruning completely. No pruning is performed, and no pruning plots are generated.

REDUCEDERROR <(*prune-options***)>**

REP <(*prune-options***)>**

requests reduced-error pruning (Quinlan 1986). Reduced-error pruning has two stages: subtree sequence generation and subtree selection. The validation data are used for both stages. The PARTITION statement is required. For more information, see the section "Pruning" on page 1833.

PROC TREESPLIT generates a pruning plot that shows the requested error metric as a function of the number of leaves in the subtree.

You can specify the following *prune-options*:

LEAVES=*number* **| ALL**

selects the subtree that has the requested *number* of leaves, or if no subtree with exactly that *number* of leaves is available, selects the subtree whose number of leaves is less than and closest to *number*. LEAVES=ALL selects the largest tree after removing splits that do not improve the error.

By default, PRUNE=COSTCOMPLEXITY.

SAVESTATE Statement

SAVESTATE RSTORE=*libref.data-table* **;**

The SAVESTATE statement creates an analytic store for the model and saves it as a binary object in a data table. You can use the analytic store in the ASTORE procedure to score new data. For more information, see Chapter 4, "ASTORE Procedure" (*SAS Viya: Machine Learning Procedures*).

You must specify the following option:

RSTORE=*libref.data-table*

specifies a data table in which to save the analytic store for the model. *libref.data-table* is a two-level name, where *libref* refers to the library, and *data-table* specifies the name of the output data table. For more information about this two-level name, see the DATA= option and the section "Using CAS Sessions and CAS Engine Librefs" on page 1783.

VIICODE Statement

VIICODE < *options* > **;**

The VIICODE statement writes SAS DATA step code to a file or to a catalog entry. The SAS DATA step code creates new variables on the basis of the detected variable interactions.

You can specify the following *options* in addition to all options in the CODE statement. For more information about the syntax of the CODE statement, see the section "CODE Statement" on page 13 in Chapter 2, "Shared Concepts."

ADD
requests that the newly created variables be of the form $V + W$.

LIMIT=*number*
specifies the maximum *number* of new variables to create. By default, LIMIT=200.

MISS
requests that the generated code handle missing values.

MULTIPLY
requests that the newly created variables be of the form $V \times W$.

SUBTRACT
requests that the newly created variables be of the form $V - W$.

THRESHOLD=*number*
requests that interactions with an importance less than *number* times the maximum interaction importance be ignored, where *number* must be between 0 and 1. By default, THRESHOLD=0.0001.

If the VII= option is not specified in the PROC TREESPLIT statement, then the VIICODE statement is ignored. For more information about variable interaction importance, see the section "Variable Interaction Importance" on page 1840.

WEIGHT Statement

WEIGHT *variable* **;**

The *variable* in the WEIGHT statement is used as a weight to perform a weighted analysis of the data. Observations that have nonpositive or missing weights are not included in the analysis. The values of the WEIGHT variable can be nonintegral and are not truncated. When you omit the WEIGHT statement, all observations that are used in the analysis are assigned a weight of 1.

Details: TREESPLIT Procedure

Building a Decision Tree

Algorithms for building a decision tree use the training data to split the predictor space (the set of all possible combinations of values of the predictor variables) into nonoverlapping regions. These regions correspond to the terminal nodes of the tree, which are also known as leaves.

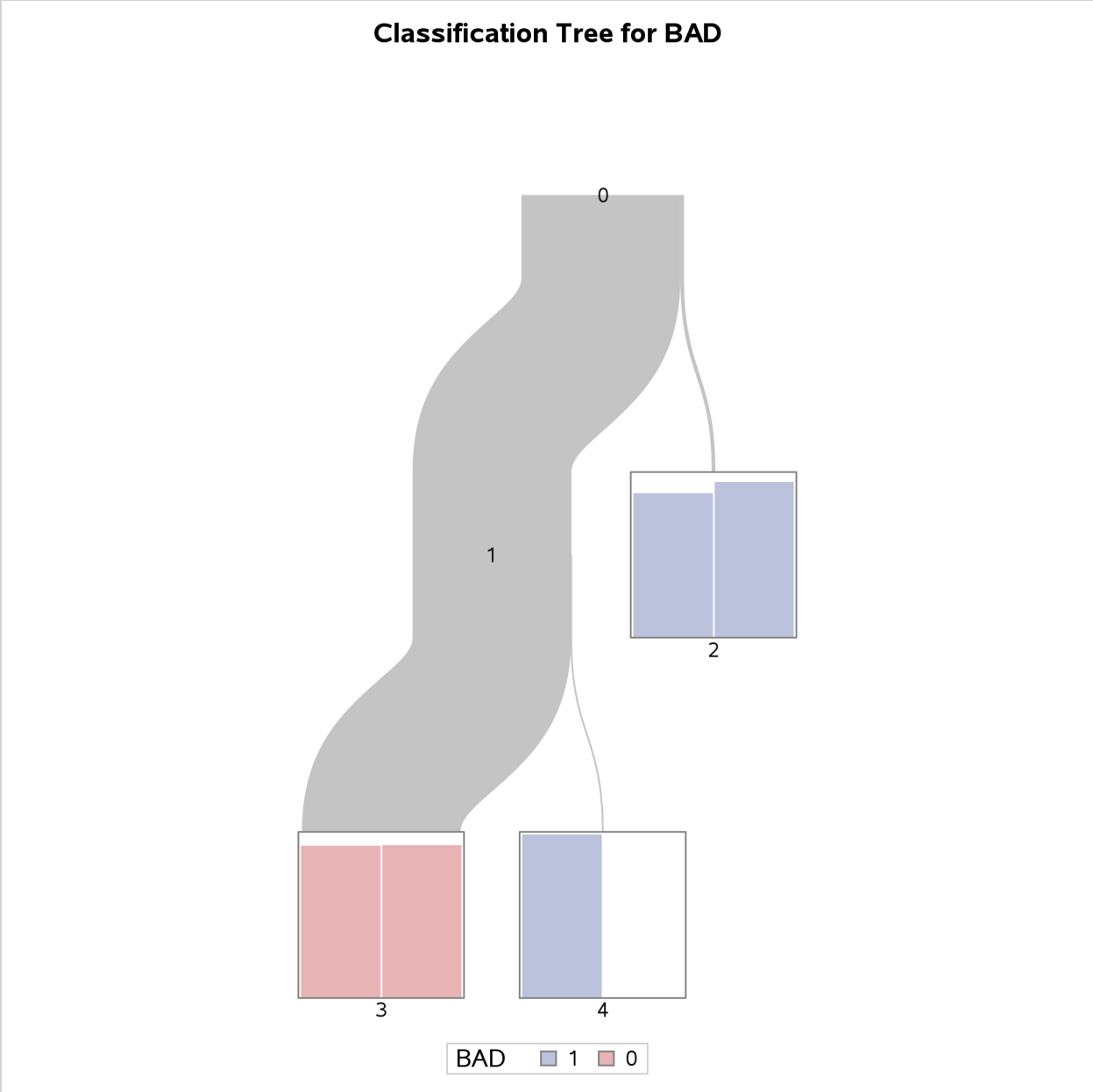
The splitting is done by recursive partitioning, starting with all the observations, which are represented by the node at the top of the tree. The algorithm splits this parent node into two or more child nodes in such a way that the values (or levels) of the response variable within each child region are as similar as possible. The splitting process is then repeated for each of the child nodes, and the recursion continues until a stopping criterion is satisfied. At that point, the tree is considered to be fully built (grown).

At each step, the split is determined by finding the best predictor variable and the best split value (or possibly a set of split values when the value of the MAXBRANCH= option is greater than 2) that optimize a specified criterion in the response variable across the child nodes to which the parent node is split. For more information about the splitting methods available in the TREESPLIT procedure, see the section "Splitting Criteria" on page 1829.

Each region, which corresponds to a terminal node, is described by a set of splitting rules that determine the observations that are assigned to the region. During scoring these rules are applied to determine the predicted values for new observations. For classification trees, the predicted value for a scored observation is the level of the response variable that is most common in the region in which the scored observation exists. For regression trees, the predicted value for a scored observation is the mean of the response variable across training observations values in the region in which the scored observation exists.

Figure 34.8, Figure 34.9, and Figure 34.10 illustrate this process. Figure 34.8 shows a classification tree of depth 2.

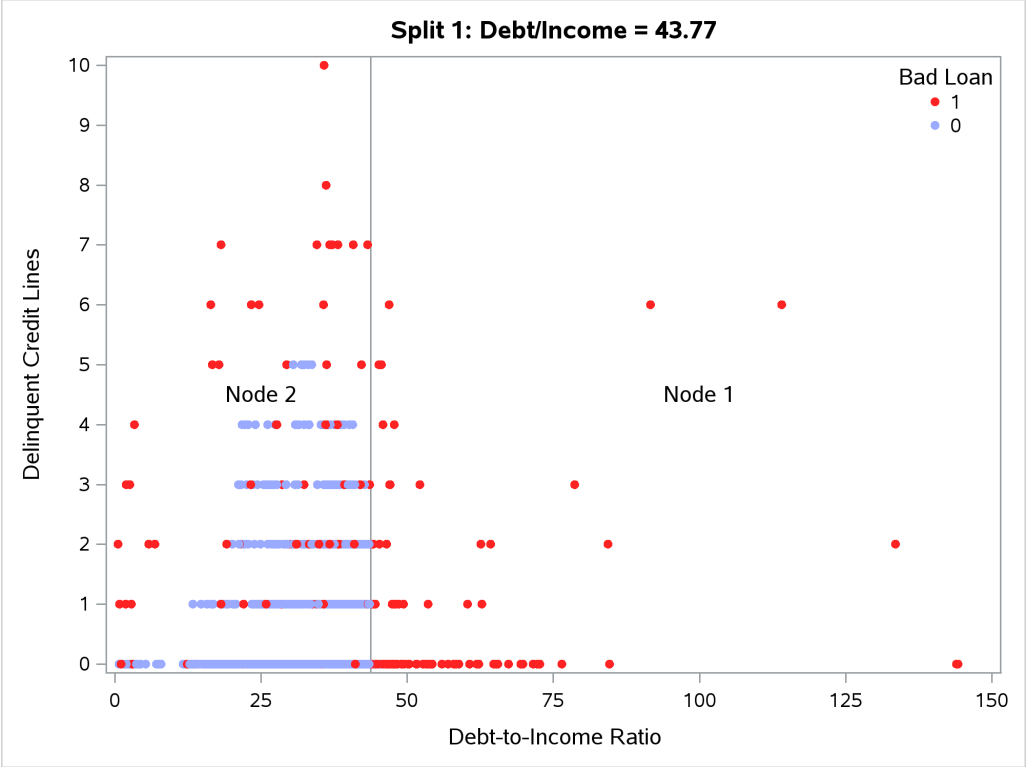
Figure 34.8 First Two Splits for the mylib.hmeq Data Table



All 4,172 observations in the data are initially assigned to node 0 at the top of the tree, which represents the entire predictor space. PROC TREESPLIT splits this space into two nonoverlapping regions, represented by node 1 and node 2. PROC TREESPLIT determines that the predictor variable and its split value that optimize the growing criterion across the child nodes are Debtinc and 43.77156, respectively. This split variable and split value constitute the splitting rule for node 0. Observations in which Debtinc $<$ 43.77156 are assigned to node 2, and observations in which Debtinc $\geq$ 43.77156 are assigned to node 1.

Figure 34.9 presents a scatter plot of the predictor space for the splitting rule for node 0. The split of the variable Debtinc divides the predictor space into node 1 and node 2; these nodes are represented by the two rectangular regions, which have different ratios of events to nonevents for the response variable.

Figure 34.9 Scatter Plot of the Predictor Space for the First Split



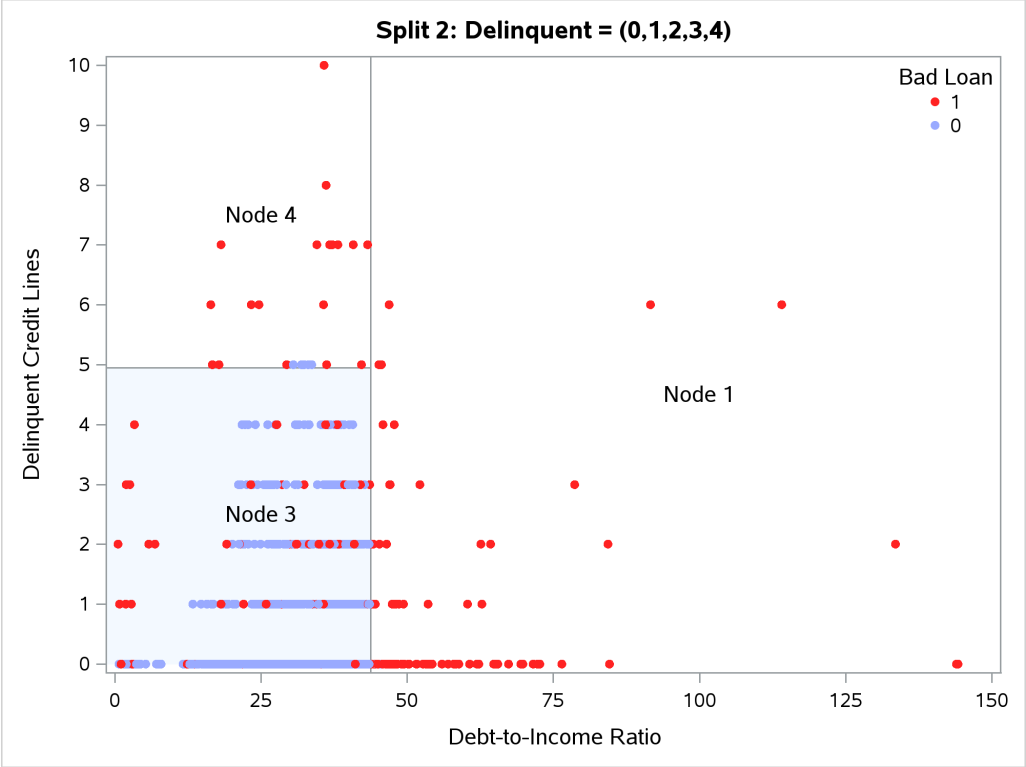
Next, the algorithm determines the splitting rule for node 2 and splits the region that is represented by node 2 into two nonoverlapping regions, represented by node 3 and node 4. PROC TREESPLIT chooses the variable Delinq and its values to optimize the growing criterion. Observations in which values of the categorical predictor variable Delinq are equal to 0, 1, 2, 3, or 4 are assigned to node 3, and observations in which the values of Delinq are equal to 5, 6, 7, 8, or 10 are assigned to node 4.

Figure 34.10 presents a scatter plot of the predictor space for the splitting rule of node 2. The split of the variable Delinq further divides the predictor space of node 2 into two regions: node 3 and node 4, which have different ratios of events to nonevents for the response variable.

Node 3 has a very high proportion of observations for which Bad is equal to 0. In contrast, Bad is equal to 1 for almost all the observations in node 4.

Note that several observations have the same values for Debtinc or Delinq, so the scatter plot gives the perception of fewer observations than there actually are.

Figure 34.10 Scatter Plot of the Predictor Space for the Second Split



This example illustrates recursive binary splitting, in which each parent node is split into two child nodes. By default, the TREESPLIT procedure creates at most two child nodes per split. You can use the MAXBRANCH= option to specify the maximum possible number of child nodes you want per split of a parent node. Based on the splitting criterion you specify, PROC TREESPLIT determines the best number of child nodes up to the value of the MAXBRANCH= option.

Splitting Criteria

The goal of recursive partitioning, as described in the section "Building a Decision Tree" on page 1826, is to subdivide the predictor space in such a way that the values of the response variable for the observations within a terminal node are similar as possible. The TREESPLIT procedure provides two types of criteria for splitting a parent node: criteria that maximize a decrease in node impurity, as defined by an impurity function, and criteria that are defined by a statistical test. You can specify the criterion in the GROW statement. If you do not specify a criterion in the GROW statement, the entropy criterion is used for classification trees and the RSS criterion is used for regression trees.

Criteria Based on Impurity

The entropy, Gini index, and RSS criteria decrease impurity. The impurity of a parent node τ is defined as $i(\tau)$, a nonnegative number that is equal to zero for a pure node (a node in which all the observations have the same value of the response variable). Nodes in which the observations have very different values of the response variable have a large impurity.

The TREESPLIT procedure selects the best split variable and the best split value to produce the highest reduction in impurity,

$$\Delta i(s,\tau) = i(\tau) - \sum_{b=1}^{B} p(\tau_b|\tau) i(\tau_b)$$

where τ_b denotes the bth child node, $p(\tau_b|\tau)$ is the proportion of observations in τ that are assigned to τ_b, and B is the number of branches after splitting τ.

Impurity Reduction Criteria for Classification Trees

The following impurity reduction criteria available for classification trees are based on different impurity functions $i(\tau)$:

- Entropy criterion: The entropy impurity of node τ is defined as

  $$i(\tau) = -\sum_{j=1}^{J} p_j \log_2 p_j$$

  where p_j is the proportion of observations that have the jth response value.

- Gini index criterion: This criterion defines $i(\tau)$ as the Gini index that corresponds to the ASE of a class response and is given by

  $$i(\tau) = -\sum_{j=1}^{J} p_j^2$$

  For more information, see Hastie, Tibshirani, and Friedman (2009).

Impurity Reduction Criterion for Regression Trees

Only one impurity reduction criterion, the RSS criterion, is available for regression trees. This criterion, also called the ANOVA criterion, defines $i(\tau)$ as the residual sum of squares,

$$i(\tau) = \frac{1}{N(\tau)} \sum_{i=1}^{N(\tau)} (Y_i - \overline{Y})^2$$

where $N(\tau)$ is the number of observations in τ, Y_i is the response value of observation i, and $\overline{Y}$ is the average response of the observations in τ.

Criteria Based on Statistical Test

The chi-square, F test, and CHAID criteria are defined by statistical tests. These criteria calculate the worth of a split by testing for a significant difference in the response variable across the branches that are defined by a split. The worth is defined as $-\log(p)$, where p is the p-value of the test. You can adjust the p-values for these criteria by specifying the BONFERRONI option in the GROW statement.

Statistical Criterion for Classification Trees

In the chi-square criterion for categorical response variables, the worth is based on the p-value for the Pearson chi-square test, which compares the frequencies of the levels of the response across the child nodes.

Statistical Criterion for Regression Trees

In the F-test criterion for continuous response variables, the worth is based on the F test for the null hypothesis that the means of the response values are identical across the child nodes. The test statistic is

$$F = \frac{\mathrm{SS}_{\mathrm{between}}/(B-1)}{\mathrm{SS}_{\mathrm{within}}/(N(\tau)-B)}$$

where

$$\mathrm{SS}_{\mathrm{between}} = \sum_{b=1}^{B} N(\tau_b)(\overline{Y}(\tau_b) - \overline{Y}(\tau))^2$$

$$\mathrm{SS}_{\mathrm{within}} = \sum_{b=1}^{B} \sum_{i=1}^{N(\tau_b)} (Y_{bi}) - \overline{Y}(\tau_b))^2$$

Statistical Criterion for Both Classification and Regression Trees

Available for both categorical and continuous response variables, CHAID, first described by Kass (1980), is an approach that regards every possible split as representing a test. CHAID tests the hypothesis of no association between the values of the response (target) variable and the branches of a node. The Bonferroni adjusted probability is defined as $m\alpha$, where α is the significance level of a test and m is the number of independent tests.

Splitting Nominal Values

Methods

PROC TREESPLIT evaluates a nominal variable in a node in one of three ways:

- enumerate: evaluate all possible splits
- sort: evaluate splits that preserve a sort order of the values
- reject: reject the variable without considering any splits

The number of categories K that are present in a node of an input variable X determines the choice. A different choice might be used for a different variable in the same node, or for the same variable in a different node because K might be different in different nodes. Let s denote the value of the SORT= suboption of the NOMSEARCH option, and let c denote the value of the MAXCATEGORIES= suboption. The following rules determine which choice PROC TREESPLIT uses:

- Enumerate when $K \leq \min(15, s, c)$.
- Sort when $\min(s, 16) \leq K < c$.
- Reject the variable when $c \leq K$.

The enumerate choice examines all possible splits. The number of splits grows exponentially with the number of categories K.

The sort choice first sorts the categories and then finds the best split that preserves the sort. The sort is on z_k, which is defined as

$$z_k = \frac{\Sigma_{i \text{ in } k}\ g_i + r\bar{g}}{n_k + r}$$

where

- k is a category
- Σ sums over observations in category k
- g_i is either the interval target value or the majority target class indicator
- r is the NOMSEARCH(SHRINKAGE=) suboption value
- $\bar{g}$ is the average of g_i across all categories
- n_k is the number of observations in category k

For a binary or nominal target, g_i is 1 if observation i is of the majority class in the node, and 0 otherwise.

Risks of Many Categories

Algorithms that search for a splitting rule preferentially select nominal inputs that contain lots of categories because of the large number of ways the categories can be combined to match noise in the training data. Irrelevant nominal inputs commonly supplant predictive inputs, creating spurious splits that can degrade accuracy on new data.

Memory depletion is another risk, especially in the output model table. The branch assignment of every category is stored. The OUTMODEL= table contains a row for every node in the model and a column for every category assigned to that node. The table contains enough columns to accommodate the node with the most category assignments, even if only one node requires it. A single nominal input that contains lots of categories can create a model so large as to exhaust memory. The MAXCATEGORIES=*number* suboption prevents splitting a node with a nominal input that has more than *number* categories in the node. An input with more than *number* categories in one node and fewer than *number* categories in a second node is eligible for splitting the second node but not the first.

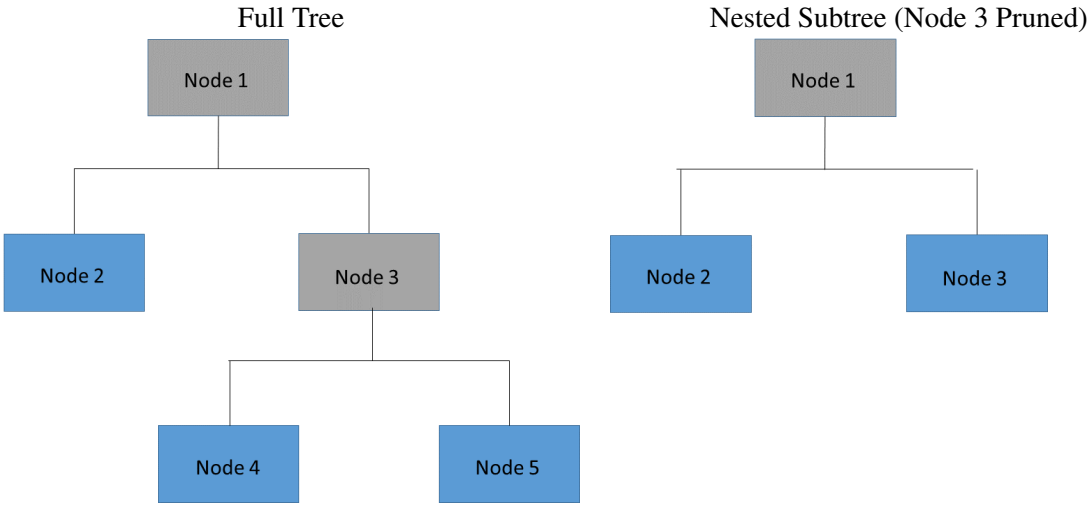
Scoring can take noticeably longer when a splitting rules include many categories.

Pruning

The TREESPLIT procedure creates a classification or regression tree by first growing a tree as described in the section "Splitting Criteria" on page 1829. This usually results in a large tree that provides a good fit to the training data. The problem with this tree is its potential for overfitting the data: the tree can be tailored too specifically to the training data and not generalize well to new data. The solution is to find a smaller subtree that results in a low error rate on the validation data.

It is often prohibitively expensive to evaluate the error on all possible subtrees of the full tree. A more practical strategy is to focus on a sequence of nested trees that are obtained by successively pruning leaves from the tree. Figure 34.11 shows an example of pruning in which node 3's leaves (terminal nodes 4 and 5) are removed to create a nested subtree of the full tree. In the nested subtree, node 3 is now a leaf that contains all the observations that were previously in nodes 4 and 5. This process is repeated until only the root node remains.

Figure 34.11 Tree and Pruned Subtree



Many different methods have been proposed for pruning in this manner. These methods address both how to select which nodes to prune to create the sequence of subtrees and how then to select the optimal subtree from this sequence as the final tree. You can use the PRUNE statement in PROC TREESPLIT to specify which pruning method to apply and related options. Several well-known pruning methods, described in this section, are available, and you can override the final selected tree based on your preferences or domain knowledge.

Cost-Complexity Pruning

Cost-complexity pruning is a widely used pruning method that was originally proposed by Breiman et al. (1984). You can request cost-complexity pruning for either a categorical or continuous response variable by specifying the following statement:

```
prune costcomplexity;
```

The cost-complexity pruning method helps prevent overfitting by making a trade-off between the complexity (size) of a tree and the error rate Thus large trees with a low error rate are penalized in favor of smaller trees.

The cost complexity of a tree T is defined as

$$\mathrm{CC}(T) = \mathrm{R}(T) + \alpha|T|$$

where R(T) represents T's error rate, $|T|$ represents the number of leaves on T, and the complexity parameter α represents the cost of each leaf. For a categorical response variable, the misclassification rate is used for the error rate, R(T); for a continuous response variable, the residual sum of squares (RSS), also called the sum of square errors (SSE), is used for the error rate. Only the training data are used to evaluate cost complexity. Breiman et al. (1984) show that for each value of α, there is a subtree of T that minimizes cost complexity. When $\alpha = 0$, this is the full tree, T_0. As α increases, the corresponding subtree becomes progressively smaller, and the subtrees are in fact nested. Then, at some value of α, the root node has the minimal cost complexity for any α greater than or equal to that value. Because there are a finite number of possible subtrees, each subtree corresponds to an interval of values of α; that is,

$$\begin{aligned}
[0, \alpha_1) &= \text{interval where } T_0 \text{ (the full tree) has minimal cost complexity} \\
[\alpha_1, \alpha_2) &= \text{interval where } T_1 \text{ has minimal cost complexity} \\
&\vdots \\
[\alpha_m, \infty) &= \text{interval where } T_m \text{ (the root node) has minimal cost complexity}
\end{aligned}$$

PROC TREESPLIT uses weakest-link pruning, as described by Breiman et al. (1984), to create the sequence of $\alpha_1, \ldots, \alpha_m$ values and the corresponding sequence of nested subtrees, $T_1, \ldots, T_m$.

Finding the optimal subtree from this sequence is then a question of determining the optimal value of the complexity parameter α. This is performed either by using the validation partition (when you use the PARTITION statement to reserve a validation holdout sample) or by using cross validation. In the first case, the subtree in the pruning sequence that has the lowest validation error rate is selected as the final tree. When there is no validation partition, k-fold cross validation can be applied to cost-complexity pruning to select a subtree that generalizes well and does not overfit the training data (Breiman et al. 1984; Zhang and Singer 2010). The algorithm proceeds as follows after creating the sequence of subtrees and α values by using the entire set of training data as described earlier:

1. Randomly divide the training observations into k approximately equal-sized parts, or folds.
2. Define a sequence of β values as the geometric mean of the endpoints of the $[\alpha_i, \alpha_{i+1})$ intervals (that is, $\beta_i = \sqrt{\alpha_i \times \alpha_{i+1}}$) to represent the intervals.
3. For each of the k folds, hold out the current fold for validation and use the remaining $k - 1$ folds for the training data in the following steps:
   a) Grow a tree as is done using the full training data set with the same splitting criterion.
   b) Using the $\beta_1, \ldots, \beta_m$ values that are calculated in step 2, create a sequence of subtrees for each β_i as described in the pruning steps given earlier, but now using β_i as a fixed value for α and minimizing the cost complexity, CC(T), to select a subtree at each pruning step.
   c) For each β_i, set T_{ij} to be the subtree that has the minimum cost complexity from the sequence for the jth fold.
   d) Calculate the error for each T_{ij} by using the current (jth) fold (the one omitted from the training).

4. Now the error rate can be averaged across folds, and the β_i that has the smallest average error is selected. The tree T_i from pruning the complete training data that corresponds to the selected β_i is the final selected subtree.

Because the k-fold cross validation is done sequentially, the process can take a long time for deep trees or for large amounts of data. Consider using the PARTITION statement for pruning large amounts of data.

The TREESPLIT procedure provides two plots that you can use to tune and evaluate the pruning process: the cost-complexity analysis plot and the cost-complexity pruning plot.

When performing cost-complexity pruning with cross validation (that is, no PARTITION statement is specified), it is recommended that you examine the cost-complexity analysis plot that is created by default. This plot displays the error as a function of the complexity parameters β_i, and it uses a vertical reference line to indicate the β_i that minimizes the error. You can use this plot to examine alternative choices for β_i. For example, you might prefer to select a smaller tree that has only a slightly higher error rate. You can specify LEAVES=SE to use the 1-SE rule developed by Breiman et al. (1984).

When you specify validation data by using the PARTITION statement, the cost-complexity pruning plot displays the error rate $R(T)$ as a function of the number of leaves $|T|$ for both the training and validation data. This plot indicates the final selected tree, the tree with the minimum $R(T)$ for the validation data, by using a vertical reference line. Like the cost-complexity analysis plot that is produced when you perform cross validation, this plot can help you identify a smaller tree that has only a slightly higher validation error rate. For an example of this plot, see Output 34.1.5 in "Example 34.1: Creating a Binary Classification Tree with Validation Data" on page 1856.

You can use the LEAVES= option in the PRUNE statement to select a tree that has a specified number of leaves or fewer if a smaller subtree has the same error.

Alternatively, you can use the ALPHA= option in the PRUNE statement to select a tree that has a specified value of the cost-complexity value α.

The computation of the subtree sequence can take a long time when the MAXDEPTH= option value is large. This will be remedied in a future release.

C4.5 Pruning

Quinlan (1987) first introduced pessimistic pruning as a method of pruning classification trees. In this method, the estimate of the true error rate is increased by using a statistical correction in order to prevent overfitting. C4.5 pruning (Quinlan 1993) evolved from pessimistic pruning to employ an even more pessimistic (that is, higher) estimate of the true error rate. An advantage of methods such as pessimistic and C4.5 pruning is that they enable you to use all the data for training instead of requiring a holdout sample. In C4.5 pruning, the upper confidence limit of the true error rate based on the binomial distribution is used to estimate the error rate. PROC TREESPLIT implements a C4.5 algorithm variant that uses the beta distribution in place of the binomial distribution to estimate the upper confidence limit. This pruning method is available only for categorical response variables and it uses only training data for tree pruning. It is implemented by the following statement:

```
prune C45;
```

The C4.5 pruning method follows these steps:

1. Grow a tree from the training data table, and call this full, unpruned tree T_0.

2. Set $i = 0$, and do the following until T_i is only the root node:

   a) For each leaf (terminal node) in the tree T_i, solve the following equation for p_l (which is the adjusted prediction error rate for leaf l):

   $$\alpha = 1 - \frac{\Gamma(N_l + 1)}{\Gamma(F_l + 1)\,\Gamma(N_l - F_l)} \int_0^{p_l} \nu^{F_l} (1 - \nu)^{N_l - F_l + 1}\, \mathrm{d}\nu$$

   Here the confidence level α is the value of the CONFIDENCE= option in the PRUNE statement, F_l is the number of failures (misclassified observations) at leaf l, N_l is the number of observations at leaf l, and the function $\Gamma(x)$ is defined as

   $$\Gamma(x) = \int_0^{\infty} \nu^{x-1} e^{-\nu} \mathrm{d}\nu$$

   b) Given these values of p_l, use the following formula for the prediction error E_i of tree T_i:

   $$E_i = \sum_{l \in T_i} N_l\, p_l$$

   c) For each node in tree T_i that has only leaves as children, create a candidate subtree by pruning those leaves.

   d) For each candidate subtree, use the equations from steps 2 and 3 to calculate its prediction error. Then select the candidate subtree that has the largest decrease (or smallest increase) in prediction error, E_i. Let this be the next subtree in the sequence, T_{i+1}.

   e) Set $i = i + 1$

3. Calculate the change in error between each pair of consecutive subtrees, $\Delta_i = E_i - E_{i-1}$ for each $i = 1, \ldots, m$.

4. Find the smallest integer j such that $\Delta_j > 0$.

5. Select the subtree T_{j-1} as the final subtree.

Reduced-Error Pruning

Quinlan's reduced-error pruning (1987) performs pruning and subtree selection based on minimizing the error rate in the validation partition at each pruning step and then in the overall subtree sequence. The error rate is based on the misclassification rate for a categorical response variable and on the ASE for a continuous response. The following PRUNE statement implements reduced-error pruning:

```
prune reducederror;
```

Reduced-error pruning creates a sequence of subtrees from the largest tree, T_0, to the root node, T_m. The subtree that has the smallest validation error is then selected as the final subtree. Pruning could be stopped as soon as the error begins to increase in the validation data as originally described by Quinlan; continuing to prune to create a subtree sequence back to the root node enables you to select a smaller tree that still has an acceptable error rate, as discussed in the next section.

User Specification of Subtree

You might want to select a different tree from the one selected by default when you use cost-complexity or reduced-error pruning to create the sequence of subtrees. For example, you might have a subtree that has a slightly larger error but is smaller, and thus simpler, than the subtree that has the minimum error according to reduced-error pruning. You can override the selected subtree and instead select the subtree that has *n* leaves and was created by cost-complexity or reduced-error pruning, where *n* is specified in the LEAVES= option in the PRUNE statement. In addition, if you are using cost-complexity pruning, you can override the selected subtree by using the ALPHA= option in the PRUNE statement.

Alternatively, you might want to select the largest tree that is created in one of the following ways:

- Specify LEAVES=ALL in the PRUNE statement to still see the statistics for the sequence of subtrees that are created according to the specified pruning error measure, even though the largest (unpruned) tree is selected as the final subtree.
- Specify the following statement to select the largest tree with no pruning performed:

  ```
  prune none;
  ```

  Statistics are not calculated and plots are not created for a sequence of subtrees.

Scoring

After you create a tree model, you can apply it to training or test data for model assessment or to new data for making predictions. The process of applying a model to a data table is called *scoring*. You can score data as described in the following sections.

Scoring the Input Data Table

Usually, the purpose of scoring training data is to diagnose the model. The training data table is the data table that you specify with the DATA= option. To score the training data, use the OUTPUT statement to create an output data table that contains one observation for each observation in the training data. You can specify the output data table by using the OUT= option in the OUTPUT statement.

In the following example, the input data table (mylib.hmeq) is scored after the tree model has been created:

```
proc treesplit data=mylib.hmeq;
   class Bad Delinq Derog Job Ninq Reason;
   model Bad = Delinq Derog Job Ninq Reason;
   output out=mylib.scored;
run;
```

For classification trees, the scored data table also contains one new variable for each level of the response variable. These new variables have the prefix “P_”; for all observations in the same leaf, these new variables represent the proportion of the training observations in that leaf that have that particular response level. For example, if the name of the categorical response variable is Color and it has two levels, `'Blue'` and `'Green'`, then the scored data table contains the variable P_ColorBlue (which provides the proportion

of training data in this leaf that have the response level `'Blue'`) and the variable P_ColorGreen (which provides the proportion of training data in this leaf that have the response level `'Green'`).

For regression trees, the scored data table contains exactly one new variable with the prefix "P_", which represents the average value of the response variable for all observations in the same leaf. For example, if the name of the continuous response variable is logSalary, then the scored data table contains one new variable, P_logSalary, which represents the average value of the response variable logSalary in the training data for observations in the same leaf.

Scoring Using DATA Step Code

You can use the CODE statement to generate SAS DATA step code that you can use to score new data. The following example uses PROC TREESPLIT to produce SAS DATA step code:

```
proc treesplit data=mylib.hmeq;
   class Bad Delinq Derog Job Ninq Reason;
   model Bad = Delinq Derog Job Ninq Reason;
   code file="treesplit_data_step.sas";
run;
```

Measures of Model Fit

The TREESPLIT procedure measures model fit based on the misclassification rate for classification trees or the average square error for regression trees, as follows:

- The misclassification rate is calculated from the number of incorrectly predicted observations:

$$\text{Misc} = \frac{1}{N_0} \sum \begin{cases} 0 & \text{if prediction is correct} \\ 1 & \text{otherwise} \end{cases}$$

- The average square error (ASE) for regression trees is defined as

$$\text{ASE} = \frac{\text{RSS}}{N_0}$$

Variable Importance

A training data table can contain a large number of predictor variables, some of which are useful for predicting the response variable and others of which are not. You can use the TREESPLIT procedure to select the most useful predictor variables on the basis of variable importance (Breiman et al. 1984). (See "Example 34.3: Assessing Variable Importance" on page 1865.) Variable importance is an indication of which predictor variables are the most useful for predicting the response variable.

The most important variables might not be the ones near the top of the tree. The TREESPLIT procedure implements several methods for computing variable importance, which are described in the following subsections. By default, PROC TREESPLIT calculates the variable importance by three methods (count-based, change in the residual sum of square errors, and relative importance), and outputs the results from these variable importance methods in one table. You can request that the TREESPLIT procedure also calculate the variable importance by random branch assignment (RBA) by specifying the RBAIMP option.

Count-Based Importance Method

The count-based variable importance simply counts the number of times in the tree that a particular variable is used in a split.

Residual Sum of Squares (RSS) Importance Method

This method measures variable importance based on the change of RSS when a split is found at a node. The change for variable v is

$$\Delta_d^v = \mathrm{RSS}_d - \sum_i \mathrm{RSS}_i^d$$

where

- d denotes the node
- i denotes the index of a child that this node includes
- RSS_d is the RSS if the node is treated as a leaf
- RSS_i^d is the RSS of the node after it has been split

If the change in RSS is negative (which is possible when you measure the validation set), then the change is set to 0.

The residual sum of squares for regression trees is defined as

$$\mathrm{RSS} = \sum_\lambda \sum_{i \in \lambda} \left(y_i - \hat{y}_\lambda^T \right)^2$$

where

- i is an observation on leaf λ
- y_i is the predicted value of the response variable of observation i

- $\hat{y}_{\lambda}^{T}$ is the actual value of the response variable on leaf λ

The residual sum of squares for classification trees is defined as

$$\mathrm{RSS} = \sum_{\lambda} \sum_{\Phi} N_{\Phi}^{\lambda} \left[\sum_{\tau \neq \Phi} \left(P_{\tau}^{\lambda} \right)^2 + \left(1 - P_{\Phi}^{\lambda} \right)^2 \right]$$

where

- Φ is the actual response level
- N_{Φ}^{λ} is the number of observations on leaf λ that have response level Φ
- P_{τ}^{λ} is the posterior probability for the response level τ on leaf λ
- P_{Φ}^{λ} is the posterior probability for the actual response level Φ on leaf λ

The RSS-based importance for a single variable, v, is then defined as

$$\sum_{d=1}^{D} \Delta_{d}^{v}$$

where D is the total number of nodes in which v was used as the splitting variable.

Relative Variable Importance Method

The relative variable importance metric is a number between 0 and 1, which is calculated in two steps:

1. PROC TREESPLIT finds the maximum RSS-based variable importance.
2. For each variable, PROC TREESPLIT calculates the relative variable importance as the RSS-based importance of this variable divided by the maximum RSS-based importance among all the variables.

The RSS and relative importance are calculated from the validation data. If no validation data exist, they are calculated instead from the training data.

Variable Interaction Importance

In some cases, interactions of variables are of more interest than a single variable. When you specify the VII= option in the PROC TREESPLIT statement, the procedure computes variable interaction importance as follows.

The two-way interaction importance for the variables v and w is

$$\sum_{d,e} \frac{\Delta_{d}^{v} + \Delta_{e}^{w}}{\Delta_{d}^{v}}$$

where

- d and e denote nodes, where node d is the parent of node e

- v and w are variables, where v is the splitting variable for node d and w is the splitting variable for node e
- this summation is across all parent-child node combinations with respect to the variables v and w

The three-way interaction importance for the variables v, w, and x is

$$\sum_{d,e,f} \frac{\Delta_d^v + \Delta_e^w + \Delta_f^x}{\Delta_d^v + \Delta_e^w}$$

where

- d, e, and f denote nodes, where node d is the parent of node e, which is the parent of node f
- v, w, and x are variables, where v is the splitting variable for node d, w is the splitting variable for node e, and x is the splitting variable for node f
- this summation is across all parent-child-grandchild node combinations with respect to the variables v, w, and x

When comparing variable importance values, you should compare a two-way interaction only with another two-way interaction, or a three-way interaction only with another three-way interaction. Comparing a two-way interaction to a three-way interaction is not meaningful, because the values of the importance are not on the same scale.

In addition to calculating the variable interaction importance, you can also generate SAS DATA step code to create interaction variables and save the code to a file by using the VIICODE statement. When you run the generated SAS DATA step code on the original data or on new data, interaction variables are created.

Created interaction variables are of the form $W + V$, $W - V$, or $W \times V$, where the values that are used in the calculations come from the variables whose interactions are determined to be important. The values that are used are as follows:

- For continuous variables, the raw value of the variable is used.
- For categorical variables, the proportion of observations in the training set that is assigned to a specific branch is used.

Random Branch Assignment Importance Method

The random branch assignment (RBA) method computes the importance of an input variable v by comparing how well the data fit the predictions before and after they are modified. Neville and Tan (2014) motivate and introduce the RBA method of variable importance. To modify the predictions, the TREESPLIT procedure replaces all splitting rules that use variable v by a rule that randomly assigns an observation to a branch. The probability of assigning an observation to a branch is proportional to the number of observations that are assigned to the branch in the current data. The current data are the training data when RBA is computed during training. Otherwise, the current data are those being scored on an existing model.

The RBA importance can be expressed mathematically as

$$I_{\mathrm{RBA}}(v) \propto \sum_{i=1}^{n} \mathrm{Loss}(y_i, \ddot{y}_i) - \sum_{i=1}^{n} \mathrm{Loss}(y_i, \hat{y}_i)$$

where $\ddot{y}_i$ is the modified prediction for observation i and $\hat{y}_i$ is the standard prediction.

For an continuous response, PROC TREESPLIT computes the RBA importance of squared error loss. For a categorical response, PROC TREESPLIT uses the misclassification rate as the loss function.

Neville and Tan (2014) motivate and introduce the RBA method of variable importance.

Hyperparameter Tuning

The quality of the predictive model that a machine learning algorithm creates depends on the values for various options that govern the training process; these options are called *hyperparameters*. The default values of these hyperparameters might not be suitable for all applications. In order to reduce the manual effort in adjusting these hyperparameters, you can use the AUTOTUNE statement to identify the best settings for them.

To tune hyperparameters, the AUTOTUNE statement directs the procedure to engage a search method (tuner) that searches for the best possible combination of values of these hyperparameters while trying to minimize or maximize an objective function. The objective function is a validation error or accuracy estimate—for example, MISC (misclassification error) for nominal targets or ASE (average square error) for interval targets. The tuning process involves multiple evaluations of the objective function (one for each model hyperparameter configuration it evaluates) and might include multiple iterations, depending on the specified search method. The tuning process also requires data partitioning in order to avoid overfitting to training data; models are trained on a training partition and validated on a validation partition to obtain the objective function value for each chosen hyperparameter configuration. The options for data partitioning are described in the next section, followed by descriptions of the default hyperparameter tuning process and determination of sizing for parallel tuning.

Data Partitioning

Each evaluation of the objective function can consist of one or several training and scoring (validation) executions, depending on whether single-partition validation or k-fold cross validation is used. In both cases, the tuner partitions the data unless the PARTITION statement is used (in this case, the tuner does not partition the training data, but instead uses the data roles that are specified in the PARTITION statement for training and validation).

Single-Partition Validation

By default, the tuner uses single-partition validation. In this process, the tuner partitions all the data into two subsets: one subset for model training and one subset for model validation. For each newly generated configuration of hyperparameters, a new model is trained on the training subset and then the validation subset is scored using the trained model to find the resulting objective function value.

The AUTOTUNE statement calls the **`sampling`** action set, which creates the training and validation partitions. The **`stratified`** action in the **`sampling`** action set is used for a target of nominal type (if all target levels can be included in both the training and validation partitions), and the **`srs`** action in the **`sampling`** action

set is used for a target of interval type and for cases in which stratified sampling is not possible. By default, a validation partition of 30% is used and the remaining 70% is used for model training. The validation partition size can be adjusted by specifying the FRACTION= option in the AUTOTUNE statement. For more information about the **sampling** action set, see Chapter 24, "Sampling and Partitioning Action Set" (*SAS Visual Statistics: Programming Guide*).

Cross Validation

If *k*-fold cross validation is specified, the tuner partitions all the data into *k* subsets (folds). For each fold, a new model is trained on all folds except the selected (holdout) fold (that is, it is trained on *k*–1 folds) and then validated using the selected (holdout) fold. The objective function value is averaged over the set of fold validation executions to obtain a single error estimate value. Cross validation is specified using the KFOLD= option in the AUTOTUNE statement.

For efficiency, the cross validation process might be terminated before all *k* folds are evaluated. Cross validation is terminated under the following conditions: the validation score is 2 times worse than the current best score after the first fold, the validation score is 1.75 times worse than the current best score after two folds, or the validation score is 1.5 times worse than the current best score after three or more folds.

Hyperparameter Tuning Process

By default, the tuner chooses the model hyperparameter configurations to evaluate by using a hybrid optimization strategy that is based on a genetic algorithm (GA), which applies principles of natural selection and evolution to find an improved configuration. The tuner performs the following steps:

1. A default model configuration (default values of select model hyperparameters) is evaluated first and designated as Iteration 0. The objective function value is obtained by using either single-partition validation or *k*-fold cross validation and then recorded for comparison.

2. An initial set of hyperparameter configurations, also called a "population," is generated using a technique called Latin hypercube sampling (LHS). Each configuration of hyperparameters in the Latin hypercube sample is evaluated, and the objective function value for each is again recorded for comparison. This set becomes Iteration 1. All or a portion of the sample configurations can be evaluated in parallel, as described in the section "Determining the Number of Parallel Evaluations" on page 1844.

3. The best model configurations from the Iteration 1 are used to generate the next population of model configurations, Iteration 2, which are then evaluated, in parallel when possible (see the section "Determining the Number of Parallel Evaluations" on page 1844). This process is repeated for the remaining iterations, as long as the maximum number of evaluations or the maximum time has not been reached. The default number of iterations is 5, and the default maximum time is 36,000 seconds (10 hours).

4. The best model configuration (the one that minimizes or maximizes the objective function) is reevaluated by executing a single model training and scoring, and information about the model training and scoring for this configuration is returned.

5. All evaluated model configurations are ranked, and the hyperparameter and objective function values of the top 10 configurations are returned in the TunerResults table.

You can specify four alternative search methods: Bayesian, random, LHS, and grid. The preceding steps are performed differently, depending on the search method that you use:

- For the Bayesian search method, the preceding steps are performed in the same sequence, with the following changes. Instead of using genetic operations to generate the new combinations of hyperparameter values, an internal surrogate model is created after the first iteration. This surrogate model is then explored for potential candidate configurations, and the new set of configurations is created and evaluated in parallel. After each iteration, the internal surrogate model is updated with the new data and explored again for new potential configurations.
- For the random, LHS, or grid search method, step 3 in the preceding sequence is eliminated; a single sample of candidate configurations is generated and evaluated in step 2. This single sample of configurations is generated in one of the following ways:
  - randomly, for the random search method
  - by using a Latin hypercube sample, for the LHS search method
  - by creating all possible combinations of hyperparameter levels, for the grid search method. If the VALUELIST= option is specified for any hyperparameter, the levels are taken from that list. Otherwise, the tuner takes the lower bound, the upper bound, and the middle value between the two bounds and uses those values as the grid levels for each hyperparameter. The total number of grid configurations is the product of the numbers of levels for each hyperparameter. If the number of hyperparameters and the number of levels is greater than a very small value, then the total number of combinations can be many hundreds or thousands. If the MAXEVALS= option is specified, the tuner uses it to terminate execution of the grid search method before all grid points are evaluated.

If the tuner had partitioned the data for training and scoring (if the PARTITION statement is not specified), then the final (best) model configuration is trained on all the data and scored on all the data; as a result the final model that is returned and its reported validation score might not match the value that is observed during tuning, when the model was trained to a portion of the data and validated on the remaining validation partition.

Determining the Number of Parallel Evaluations

By default, the number of possible parallel evaluations during step 2 and step 3 (if performed) in the section "Hyperparameter Tuning Process" on page 1843 is determined by the population size for the GA or Bayesian search method or by the sample size for the random or LHS sampling method. The maximum number of parallel evaluations is limited by the total number of worker nodes that are connected to the server divided by the number of worker nodes that are used by the parallel sessions that are created. For example, if the server is configured with 100 worker nodes and four workers are used in the sessions, at most 25 parallel evaluations are performed by default. This limit can be overridden by a factor of 2, resulting in up to 50 parallel evaluations in this example scenario. Because of the data movement that parallel subsessions produce, repeated runs with the same seed can produce different results.

Specifically, the number of parallel evaluations is determined as follows:

1. If you do not specify the number of workers to use in parallel subsessions by using the NSUBSESSIONWORKERS= option in the AUTOTUNE statement, the number of workers is determined using

the size of the data table: NSUBSESSIONWORKERS = 1 + *nDataRows* * *nDataColumns* * 2E–8. If determined on the basis of the data table size, the number of workers in each subsession is fairly aggressively set at one node per 50 million values. The efficiency benefits of tuning in parallel generally outweigh the time savings from distributed training if the data can be managed on fewer nodes. For example, if training a single model on four workers takes 1 minute versus 2 minutes on one worker, then training four models in parallel on one worker each requires only 2 minutes, whereas training four sequential models each on four workers would require 4 minutes.

2. The number of possible parallel evaluations is then determined either as one less than the population size for the GA or Bayesian search method, or as the sampling size for the random or LHS search method.

3. The number of parallel evaluations is then limited by the server configuration:

   - In single-machine mode, if the number of possible parallel evaluations is greater than 4 and not specified in the NPARALLEL= option in the AUTOTUNE statement, it is limited to 4. You can override this limit up to a value of 32 by specifying the NPARALLEL= option (the value of the NPARALLEL= option is reduced if it is greater than 32).
   - In distributed mode, the upper limit for the number of parallel evaluations is calculated as W/n, where W is the number of workers connected to the server and n is the number of workers in the parallel subsessions. You can override this limit by a factor of 2 by specifying the NPARALLEL= option; the new limit is $2W/n$ (the value of the NPARALLEL= option is reduced if it is greater than $2W/n$).

Lookup Table and History Table

Tuning a predictive model often requires multiple runs of a tuning action and various adjustments of the hyperparameter bounds or their initial values. This often means training a large number of model configurations for each new tuning process, some of which might be duplicates of configurations from the previous tuning runs. To avoid rerunning the same configurations, you can use a lookup table, which you specify in the LOOKUPTABLE= option in the AUTOTUNE statement. You can create this lookup table manually, such as by using a DATA step, or you can produce it automatically in any procedure call in which you specify the HISTORYTABLE= option in the AUTOTUNE statement. Before evaluating a new predictive model configuration, the tuner first checks the lookup table for a potential match. The lookup table must contain all the hyperparameter columns and the objective column; if any required columns are not found, the tuner rejects the lookup table. If a matching configuration is found, its objective value is read from the lookup table and the configuration is not evaluated. The tuner matches only the hyperparameter values and ignores all other columns in the lookup table. For convenience, when you do multiple runs of a procedure in which you use autotuning, you can collect the history of all runs in one table by using the APPENDLOOKUP option in the AUTOTUNE statement. This causes the tuner to append all unused configurations from the lookup table to the history table, which can then be used again as the lookup table for the next procedure call in which you use autotuning. It is essential that the lookup table contain only the appropriate data. If any nontuned parameters have changed, or if the data that are used for the predictive model have changed, then the objective values inside the lookup table might no longer be valid for the new tuning process. The tuner is oblivious to any changes in the problem setup between any two runs of the tuning process and will use the objective values from the lookup table as long as the hyperparameter values match the configuration to be evaluated.

You can use the AUTOTUNE statement to tune the following options in the PROC TREESPLIT statement:

- the value of the NUMBIN= option for the number of bins for continuous predictors
- the value of the MAXDEPTH= option for the maximum depth of the decision tree
- the criterion on the GROW statement for how the nodes are split

k-fold Cross Validation

The CROSSVALIDATION statement performs *k*-fold cross validation to assess the accuracy of a model. During cross validation, all data are divided into *k* subsets (folds), where *k* is the value of the KFOLD= option. For each fold, a new model is trained on the (*k*–1) folds and then validated using the selected (holdout) fold. The assessment metrics are then averaged across all the holdout folds. The CROSSVALIDATION statement creates a table that has *k*+1 rows. The first *k* rows contain the assessment metrics for each holdout fold, and the last row contains the average across all the holdout folds. Which metrics the table includes depends on whether the target is nominal or interval. For nominal targets, the table includes the values of square error, mean consequential error, and multiclass log loss. For interval targets, the table includes the values of square error, absolute error, and square logarithmic error.

Determining the Number of Parallel Evaluations

The number of parallel fold evaluations is determined by the following two-step process:

1. The number of worker nodes to use in parallel subsessions is determined as follows:
   a) If you specify a value greater than 0 in the NSUBSESSIONWORKERS= option to indicate the number of workers to use in parallel subsessions, then that value is used for each subsession.
   b) If you do not specify a value in the NSUBSESSIONWORKERS= option, or if you specify a value of 0, then the number of workers for each subsession is the same as the number of workers that are used in the parent session. If the number of workers that are used in the parent session times the number of folds is greater than the number of workers available on the server, then the number of workers for each subsession is reduced to the number of workers on the server divided by the number of folds. The number of workers in each subsession is then increased if necessary, according to the size of the data table, as follows: NSUBSESSIONWORKERS= 1+*Number of Observations***Number of Columns* / 50,000,000 (the number of worker nodes in each subsession is at least one node per 50 million values).
2. The number of parallel evaluations is then limited by the server configuration as follows:
   a) In single-machine mode, if the number of folds is greater than 16, then the number of parallel evaluations is limited to 16. Otherwise, the number of parallel evaluations is equal to the number of folds.
   b) In distributed mode, the upper limit for the number of parallel evaluations is calculated as 2**W*/*n*, where *W* is the number of workers that are connected to the server and *n* is the number of workers in the parallel subsessions.

Bias Mitigation

With the widespread use of AI and machine learning models in decision-making processes, there is concern about the social impacts of these models. Bias can be embedded in both the training data and the trained model, either of which can lead to unfair or biased predictions for certain groups or populations. To reduce bias in the data and models, researchers have developed three main categories of bias mitigation methods:

- Preprocessing methods, which transform the data prior to model training. These methods try to eliminate the correlations between the input features and the sensitive feature while retaining as much information from the input data as possible.
- In-process methods, which consider fairness constraints during the model training process. These methods actively adjust the model parameters during training to create a model that produces fair predictions and classifications.
- Postprocessing methods, which do not directly alter the input data or the model training process. These methods adjust the predictive outputs from the learned models to compensate for the bias in the model.

You can reduce bias in the trained model by specifying the MITIGATEBIAS statement. This statement uses the exponentiated gradient reduction (EGR) algorithm, which is an in-process method of mitigating bias.

Exponentiated Gradient Reduction (EGR) Algorithm

The exponentiated gradient reduction (EGR) algorithm was introduced at ICML 2018 (Agarwal et al. 2018). As an in-process algorithm, it considers fairness constraints during the model training process. The EGR algorithm has three main advantages. First, it is model-agnostic, which means that you can apply it to a wide variety of machine learning models. The EGR algorithm works like a wrapper that is built around existing machine learning models. The wrapper iteratively reweights and possibly relabels the data, and then it trains a new classifier model on the basis of reweighted data in each iteration. By doing so, it reduces the complex optimization problem of training a machine learning model with fairness constraints to a series of standard model training problems without the fairness constraints. Second, the EGR algorithm supports various fairness measurements, such as demographic parity, equal opportunity, and equalized odds. Finally, the algorithm has strong theoretical support in terms of its optimization trade-off between fairness and accuracy. In theory, the EGR algorithm returns solutions that satisfy all the fairness constraints and also achieve the optimal error rate of the best fair solution.

Training a fair machine learning model is in general a nonlinear programming problem. In nonlinear optimization tasks, it is not uncommon to get suboptimal solutions because the constraints are infeasible or because the algorithm is stuck in a local minimum. As with many other machine learning algorithms, the effectiveness of the EGR algorithm depends on many factors, such as the quality of the input data, the selection of fairness constraints, the quality of the machine learning model that it wraps around, and the values of the hyperparameters.

PROC TREESPLIT supports bias mitigation for multilevel nominal target variables and multilevel nominal sensitive variables. Note that when you include a large number of levels in the sensitive variable, the EGR algorithm searches more complex optimization scenarios; this can result in suboptimal solutions. Therefore, the solutions from this algorithm might not always be desirable when the number of levels in the sensitive variable is high. You should review and validate your sensitive variable, training models, and fairness constraints before applying the EGR algorithm.

Fairness Constraints

PROC TREESPLIT supports the following fairness measurements: demographic parity, predictive parity, equal opportunity, and equalized odds. The goal of the EGR algorithm is to bring the fairness metrics of groups as close together as possible, as well as to convert fair machine learning to a sequence of standard cost-sensitive machine learning problems whose solutions yield a randomized classifier with the lowest error rate, subject to the desired constraints. As a result, the EGR algorithm iteratively trains the machine learning models by using different weights to reduce the fairness disparities without significantly increasing the loss of the final machine learning model. For example, when there are two protected groups in the sensitive variable, the fairness constraints that the EGR algorithm requires for demographic parity are as follows:

$$\mu_1^t - \mu_0^t \le c$$
$$\mu_1^t - \mu_0^t \ge -c$$
$$\mu_2^t - \mu_0^t \le c$$
$$\mu_2^t - \mu_0^t \ge -c$$

where μ_1^t is the demographic parity for the first protected group during iteration t; μ_2^t is the demographic parity for the second protected group during iteration t; μ_0^t is the demographic parity for the entire population during iteration t; and c is the predefined violation tolerance threshold.

Exponentiated Gradient Reduction Algorithm Workflow

Two key hyperparameters in the EGR algorithm are specified by the BOUND= option (denoted by B) and the LEARNINGRATE= option (denoted by lr); both are available in the MITIGATEBIAS statement. To derive the fair machine learning algorithm, the fairness constraints (see the section "Fairness Constraints" on page 1848) are added to the loss function of the cost-sensitive machine learning model by introducing the Lagrange multipliers $\lambda_{j,t}$, where j is the index in the constraint set. The multipliers $\lambda_{j,t}$ are updated on the basis of the degree of constraint violations $\gamma_{j,t}$, B, and lr. Then $\lambda_{j,t}$ is used to derive the new weights for each observation in the input data set for the next round of training. Note that the values of B and lr play important roles in the performance of the EGR algorithm, based on experiments. The next section discusses how to come up with appropriate values for B and lr.

Bound Value Recommendation

Because the EGR algorithm is designed to strike a balance between model accuracy and fairness constraint violations, it does not perform well unless you specify appropriate values for the BOUND= and LEARNINGRATE= options in the MITIGATEBIAS statement. The BOUND= option is important for the sample reweighting and relabeling process. It controls how heavily the EGR algorithm penalizes fairness constraint violations, and therefore it determines the trade-off between the fairness constraints and the model accuracy. When the bound is too small, there is not enough momentum to drive the model close enough to the saddle points in the solution sets. In this case, the bias metric values of the different subgroups do not approach each other.

The LEARNINGRATE= option controls how quickly the algorithm reduces fairness constraint violations. If the learning rate is too small, the algorithm converges very slowly, resulting in a long training process; if the learning rate is too large, the algorithm becomes unstable and might not converge. In general, the learning rate and the bound value are coupled, and changing one of these values changes the optimal value of the other.

The MITIGATEBIAS statement offers a built-in bound value recommendation function to automatically search for a good candidate bound value when you request it. To eliminate the coupling effect of the bound

and the learning rate, PROC TREESPLIT fixes the initial learning rate at 0.01 during the tuning process. The procedure runs the bound value recommendation function if you specify the TUNEBOUND option. This function finds the best bound value through multiple iterations by analyzing the convergence patterns of the fairness metrics. PROC TREESPLIT first runs the EGR algorithm with some initial bound value for a few iterations, while monitoring the convergence pattern for signs of stagnation or overshooting to determine whether the bound value is too small or too large. It then adjusts the bound value until it finds an appropriate value or reaches the maximum number of iterations that are allowed. After the recommendation process, PROC TREESPLIT applies the recommended bound value to the EGR algorithm to train a final model. Note that because of the nonlinear nature of the optimization process, the bound value recommendation is also subject to the problem of local minima. This recommendation should be used as a general guideline for setting the bound value; you might find a better solution by further fine-tuning.

Displayed Output

The following sections describe the output that PROC TREESPLIT produces. The output is organized into various tables, which are discussed in the order of appearance.

Model Information

The "Model Information" table contains the settings of the training parameters. This table also contains some basic information about the resulting decision tree. This table is produced by default.

Number of Observations

The "Number of Observations" table contains the number of observations that are read from the input data table and the number of observations that are used in the analysis. When you specify the PARTITION statement, the table also indicates the number of observations that are used in each partition. This table is produced by default.

CV Statistics

The "CV Statistics" table contains information about *k*-fold cross validated cost-complexity pruning, such as the alpha values, the corresponding size of the tree, and the corresponding minimum, average, and maximum error. The table is generated when you specify the CVCC option in the PROC TREESPLIT statement, and you are also using *k*-fold cross validated cost-complexity pruning.

Tree as a Table

The "Tree as a Table" table is used for generating the WholeTreePlot and the ZoomedTreePlot. This table is produced when you specify the TREEPLOT option in the PROC TREESPLIT statement.

Tree Performance

The "Tree Performance" table contains statistics that measure the model's goodness of fit. If you specify the PARTITION statement, the table also indicates these statistics on all partitions of the data.

Variable Importance

The "Variable Importance" table displays variable importance based on residual sum of square errors, which is explained in the section "Residual Sum of Squares (RSS) Importance Method" on page 1839. This table is produced by default.

RBA Variable Importance

The "RBA Variable Importance" table displays variable importance based on the random branch assignment (RBA) method, which is explained in the section "Random Branch Assignment Importance Method" on page 1841. This table is produced by the RBAIMP option in the PROC TREESPLIT statement.

Reduced Error

The "Reduced Error" table contains the pruning results of reduced-error pruning that uses validation data. This is the table of the data shown in the PruningPlot. This table is generated when you specify the PRUNINGTABLE option and are using reduced-error pruning in the PRUNE statement with validation data.

Cost Complexity

The "Cost Complexity" table contains the pruning results of cost-complexity pruning that uses validation data. This is the table of the data shown in the PruningPlot. This table is generated when you specify the PRUNINGTABLE option and are using cost-complexity pruning in the PRUNE statement with validation data.

OutputCasTables Table

The OutputCasTables table is a special table that has information about each CAS table that is created during a CAS action execution. The information for each CAS table consists of the CAS table name, the caslib in which the table resides, and the number of columns and rows in the CAS table.

Tuner Information

The "Tuner Information" table displays the setup values that the tuner uses. This table is produced by the AUTOTUNE statement.

Tuner Summary

The "Tuner Summary" table displays statistics about the tuning process. This table is produced by the AUTOTUNE statement.

Tuner Timing

The "Tuner Timing" table displays the total time spent on different tasks during tuning. This table is produced by the AUTOTUNE statement.

Best Configuration

The "Best Configuration" table displays the hyperparameters and objective function values for the best configuration. This table is produced by the AUTOTUNE statement.

Tuner Results

The "Tuner Results" table displays the values of the hyperparameters, the objective function for the default configuration (Iteration 0), and up to 10 of the best configurations that are found. This table is produced by the AUTOTUNE statement.

Evaluation History

The "Evaluation History" tables displays the values of the hyperparameters and the objective function for all configurations. This table is produced by the AUTOTUNE statement, either by default or when EVALHISTORY=ALL.

Predicted Probability Names

The "Predicted Probability Names" table indicates the names of the created variables in either the score code or the data table that is specified in the OUT= option in the OUTPUT statement. This table is produced when you specify the PRINTTARGET option in the PROC TREESPLIT statement.

Constraint History

The "Constraint History" table lists the values of the various constraints at each iteration of the EGR algorithm. This table is produced by the MITIGATEBIAS statement.

Group Metrics History

The "Group Metrics History" table displays the metrics for accuracy and fairness for each subpopulation over each iteration of the EGR algorithm. This table is produced by the MITIGATEBIAS statement.

Iteration History

The "Iteration History" table displays the values of accuracy and bias metric at each iteration of the EGR algorithm. This table is produced by the MITIGATEBIAS statement.

Mitigation Information

The "Mitigation Information" table displays information about the options that you use for bias mitigation. This table is produced by the MITIGATEBIAS statement.

Timing

The "Timing" table displays the breakdown of how long bias mitigation spent on the various tasks of training models. This table is produced by the MITIGATEBIAS statement.

Weight History

The "Weight History" table displays the weights that are assigned by the EGR algorithm for each level of the response (target) variable and for each subpopulation. This table is produced by the MITIGATEBIAS statement.

ODS Table Names

PROC TREESPLIT assigns a name to each table it creates. You can use these names to refer to the table when you use the Output Delivery System (ODS) to select tables and create output data tables. Table 34.5 lists these names.

Table 34.5 ODS Tables Produced by PROC TREESPLIT

Table Name	Description	Statement	Option
BestConfiguration	Hyperparameters and objective function values for the best configuration	AUTOTUNE	Default
ConstraintHistory	Values of the various constraints at each iteration of the EGR algorithm	MITIGATEBIAS	Default
CostComplexity	Information about the number of leaves and the error for cost-complexity pruning when validation data are used	PROC TREESPLIT	PRUNINGTABLE
CrossValidateMLFitStat	Per-fold and average assessment metrics of *k*-fold cross validation	CROSSVALIDATION	Default
CrossValidationValues	Results for cost-complexity pruning based on cross validation	PROC TREESPLIT	CVCC
EvaluationHistory	Values of the hyperparameters and the objective function for all configurations	AUTOTUNE	Default / EVALHISTORY=ALL
GroupMetricsHistory	Metrics for accuracy and fairness for each subpopulation over each iteration of the EGR algorithm	MITIGATEBIAS	Default
IterationHistory	Values of the model accuracy and bias metric over each iteration of the EGR algorithm	MITIGATEBIAS	Default
MitigationInformation	Information about the options that are used for bias mitigation	MITIGATEBIAS	Default
ModelInfo	Information about the modeling environment	PROC TREESPLIT	Default

Table 34.5 *continued*

Table Name	Description	Statement	Option
NObs	Observation information	PROC TREESPLIT	Default
OutputCasTables	See the section "OutputCasTables Table" on page 1850	PROC TREESPLIT / OUTPUT	OUTMODEL= / Default
PredName	Predicted name information for interval targets	PROC TREESPLIT	PRINTTARGET
PredProbName	Predicted probability name information for nominal targets	PROC TREESPLIT	PRINTTARGET
PredIntoName	Predicted name information for nominal targets	PROC TREESPLIT	PRINTTARGET
RBAImportance	Random branch assignment variable importance	PROC TREESPLIT	RBAIMP
ReducedError	Information about the number of leaves and the error for reduced-error pruning when validation data are used	PROC TREESPLIT	PRUNINGTABLE
Timing	Breakdown of how long bias mitigation spent on the various tasks of training models	MITIGATEBIAS	Default
TreePerformance	Fit statistics for the selected tree	PROC TREESPLIT	Default
TreePlotTable	The table that is used to generate the WholeTreePlot and ZoomedTreePlot	PROC TREESPLIT	TREEPLOT
TunerInfo	Setup values used by the tuner	AUTOTUNE	Default
TunerResults	Values of the hyperparameters and the objective function for the default configuration (Iteration 0) and up to 10 best configurations found	AUTOTUNE	Default
TunerSummary	Statistics about the tuning process	AUTOTUNE	Default
TunerTiming	Total time spent on different tasks while tuning	AUTOTUNE	Default
VariableImportance	Variable importance	PROC TREESPLIT	Default
WeightHistory	Weights that are assigned by the EGR algorithm for each level of the response (target) variable and for each subpopulation	MITIGATEBIAS	Default

ODS Graphics

You can refer to every graph that is produced through ODS Graphics by name. The names of the graphs that PROC TREESPLIT generates are listed in Table 34.6, along with the relevant PLOTS= options.

Table 34.6 Graphs Produced by PROC TREESPLIT

ODS Graph Name	Plot Description	PLOTS= Option
CrossValidationASEPlot	Cross validation cost-complexity ASE plot when the response is continuous	CVCC
CrossValidationMisclassPlot	Cross validation cost-complexity misclassification plot when the response is categorical	CVCC
IterationHistoryPlot	Iteration history of the model accuracy and bias metric when the MITIGATEBIAS statement is used	
PruningPlot	Plot of the error sum of squares, misclassification rate, or cost complexity when it is used for final tree selection	PRUNEUNTIL
WholeTreePlot	Overview plot of final tree	WHOLETREE
ZoomedTreePlot	Detailed plot of portion of final tree	ZOOMEDTREE

INPUT and TARGET Statement Syntax

In addition to the syntax that is described in the CLASS and MODEL statement sections, PROC TREESPLIT supports INPUT/TARGET syntax that you might be familiar with from other procedures. The INPUT/TARGET syntax cannot be used together with the CLASS/MODEL syntax.

This syntax requires one TARGET statement and one or more INPUT statements. If you use this syntax, then the PROC TREESPLIT statement, the TARGET statement, and the INPUT statement are required. Depending on the options in those statements, specified variables can be interval or nominal. By default, numeric INPUT variables are treated as interval (or continuous) predictors, and character INPUT variables are treated as nominal (or categorical) predictors.

INPUT *variables* < / *option* > **;**
TARGET *variable* < / *option* > **;**

INPUT Statement

INPUT *variables* < / *option* > **;**

The INPUT statement specifies predictor *variables* for the decision tree or regression tree. The value of *variable* can be a range such as "var_1–var_1000" or the special "_ALL_" value to include all variables in the data tables. As with CLASS variables, all nominal INPUT variables are padded or truncated to 32 characters.

You cannot use an INPUT statement with a MODEL or CLASS statement.

You can specify the following *option*:

LEVEL=INTERVAL | NOMINAL

specifies whether the specified predictor *variables* are interval or nominal. You can specify the following values:

INTERVAL treats all numeric *variables* as interval predictors.

NOMINAL treats all *variables* as nominal predictors.

By default, numeric *variables* are treated as interval predictors, and character *variables* are treated as nominal predictors. Specifying LEVEL=NOMINAL forces all *variables* in that statement to be treated as nominal. You cannot specify LEVEL=INTERVAL for character variables.

TARGET Statement

TARGET *variable* < / *options* > **;**

The TARGET statement names the *variable* whose values PROC TREESPLIT predicts. Missing values in the target are ignored except during scoring.

You cannot use a TARGET statement with a MODEL or CLASS statement.

You can specify the following *option*:

LEVEL=INTERVAL | NOMINAL

specifies whether the specified response *variable* is interval or nominal. You can specify the following values:

INTERVAL treats the response as an interval variable and creates a regression tree.

NOMINAL treats the response as a nominal variable and creates a decision tree.

By default, LEVEL=INTERVAL for numeric variables and LEVEL=NOMINAL for categorical variables.

Examples: TREESPLIT Procedure

The examples in this chapter are available in the GitHub repository located at `https://github.com/sascommunities/doc-supplement-casstat`.

NOTE: Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections "Using CAS Sessions and CAS Engine Librefs" on page 7 and "Loading a SAS Data Set onto a CAS Server" on page 7 in Chapter 2, "Shared Concepts."

Example 34.1: Creating a Binary Classification Tree with Validation Data

(View the complete code for this example (treesplex4.sas) in the example repository.)

A common use of classification trees is to predict whether a mortgage applicant will default on a loan. The data table Hmeq, which is in the Sampsio library that SAS provides, contains observations for 5,960 mortgage applicants. A variable named Bad indicates whether the applicant, after being approved for a loan, paid off or defaulted on the loan.

This example uses Hmeq to build a tree model that is used to score the data and can be used to score data about new applicants. Table 34.7 describes the variables in Hmeq.

Table 34.7 Variables in the Home Equity (Hmeq) Data Table

Variable	Role	Level	Description
Bad	Response	Binary	1 = applicant defaulted on the loan or is seriously delinquent 0 = applicant paid off the loan
CLAge	Predictor	Interval	Age of oldest credit line in months
CLNo	Predictor	Interval	Number of credit lines
DebtInc	Predictor	Interval	Debt-to-income ratio
Delinq	Predictor	Interval	Number of delinquent credit lines
Derog	Predictor	Interval	Number of major derogatory reports
Job	Predictor	Nominal	Occupational category
Loan	Predictor	Interval	Requested loan amount
MortDue	Predictor	Interval	Amount due on mortgage
nInq	Predictor	Interval	Number of recent credit inquiries
Reason	Predictor	Binary	`'DebtCon'` = debt consolidation `'HomeImp'` = home improvement
Value	Predictor	Interval	Value of property
YoJ	Predictor	Interval	Years at present job

The response variable for the tree model is Bad, a classification variable that has two values (0 for payment of loan and 1 for default). The other variables are predictor variables for the model. The following statements load mylib.hmeq data into your CAS session and display the first 10 observations of the data table. For this example, the statements assume that your CAS engine libref is named mylib, but you can substitute any appropriately defined CAS engine libref.

```
/* Convert variable names to mixed case */
data mylib.hmeq;
   length Bad Loan MortDue Value 8 Reason Job $7
          YoJ Derog Delinq CLAge nInq CLNo DebtInc 8;
   set sampsio.hmeq;
run;

proc print data=mylib.hmeq(obs=10); run;
```

Output 34.1.1 shows the first 10 observations of mylib.hmeq.

Output 34.1.1 Partial Listing of the mylib.hmeq Data

Obs	Bad	Loan	MortDue	Value	Reason	Job	YoJ	Derog	Delinq	CLAge	nInq	CLNo	DebtInc
1	1	1100	25860	39025	HomeImp	Other	10.5	0	0	94.367	1	9	.
2	1	1500	.	.			.	.	.	.	.	.	.
3	1	1800	48649	57037	HomeImp	Other	5.0	3	2	77.100	1	17	.
4	1	2000	.	62250	HomeImp	Sales	16.0	0	0	115.800	0	13	.
5	1	2000	45000	55000	HomeImp	Other	3.0	0	0	86.067	2	25	.
6	1	2200	24280	34687	HomeImp	Other	.	0	1	300.867	0	8	.
7	1	2300	28192	40150	HomeImp	Other	4.5	0	0	54.600	1	16	.
8	1	2400	50000	73395	HomeImp	ProfExe	5.0	1	0	.	1	0	.
9	1	2400	.	17180	HomeImp	Other	.	0	0	14.567	3	4	.
10	1	2500	15000	20200	HomeImp		18.0	0	0	136.067	1	19	.

The following statements use the TREESPLIT procedure to create a classification tree:

```
ods graphics on;

proc treesplit data=mylib.hmeq maxdepth=5;
   class Bad Delinq Derog Job nInq Reason;
   model Bad = Delinq Derog Job nInq Reason CLAge CLNo
               DebtInc Loan MortDue Value YoJ;
   prune costcomplexity;
   partition fraction(validate=0.3 seed=123);
   code file='treesplexc.sas';
run;
```

The MAXDEPTH= option specifies that the maximum depth of the tree to be grown is 5.

Specifying Bad to the left of the equals sign in the MODEL statement indicates that it is the response variable.

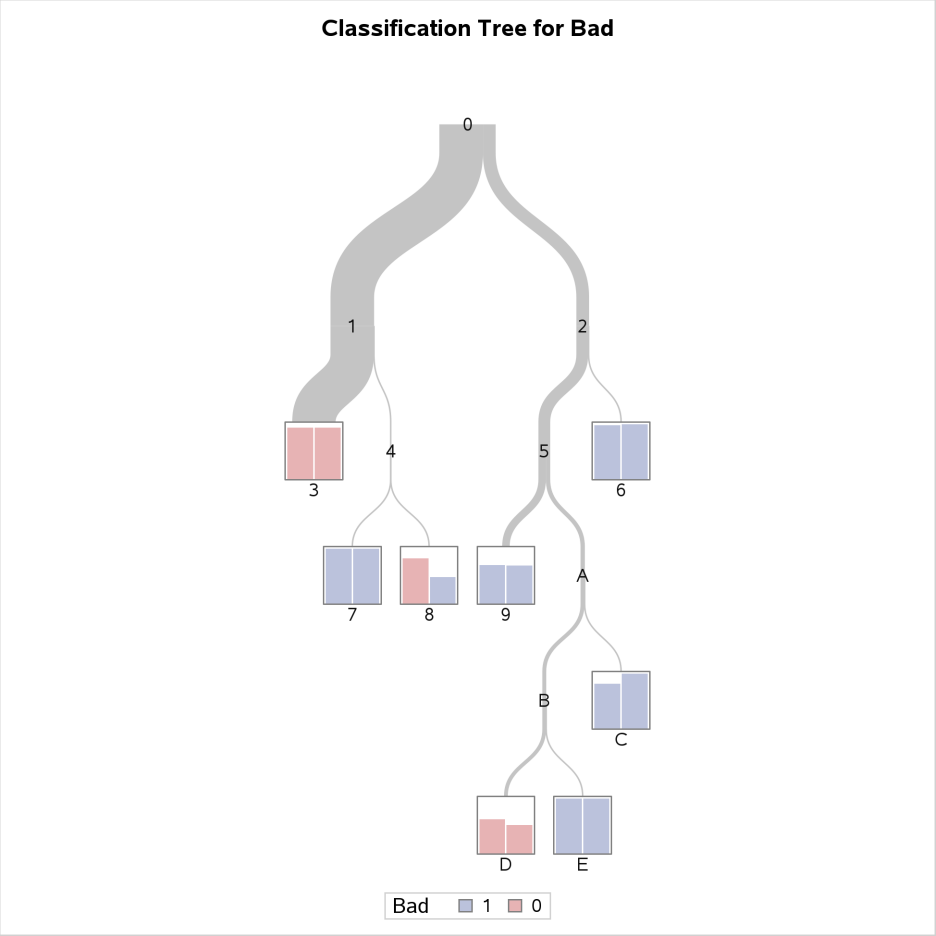
Because no GROW statement is included, PROC TREESPLIT defaults to using the entropy metric, which calculates the gain to grow the tree. The PRUNE statement requests cost-complexity pruning.

The PARTITION statement requests that the observations in Hmeq be partitioned into disjoint subsets for model training and validation. Observations are randomly selected for the validation subset with probability 0.3; the remaining observations are selected for the training subset.

The FILE= option in the CODE statement requests that SAS DATA step score code be saved to a file named *treesplexc.sas*.

The tree diagram in Output 34.1.2 provides an overview of the full tree.

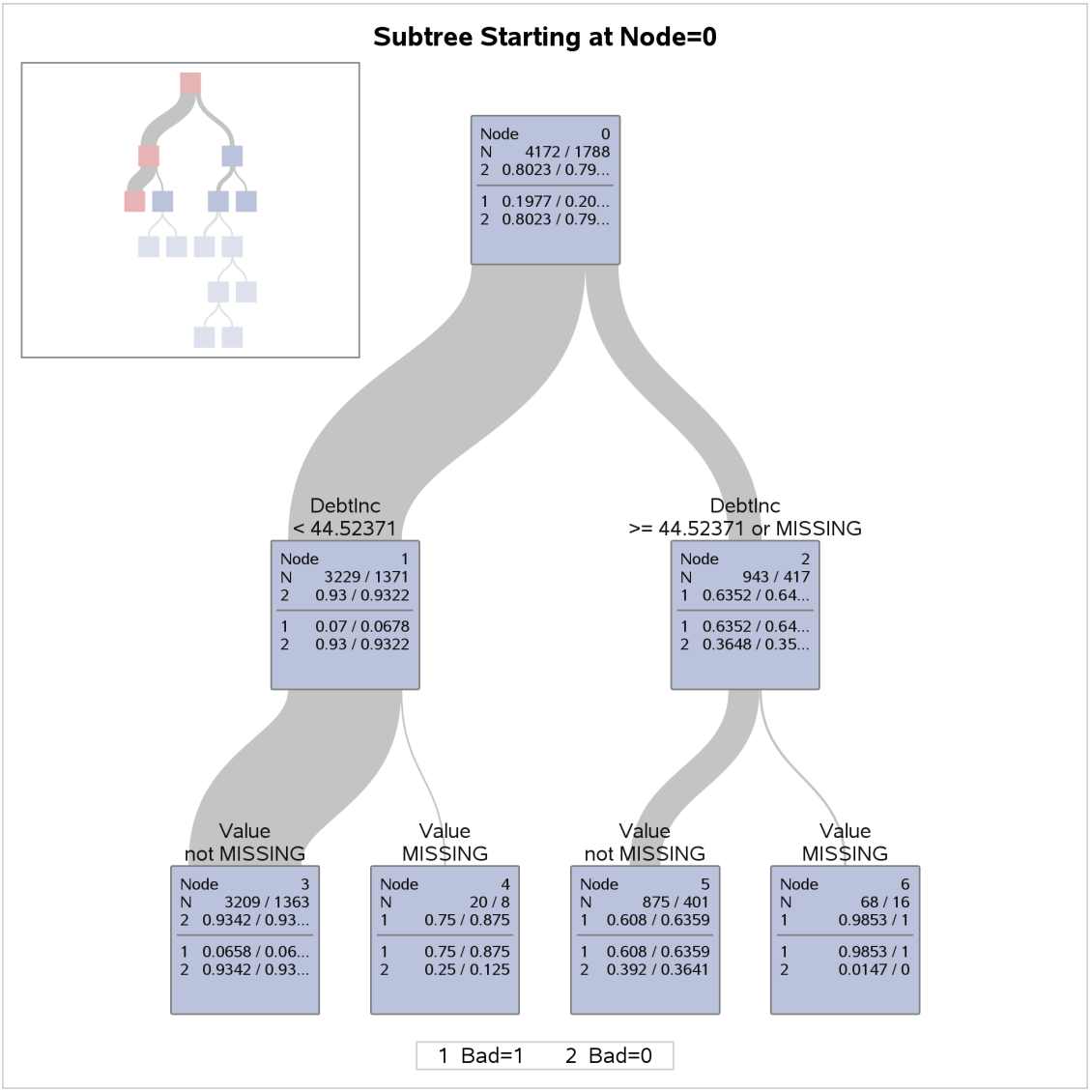
Output 34.1.2 Overview Diagram of Final Tree



The observations in terminal nodes are assigned a prediction of Bad=0 or Bad=1, as shown by the color in the plot legend. The first bar in a leaf node shows the proportion of the responses that match the prediction of Bad=0 or Bad=1 in the training partition, and the second bar in a leaf node shows the proportion of responses that match in the validation partition. The thickness of the line indicates which nodes have more total observations.

The tree diagram in Output 34.1.3 is a detailed view of the top portion of the tree. You can use the PLOTS= option in the PROC TREESPLIT statement to control which nodes are displayed.

Output 34.1.3 Detailed Tree Diagram



By default, this view provides detailed splitting information about the first three levels of the tree, including the root level. The splitting rule above each node shows the splitting variable and splitting value; the rule determines which observations from the parent node are included in the node. The first row of the table inside the node provides the node identifier. The second row provides the number of training observations, followed by a backslash, and then the number of validation observations. The third row shows the predicted response for observations in that node if classification occurs at that point, along with the proportion of training observations with that observed response. This is separated from the proportion of validation observations by a backslash. Note that the legend shows what actual value of the response variable is represented by the value shown in the node.

Output 34.1.4 displays the fit statistics for the final tree.

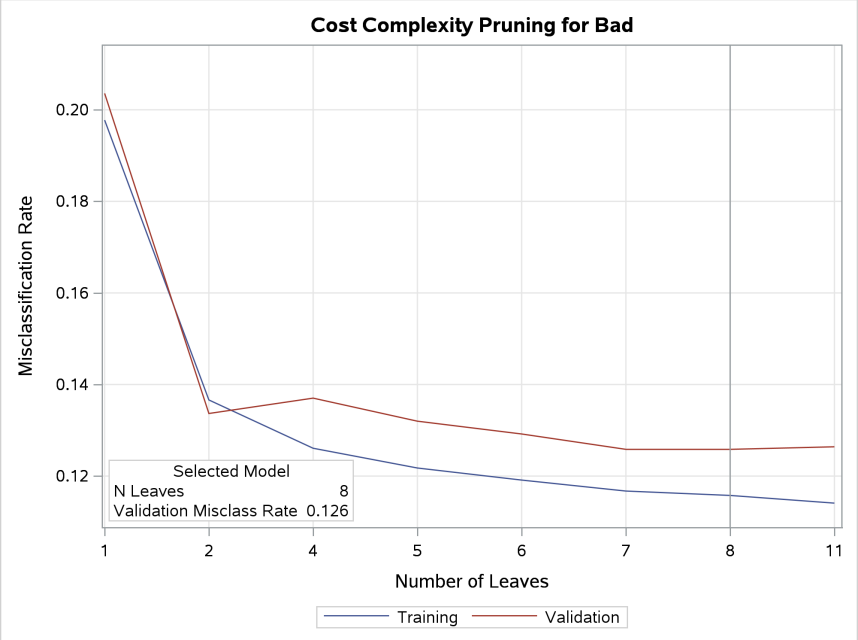
Output 34.1.4 Tree Performance

The TREESPLIT Procedure

Fit Statistics for Selected Tree		
	Number of Leaves	Misclassification Rate
Training	8	0.1158
Validation	8	0.1258

Output 34.1.5 displays the pruning plot.

Output 34.1.5 Pruning Plot



This plot displays the misclassification rates for the training and validation data as the tree is pruned. The vertical line shows the selected final tree that has the lowest misclassification rate for the validation data.

Creating Score Code and Scoring New Data

In addition to seeing information about the tree model, you might be interested in applying the model to predict the response variable in other data tables where the response is unknown. You can use the score code file *treesplexc.sas* (which was created by the FILE= option in the CODE statement) to score new data anywhere that you can run SAS DATA step code. The following is an example of using the score code file *treesplexc.sas* to score the data in Hmeq and save the results in a SAS data table named Scored:

```
data scored;
   set sampsio.hmeq;
   %include 'treesplexc.sas';
run;
```

Output 34.1.6 shows a partial listing of Scored.

Output 34.1.6 Partial Listing of the Scored Hmeq Data

Obs	BAD	LOAN	MORTDUE	VALUE	REASON	JOB	YOJ	DEROG	DELINQ	CLAGE	NINQ
1	1	1100	25860	39025	HomeImp	Other	10.5	0	0	94.367	1
2	1	1300	70053	68400	HomeImp	Other	7.0	0	2	121.833	0
3	1	1500	13500	16700	HomeImp	Other	4.0	0	0	149.467	1
4	1	1500	.	.			.	.	.	.	.
5	0	1700	97800	112000	HomeImp	Office	3.0	0	0	93.333	0
6	1	1700	30548	40320	HomeImp	Other	9.0	0	0	101.466	1
7	1	1800	48649	57037	HomeImp	Other	5.0	3	2	77.100	1
8	1	1800	28502	43034	HomeImp	Other	11.0	0	0	88.766	0
9	1	2000	32700	46740	HomeImp	Other	3.0	0	2	216.933	1
10	1	2000	.	62250	HomeImp	Sales	16.0	0	0	115.800	0

Obs	CLNO	DEBTINC	_leaf_id_	I_Bad	P_Bad1	P_Bad0
1	9	.	9	1	0.70916	0.29084
2	14	.	9	1	0.70916	0.29084
3	10	.	9	1	0.70916	0.29084
4	.	.	6	1	0.98529	0.01471
5	14	.	9	1	0.70916	0.29084
6	8	37.1136	3	0	0.06575	0.93425
7	17	.	9	1	0.70916	0.29084
8	8	36.8849	3	0	0.06575	0.93425
9	12	.	13	0	0.36918	0.63082
10	13	.	9	1	0.70916	0.29084

The data table contains the 13 original variables and 4 new variables that are created by the score code. The variable P_BAD1 is the proportion of training observations in this leaf for which BAD=1; this variable can be interpreted as the probability of default. The variable I_BAD indicates the predicted value of BAD for the observation.

You can use the preceding statements to score new data by including the new data table in place of Hmeq in the SET statement. The new data table must contain the same variables as the data that are used to build the tree model, but not the unknown response variable that you now want to predict.

Example 34.2: Creating a Regression Tree

(View the complete code for this example (treesplex3.sas) in the example repository.)

This example performs an analysis in which a linear regression model is fit. You can alternatively fit a regression tree to predict the salaries of Major League Baseball players based on their performance measures from the previous season by using almost identical code. Regression trees are piecewise constant models that, for relatively small data tables such as Sashelp.Baseball, provide succinct summaries of how the predictor variables determine the predictions. These models are usually easier to interpret than linear regression models. The Sashelp.Baseball data table contains salary and performance information for Major League Baseball players (excluding pitchers) who played at least one game in both the 1986 and 1987 seasons (Time Inc. 1987). You can load the Sashelp.Baseball data set into your CAS session by naming your CAS engine libref in the first statement of the following DATA step:

```
data mylib.baseball;
   set sashelp.baseball;
run;
```

The following statements create a regression tree model:

```
ods graphics on;

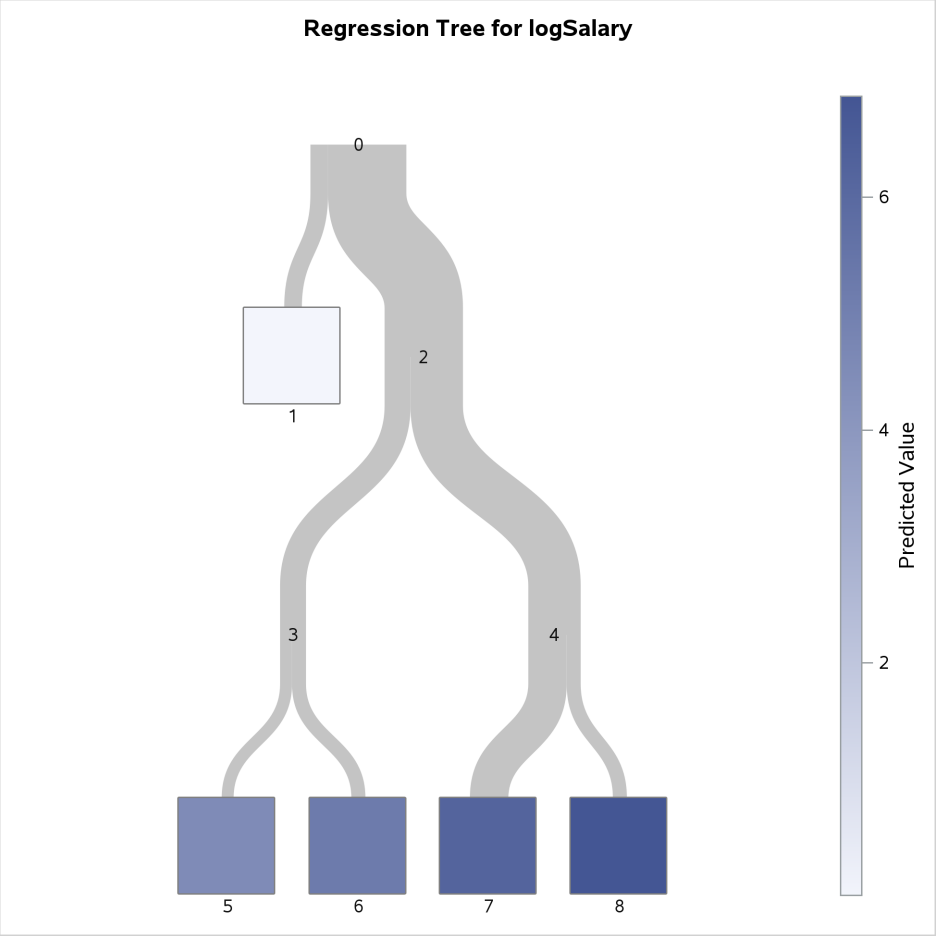
proc treesplit data=mylib.baseball maxdepth=3;
   class league division;
   model logSalary = nAtBat nHits nHome nRuns nRBI nBB
                     yrMajor crAtBat crHits crHome crRuns crRbi
                     crBB league division nOuts nAssts nError;
   output out=mylib.treesplout;
   prune none;
run;
```

Because no GROW statement is specified, the tree is grown using the RSS criterion by default. Because PRUNE none is specified, no pruning is performed. The OUTPUT statement requests generation of the data table mylib.treesplout, which contains the predicted salary from the tree model for each observation.

Much of the output for a regression tree is identical to the output for a classification tree. Where there are differences, tables and plots are displayed and discussed on the following pages.

Output 34.2.1 displays the full regression tree.

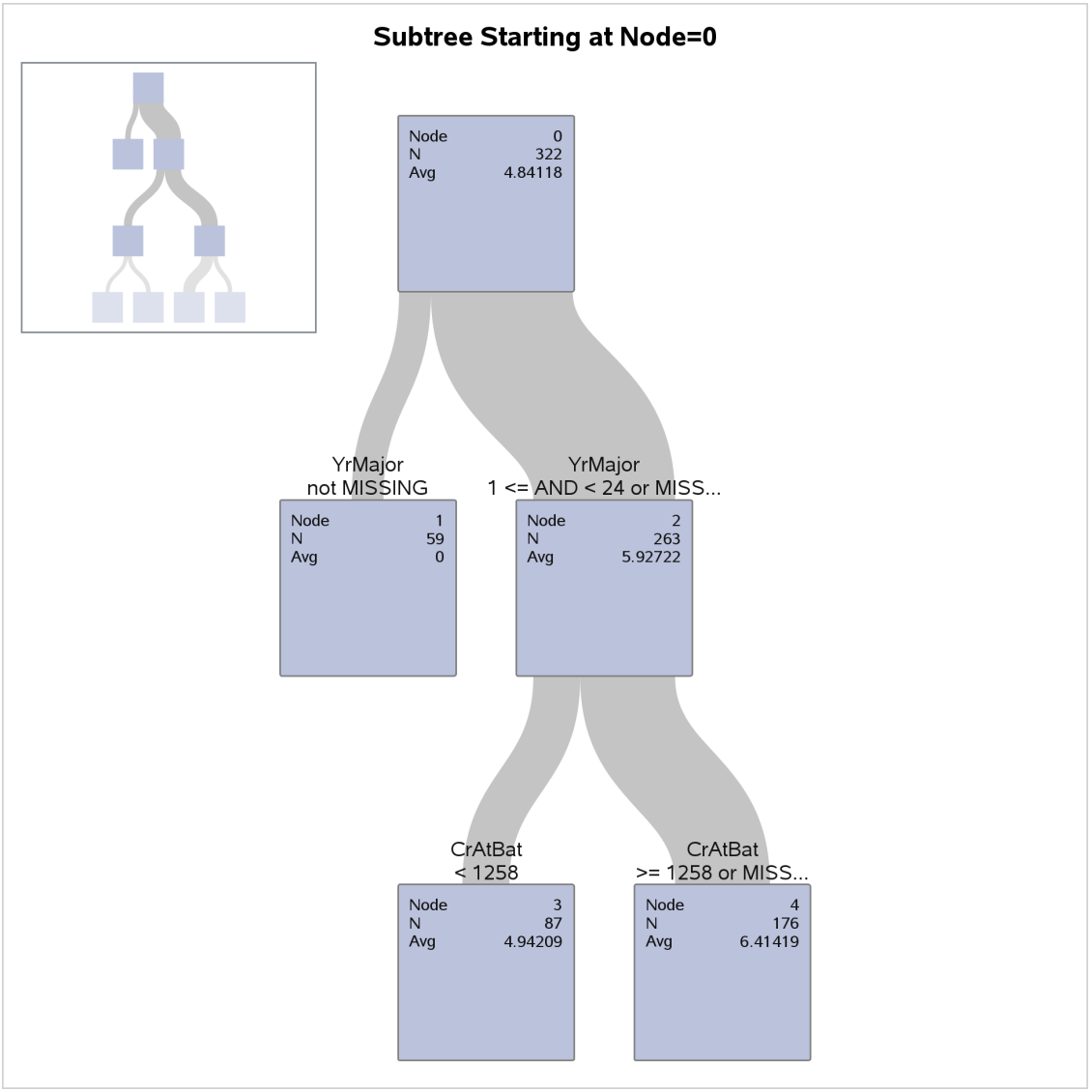
Output 34.2.1 Overview Diagram of Regression Tree



The final selected tree has eight leaves. In a regression tree, the shade of the leaves represents the predicted response value, which is the average observed logSalary for the observations in that leaf. Node 7 has the lowest predicted response value, indicated by the lightest shade of blue, and node E has the highest, indicated by the dark shade.

Output 34.2.2 shows details of the first three levels of the tree, including the root node.

Output 34.2.2 Detailed Diagram of Regression Tree



As in Output 34.1.3, this diagram displays split variables and split values for the nodes, along with the exact predicted response value, which is the average observed response, in each node.

Output 34.2.3 displays the fit statistic for the final regression tree (the only fit statistic provided for a regression tree is the ASE).

Output 34.2.3 Regression Tree Performance

The TREESPLIT Procedure

Fit Statistics for Selected Tree		
	Number of Leaves	Average Square Error
Training	5	0.2156

Output 34.2.4 is a partial display of the mylib.treesplout data table that is created when you specify the OUTPUT statement.

Output 34.2.4 Scored Predictor Data Table

Obs	P_logSalary	_DT_PredStd_	_LeafID_	_Residual_
1	4.56042	0.24843	5	.
2	6.24707	0.54820	7	0.03246
3	4.56042	0.24843	5	0.31193
4	6.87289	0.48707	8	-0.13018
5	6.24707	0.54820	7	-0.06285
6	4.56042	0.24843	5	.
7	6.87289	0.48707	8	.
8	4.56042	0.24843	5	-0.18451
9	6.24707	0.54820	7	-0.40794
10	6.24707	0.54820	7	-0.37300

The variable P_logSalary contains the predicted salaries on the log scale. Note that all observations in the same leaf have the same predicted response. The OUT= data table can contain additional variables from the DATA= data table if you specify them by using the COPYVARS= option.

Example 34.3: Assessing Variable Importance

(View the complete code for this example (treesplex5.sas) in the example repository.)

This example creates a classification tree model to determine important variables (parameters) during the manufacture of a semiconductor device. Some of the variables that are involved in the manufacturing process are: gTemp, the growth temperature of substrate; aTemp, the annealing temperature; Rot, rotation speed; Dopant, the atom used during device growth; and Usable, which indicates whether the device is usable.

The following statements create a data table named mylib.MBE_Data, which contains measurements for 20 devices:

```
data mylib.MBE_Data;
   label gTemp  = 'Growth Temperature of Substrate';
   label aTemp  = 'Anneal Temperature';
   label Rot    = 'Rotation Speed';
   label Dopant = 'Dopant Atom';
   label Usable = 'Experiment Could Be Performed';

   input gTemp aTemp Rot Dopant $ 39-40 Usable $ 47-54;
   datalines;
   384.614    633.172    1.01933      C       Unusable
   363.874    512.942    0.72057      C       Unusable
   397.395    671.179    0.90419      C       Unusable
   389.962    653.940    1.01417      C       Unusable
   387.763    612.545    1.00417      C       Unusable
   394.206    617.021    1.07188      Si      Usable
   387.135    616.035    0.94740      Si      Usable
   428.783    745.345    0.99087      Si      Unusable
   399.365    600.932    1.23307      Si      Unusable
```

```
455.502    648.821    1.01703      Si      Unusable
387.362    697.589    1.01623      Ge      Usable
408.872    640.406    0.94543      Ge      Usable
407.734    628.196    1.05137      Ge      Usable
417.343    612.328    1.03960      Ge      Usable
482.539    669.392    0.84249      Ge      Unusable
367.116    564.246    0.99642      Sn      Unusable
398.594    733.839    1.08744      Sn      Unusable
378.032    619.561    1.06137      Sn      Usable
357.544    606.871    0.85205      Sn      Unusable
384.578    635.858    1.12215      Sn      Unusable
;
```

The following statements create the tree model:

```
proc treesplit data=mylib.MBE_Data maxdepth=6;
   class Usable Dopant;
   model Usable = gTemp aTemp Rot Dopant;
   prune none;
run;
```

Output 34.3.1 shows the "Variable Importance" table.

Output 34.3.1 Variable Importance

The TREESPLIT Procedure

Variable Importance

Training

Variable	Importance	Relative Importance	Count
Dopant	2.7000	1.0000	1
aTemp	1.0500	0.3889	1

This table shows that the predictor Dopant has the largest importance value. This means that the atom used during device growth is the most important consideration in determining the usability of the sample. The column Count indicates how many times the variable was used in for splits within the decision tree.

Example 34.4: Mitigating Bias of a Classification Tree

(View the complete code for this example (treesplex6.sas) in the example repository.)

This example creates a classification tree model while minimizing the demographic parity for the Job variable. The response variable for the tree model is Bad, a classification variable that has two values (0 for payment of loan and 1 for default). The other variables are predictor variables for the model.

The following statements load the data table mylib.hmeq into your CAS session. For this example, the statements assume that your CAS engine libref is named mylib, but you can substitute any appropriately defined CAS engine libref.

```
data mylib.hmeq;
   set sampsio.hmeq;
run;
```

The following statements use the TREESPLIT procedure to create a classification tree without minimizing the demographic parity:

```
proc treesplit data=mylib.hmeq;
   class Bad Reason;
   model Bad = CLAge CLNo DebtInc Derog
               Loan MortDue Reason Value;
   prune off;
   output out=mylib.scored copyvars=(_ALL_);
   saveState rstore=mylib.treeModel;
run;
```

Specifying Bad to the left of the equals sign in the MODEL statement indicates that it is the response variable. The PRUNE statement specifies no pruning. The OUTPUT statement generates the data table mylib.scored, which contains the predicted mortgage default from the tree model for each observation and all variables from the input data table as copy variables. The SAVESTATE statement creates an analytic store for the model and saves it as a binary object in the data table mylib.treeModel.

Output 34.4.1 displays the fit statistics for the tree model.

Output 34.4.1 Fit Statistics Table

The TREESPLIT Procedure

Fit Statistics for Selected Tree

	Number of Leaves	Misclassification Rate
Training	63	0.1084

After you train a classification tree model, you might want to view the demographic parity of the trained model. You can do this by running the ASSESSBIAS procedure.

The following statements use PROC ASSESSBIAS to assess the bias measurements of the model:

```
proc assessbias data=mylib.scored;
   target bad / event="1" level=nominal;
   var P_Bad1;
   fitstat pvar=P_BAD0 / pevent="0";
   sensitiveVar Job;
run;
```

For more information about PROC ASSESSBIAS, see Chapter 3, "ASSESSBIAS Procedure" (*SAS Viya: Machine Learning Procedures*). Output 34.4.2 displays the bias metric values of the model. PROC ASSESSBIAS also produces other output that is not shown here.

Output 34.4.2 Bias Metrics Table

The ASSESSBIAS Procedure

Bias Metrics					
Bias Statistic	Bias Statistic Label	Bias Statistic Value	Base Level	Compare Level	Note
DemographicParity	Demographic Parity (Statistical Parity)	0.1476	Self		
PredictiveParity	Predictive Parity	0.1265	Self		
EqualAccuracy	Equal Accuracy	0.0804	Office	Sales	
EqualizedOdds	Equalized Odds	0.1534	Self	Sales	The maximum TPR difference is greater than the maximum FPR difference.
EqualOpportunity	Equal Opportunity	0.1534	Self	Sales	

In order to use the MITIGATEBIAS statement to reduce the demographic parity, you must specify the levels of the response (target) variable and the corresponding predicted variables. You can use the DESCRIBE statement in the ASTORE procedure to obtain this information. The following statements use PROC ASTORE to describe the model:

```
proc astore;
   describe rstore=mylib.treeModel;
run;
```

Output 34.4.3 lists the output variables that the model creates. The predicted variables are P_BAD1 and P_BAD0. Their corresponding response (target) levels are 1 and 0, respectively.

Output 34.4.3 Output Variables Table

The ASTORE Procedure

Output Variables			
Name	Length	Type	Label
P_BAD1	8	Num	Predicted: BAD=1
P_BAD0	8	Num	Predicted: BAD=0
I_BAD	12	Character	Into: BAD
WARN	4	Character	Warnings
_leaf_id_	8	Num	_leaf_id_

The following statements use PROC TREESPLIT to create a classification tree, while minimizing the demographic parity for Job by using the MITIGATEBIAS statement:

```
ods graphics on;

proc treesplit data=mylib.hmeq;
   class Bad Reason;
   model Bad = CLAge CLNo DebtInc Derog
               Loan MortDue Reason Value;
   prune off;
   saveState rstore=mylib.treeModel;
   mitigateBias pevents="1 0"
                pvars=(P_BAD1 P_BAD0)
```

```
                    seed=12345
                    sensitivevar=Job
                    targetEvent="1";
run;
```

The PEVENTS= option lists the levels of the response (target) variable Bad. The predicted variables correspond to the levels that are listed in the PVARS= option. The PVARS= option lists the predicted variables that contain the posterior probability for each level in model prediction that corresponds to the response (target) variable Bad. The SEED= option specifies the seed for the pseudorandom number generator to be 12345. The SENSITIVEVAR= option names the variable Job as the sensitive variable for the purpose of reducing the value of the bias measurement. The TARGETEVENT= option specifies the value 1 as the event of interest.

Output 34.4.4 displays the information about the options that are used for bias mitigation. Output 34.4.5 displays the model accuracy and bias metric values over each iteration of the EGR algorithm. Output 34.4.6 shows the iteration history, which is a graphical representation of the iteration history table. The MITIGATEBIAS statement also creates other tables, which are not shown here but are listed in the section "ODS Table Names" on page 1852.

Output 34.4.4 Mitigation Information Table

The TREESPLIT Procedure

Mitigation Information

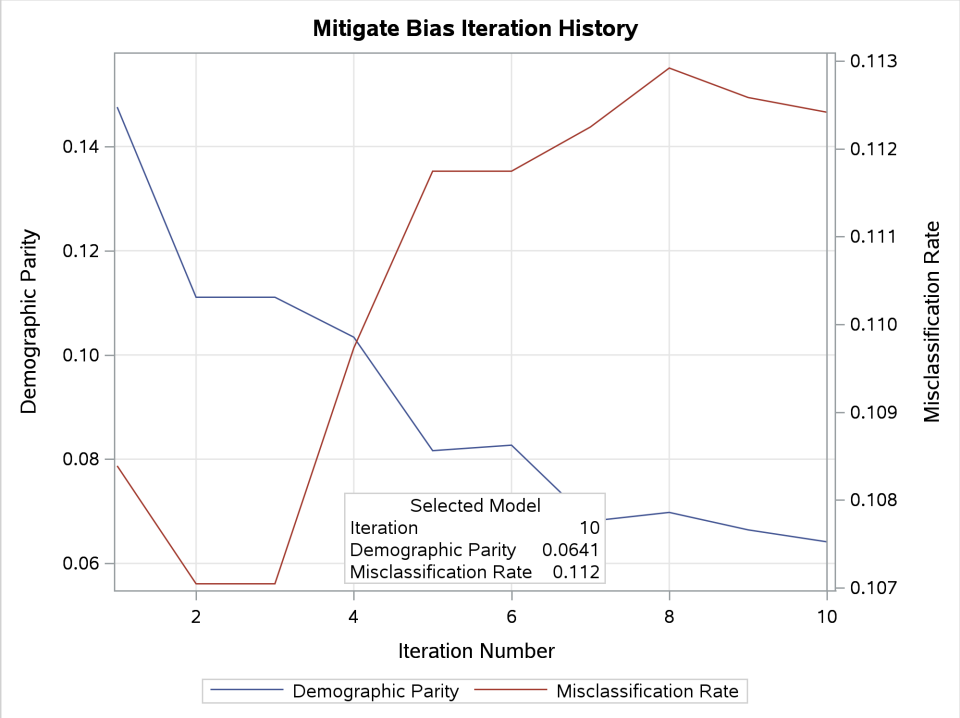
Description	Character Value	Value
Bias Metric	DEMOGRAPHICPARITY	.
Sensitive Variable	JOB	.
Response	BAD	.
Bound Value	100	100
Learning Rate	0.01	0.01
Maximum Iteration	10	10
Tolerance	0.005	0.005
Event Level	1	.

Output 34.4.5 Iteration History Table

EGR Iteration History

Iteration	Demographic Parity (Statistical Parity)	Misclassification Rate	Multi-Class Log Loss
1	0.147566	0.1084	0.2718
2	0.111078	0.1070	0.2698
3	0.111078	0.1070	0.2695
4	0.103414	0.1097	0.2769
5	0.081634	0.1117	0.2799
6	0.082688	0.1117	0.2833
7	0.068110	0.1122	0.2842
8	0.069785	0.1129	0.2895
9	0.066434	0.1126	0.2896
10	0.064139	0.1124	0.2891

Output 34.4.6 Iteration History Plot



Example 34.5: Tuning the Hyperparameters of a Decision Tree

(View the complete code for this example (treesplex7.sas) in the example repository.)

This example illustrates how you can use the AUTOTUNE statement in the TREESPLIT procedure to tune the hyperparameters of a decision tree. It uses the home equity data set Hmeq, which is in the Sampsio library.

The Hmeq data set contains 5,960 observations with 13 variables. The response variable, Bad, is a binary indicator to predict which clients will default on their loan.

The following statements load the data table Hmeq into your CAS session. For this example, the statements assume that your CAS engine libref is named mylib, but you can substitute any appropriately defined CAS engine libref.

```
data mylib.hmeq;
   set sampsio.hmeq;
run;
```

The following statements use PROC TREESPLIT with the AUTOTUNE statement to automatically tune the hyperparameters of a decision tree:

```
ods graphics on;

proc treesplit data=mylib.hmeq;
   class Bad Reason;
   model Bad = CLAge CLNo DebtInc Derog
```

```
            Loan MortDue Reason Value;
   prune off;
   autotune
      /* Tuning Parameters
      You do not need to specify any tuning parameters for the default
      tuning process. If you want to make adjustments to the default
      tuning process, uncomment the following block of code and change
      any of the tuning parameters' attributes.

      tuningParameters=(
         CRITERION   ( values=GAIN GINI GAINRATIO
                     CHAID CHISQUARE init=GAINRATIO  )
         NUMBIN      ( lb=20    ub=200   init=20  )
         MAXDEPTH    ( lb=1     ub=19    init=10  )
         MINLEAFSIZE ( values=1 5 10 20 40 80 160 320 init=5 EXCLUDE  )
      )
      */
   ;
   /* Remove this line to see all results */
   ods select BestConfiguration EvaluationHistoryPlot IterationHistoryPlot;
run;
```

The preceding statements produce the table and the plots shown in Output 34.5.1 through Output 34.5.3. The table in Output 34.5.1 displays the evaluation number, the values of the tuning parameters, and the error metric value for the best decision tree model that the tuner found. Note that the ODS SELECT statement limits the displayed results to a single table and two plots. You can remove this statement to display all tables and plots. For the full list of ODS tables that PROC TREESPLIT produces, see Table 34.5.

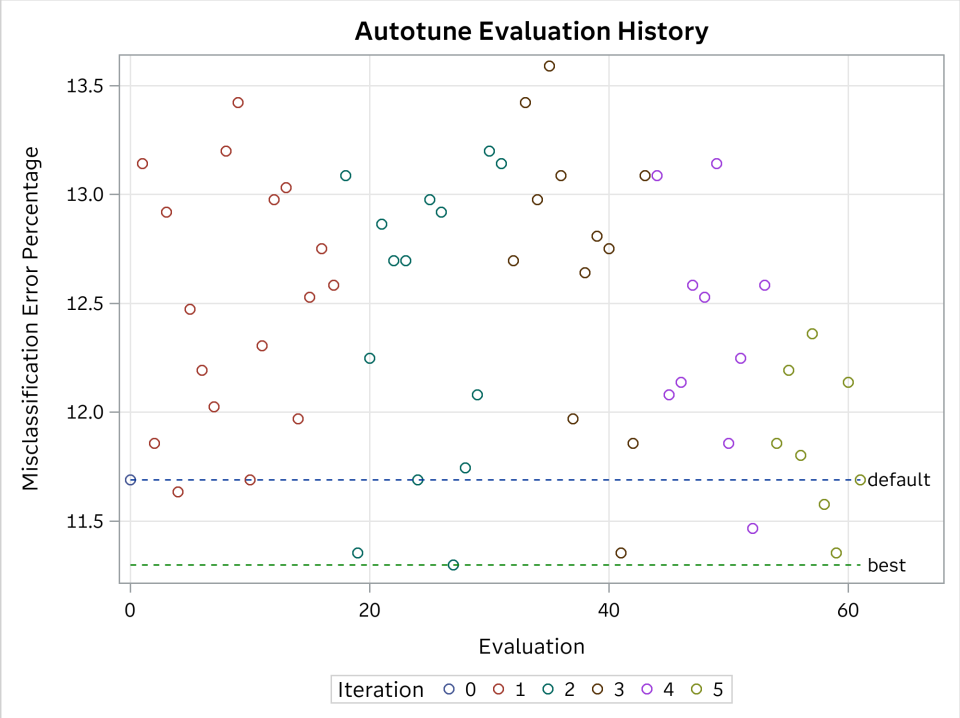
Output 34.5.1 Best Configuration Table

The TREESPLIT Procedure

Best Configuration	
Evaluation	3
Maximum Tree Levels	6
Maximum Bins	160
Criterion	GAIN
Misclassification Error Percentage	10.68

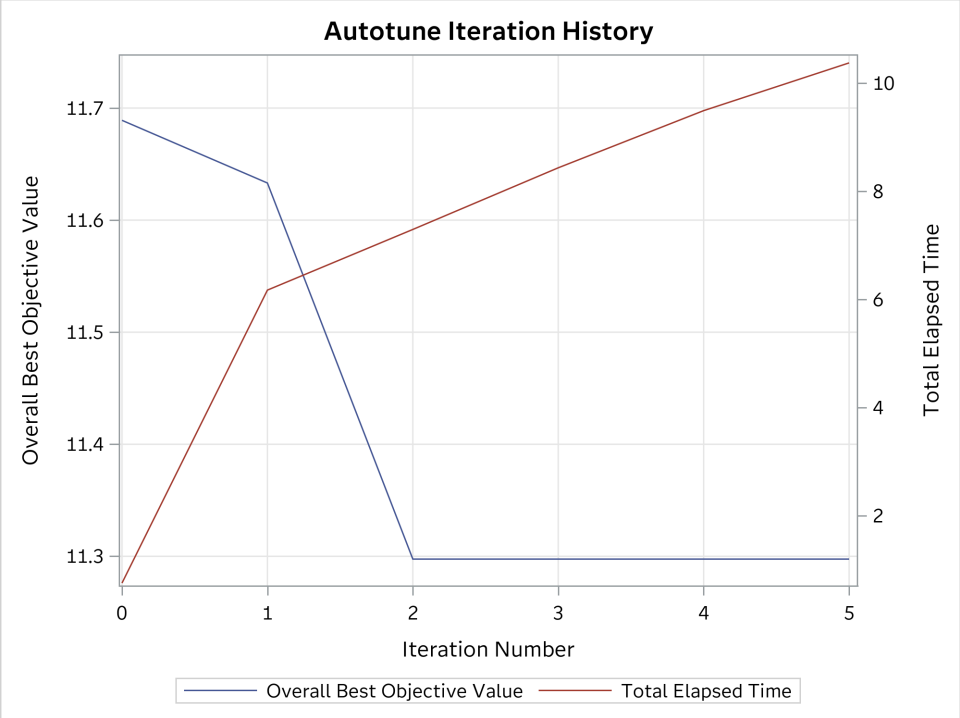
Output 34.5.2 displays a scatter plot of all configurations that the tuner tried. The objective values are shown on the Y axis, and the evaluation numbers are shown on the X axis.

Output 34.5.2 Evaluation History Plot



The plot in Output 34.5.3 displays how the best found objective value and the elapsed time changed with each iteration of the tuner.

Output 34.5.3 Iteration History Plot



References

Agarwal, A., Beygelzimer, A., Dudik, M., Langford, J., and Wallach, H. (2018). "A Reductions Approach to Fair Classification." In *Proceedings of the 35th International Conference on Machine Learning*, 60–69. PMLR.

Breiman, L., Friedman, J., Olshen, R. A., and Stone, C. J. (1984). *Classification and Regression Trees*. Belmont, CA: Wadsworth.

Hastie, T. J., Tibshirani, R. J., and Friedman, J. H. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd ed. New York: Springer-Verlag.

Kass, G. V. (1980). "An Exploratory Technique for Investigating Large Quantities of Categorical Data." *Journal of the Royal Statistical Society, Series C* 29:119–127.

Kuhn, M., and Johnson, K. (2013). *Applied Predictive Modeling*. New York: Springer.

Lichman, M. (2013). "UCI Machine Learning Repository." School of Information and Computer Sciences, University of California, Irvine. http://archive.ics.uci.edu/ml.

Neville, P. G., and Tan, P.-Y. (2014). "A Forest Measure of Variable Importance Resistant to Correlations." In *Proceedings of the 2014 Joint Statistical Meetings*. Alexandria, VA: American Statistical Association.

Quinlan, J. R. (1986). "Induction of Decision Trees." *Machine Learning* 1:81–106.

Quinlan, J. R. (1987). "Simplifying Decision Trees." *International Journal of Man-Machine Studies* 27:221–234.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. San Francisco: Morgan Kaufmann.

Time Inc. (1987). "What They Make." *Sports Illustrated* (April 20): 54–81.

Zhang, H., and Singer, B. H. (2010). *Recursive Partitioning and Applications*. 2nd ed. New York: Springer.

Chapter 35
VARIMPUTE Procedure

Contents

Overview: VARIMPUTE Procedure **1876**
Using CAS Sessions and CAS Engine Librefs 1876
Getting Started: VARIMPUTE Procedure **1877**
Syntax: VARIMPUTE Procedure **1878**
PROC VARIMPUTE Statement 1879
CODE Statement 1879
FREQ Statement 1880
INPUT Statement 1880
OUTPUT Statement 1881
WEIGHT Statement 1882
Details: VARIMPUTE Procedure **1882**
Obtaining the Statistics for Imputation 1882
Random Imputation 1882
Displayed Output 1883
Imputation Requests 1883
Imputed Values for Interval Variables 1883
Imputed Values for Nominal Variables 1883
ODS Table Names 1883
Examples: VARIMPUTE Procedure **1884**
Example 35.1: Imputation of Missing Values for HMEQ Data Table 1884

Overview: VARIMPUTE Procedure

The VARIMPUTE procedure performs variable imputation on a data table. Imputation is a common step in data preparation.

The VARIMPUTE procedure can replace numeric missing values with a specified value, with the mean or median of the nonmissing values, or with some random value between the minimum value and the maximum value of the nonmissing values.

The VARIMPUTE procedure can replace nominal missing values with a specified value and with the nonmissing value which has the most frequent appearance for that variable.

The VARIMPUTE procedure also supports formatted values including user defined formats.

When PROC VARIMPUTE calculates the mean, median, mode, or a random value, it ignores any observation that has a value that is less than or equal to 0 for any variable that is named in the FREQ statement.

Using CAS Sessions and CAS Engine Librefs

SAS Cloud Analytic Services (CAS) is the analytic server and associated cloud services in SAS Viya. This section describes how to create a CAS session and set up a CAS engine libref that you can use to connect to the CAS session. It assumes that you have a CAS server already available; contact your system administrator if you need help starting and terminating a server. This CAS server is identified by specifying the host on which it runs and the port on which it listens for communications. To simplify your interactions with this CAS server, the host information and port information for the server are stored as SAS option values that are retrieved automatically whenever this CAS server needs to be accessed. You can examine the host and port values for the server at your site by using the following statements:

```
proc options option=(CASHOST CASPORT);
run;
```

In addition to starting a CAS server, your system administrator might also have created a CAS session and a CAS engine libref for your use. You can define your own sessions and CAS engine librefs that connect to the CAS server as shown in the following statements:

```
cas mysess;
libname mylib cas sessref=mysess;
```

The CAS statement creates the CAS session named mysess, and the LIBNAME statement creates the mylib CAS engine libref that you use to connect to this session. It is not necessary to explicitly name the CASHOST and CASPORT of the CAS server in the CAS statement, because these values are retrieved from the corresponding SAS option values.

If you have created the mysess session, you can terminate it by using the TERMINATE option in the CAS statement as follows:

```
cas mysess terminate;
```

For more information about the CAS and LIBNAME statements, see the section “Introduction to Shared Concepts” on page 6 in Chapter 2, “Shared Concepts.”

Getting Started: VARIMPUTE Procedure

(View the complete code for this example at https://github.com/sascommunities/doc-supplement-casstat/tree/main/Examples/varimpute_gs.sas.)

NOTE: Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections "Using CAS Sessions and CAS Engine Librefs" on page 7 and "Loading a SAS Data Set onto a CAS Server" on page 7 in Chapter 2, "Shared Concepts."

The VARIMPUTE procedure can use any or all of six methods to impute numeric missing values. This example uses all six imputation methods to manipulate a data table. The input data table mylib.hmeq includes information about fictitious mortgages. Each observation represents an applicant for a home equity loan, and all applicants have an existing mortgage.

You can load the sampsio.hmeq data set into your CAS session in the following DATA step:

```
data mylib.hmeq;
   set sampsio.hmeq;
run;
```

The following statements perform the imputation:

```
proc varimpute data=mylib.hmeq seed=18000;
   input clage /ctech=mean;
   input delinq/ctech=median;
   input ninq/ctech=random;
   input reason/ntech=mode;
   input job/ntech=value valuescharacter=Office;
   input debtinc yoj/ctech=value cvalues=50,100;
   output out=mylib.out;
run;
```

The SEED= option specifies 18,000 as the random seed to be used in the random imputation process. The first INPUT statement (which includes the CTECH=MEAN option) requests that mean imputation be used for the variable clage. The second INPUT statement (which includes the CTECH=MEDIAN) option requests that median imputation be used for the variable delinq. The third INPUT statement (which includes the CTECH=RANDOM option) requests that random imputation be used for the variable ninq. The fourth INPUT statement (which includes the NTECH=MODE option) requests that mode imputation be used for the variable reason. The fifth INPUT statement (which includes the NTECH=VALUE option) requests that Office be imputed as the value of the variable job. The sixth INPUT statement (which includes the CTECH=VALUE) option requests that 50 be imputed as the value of variable debtinc and 100 be imputed as the value of variable yoj. The OUTPUT statement creates a new data table to contain the imputed values; it is shown in Output 35.1 and Output 35.2. The Variable column shows the original variable names from the input data table. The Imputation Method column shows the types of imputation methods: value (for character or numeric variables), median, random (between the minimum value and the maximum value of the nonmissing values), mean, and mode. For random imputation, the last column does not show any number, because the imputation is done by using a random seed. For other imputation methods, the last column shows the imputation values that are used to replace missing values. The Result Variable column shows the new variable names in the output data table.

Figure 35.1 Imputed Values for Interval Variables

The VARIMPUTE Procedure

Imputed Values for Interval Variables

Variable	Imputation Method	Result Variable	N	Number of Missing	Imputed Value
CLAGE	Mean	IM_CLAGE	5652	308	179.766
DELINQ	Median	IM_DELINQ	5380	580	0
NINQ	Random	IM_NINQ	5450	510	
DEBTINC	Value	IM_DEBTINC	4693	1267	50
YOJ	Value	IM_YOJ	5445	515	100

Figure 35.2 Imputed Values for Nominal Variables

Imputed Values for Nominal Variables

Variable	Imputation Method	Result Variable	N	Number of Missing	Imputed Value
REASON	Mode	IM_REASON	5708	252	DebtCon
JOB	Value	IM_JOB	5681	279	OFFICE

Syntax: VARIMPUTE Procedure

The following statements are available in the VARIMPUTE procedure:

PROC VARIMPUTE DATA=*libref.data-table* < *options* > **;**
CODE FILE=*filename* **;**
FREQ *variable* **;**
INPUT *variables* < */options* > **;**
OUTPUT OUT=*libref.data-table* < *option* > **;**
WEIGHT *variable* **;**

The PROC VARIMPUTE and INPUT statements are required. The INPUT statement can appear multiple times. If the same variable is specified in more than one INPUT statement, the imputation method specified in the first INPUT statement (or the default imputation method if the first statement does not specify a method) is applied.

PROC VARIMPUTE Statement

PROC VARIMPUTE DATA=*libref.data-table* < *options* > **;**

The PROC VARIMPUTE statement invokes the procedure.

You can specify the following *options*:

DATA=*libref.data-table*
names the input data table for PROC VARIMPUTE to use. The default is the most recently created data table. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to the data, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 1876.

data-table specifies the name of the input data table.

NTHREADS=*number-of-threads*
specifies the number of threads to be used. The default is the CPU count on each node.

SEED=*random-seed*
specifies an integer that is used to start the pseudorandom number generator. If you do not specify a seed or you specify a value less than or equal to 0, the seed is generated from reading the time of day from the computer's clock. If your specified seed is out of the range of an integer, the computer's clock time is used as the seed. The SEED= option enables you to reproduce the same sample output.

CODE Statement

CODE < *options* > **;**

The CODE statement writes SAS DATA step code for computing predicted values of the fitted model to a file or to a table. To score new data, you can then include the file in a DATA step, or you can specify the table in the **`runCodeTable`** action in the **`dataStep`** action set (for more information, see *SAS Viya: System Programming Guide*).

Table 35.1 summarizes the *options* available in the CODE statement.

Table 35.1 CODE Statement Options

Option	Description
COMMENT	Adds comments to the generated code
FILE=	Names the file in which to save the generated code
FORMATWIDTH=	Specifies the numeric format width for the regression coefficients
INDENTSIZE=	Specifies the number of spaces to indent the generated code
LABELID=	Specifies a number used to construct names and labels
LINESIZE=	Specifies the line size for the generated code
NOTRIM	Compares formatted values, including blank padding
OUT=	Names an output table in which to save the generated code

For more information about the syntax of the CODE statement, see the section "CODE Statement" on page 13 in Chapter 2, "Shared Concepts."

FREQ Statement

FREQ *variable* **;**

The *variable* in the FREQ statement identifies a numeric variable in the input data table that contains the frequency of occurrence of each observation. PROC VARIMPUTE treats each observation as if it appears f times, where f is the value of the FREQ *variable* for the observation. If f is not an integer, it is truncated to an integer. If f is less than 1 or missing, the observation is not used in the analysis. When you omit the FREQ statement, each observation is assigned a frequency of 1.

INPUT Statement

INPUT *variables* < */options* > **;**

The INPUT statement names one or more input *variables* and specifies the method that is used to impute missing values.

You can specify the following *options*:

CONTINUOUSTECH=*imputation-method*

CTECH=*imputation-method*

specifies the imputation method for interval variables.

You can choose from the following methods:

MEAN specifies that missing values of each variable are replaced by the algebraic mean of the nonmissing values of that variable.

MEDIAN specifies that missing values of each variable are replaced by the algebraic median of the nonmissing values of that variable.

RANDOM specifies that missing values of each variable are replaced by a random value that is drawn between the minimum value and the maximum values of the nonmissing values of that variable.

VALUE CVALUES=*value-list* specifies that missing values for each numeric input variable are replaced by the corresponding numeric value in the value-list.

By default, CONTINUOUSTECH=MEAN.

NOMINALTECH=*imputation-method*

NTECH=*imputation-method*

specifies the imputation method for nominal variables.

You can choose from the following *imputation-methods*:

MODE replaces missing values of each variable by the nonmissing value that appears most frequently for that variable.

VALUE VALUESNUMERIC=*value-list* replaces missing values for each numeric input variable by the corresponding numeric value in the *value-list*.

VALUE VALUESCHARACTER=*value-list* replaces missing values for each character input variable by the corresponding character value in the *value-list*.

By default, NOMINALTECH=MODE.

OUTPUT Statement

OUTPUT OUT=*libref.data-table* < *option* > **;**

The OUTPUT statement creates an output data table to contain the results of PROC VARIMPUTE.

You must specify the following option:

OUT=*libref.data-table*

names the output data table for PROC VARIMPUTE to use. You must specify this option before any other options. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to where the data table is to be stored, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 1876.

data-table specifies the name of the output data table.

The output data table contains the scored data.

You can also specify the following *option*:

COPYVAR=*variable*

COPYVARS=(*variables***)**

lists one or more variables from the input data table to be transferred to the output data table.

WEIGHT Statement

WEIGHT *variable* **;**

The *variable* in the WEIGHT statement is used as a weight to perform a weighted analysis of the data. Observations that have nonpositive or missing weights are not included in the analysis. The values of the WEIGHT variable can be nonintegral and are not truncated. When you omit the WEIGHT statement, all observations that are used in the analysis are assigned a weight of 1.

Details: VARIMPUTE Procedure

Obtaining the Statistics for Imputation

PROC VARIMPUTE first computes the imputation value and then imputes with that value. The statistics mean, median, mode, minimum, and maximum are computed precisely. To compute the median, PROC VARIMPUTE uses the empirical distribution function. Let n be the number of nonmissing values for a variable, and let $x_1, x_2, \ldots, x_n$ represent the ordered values of the variable. Let the 50th percentile be y, $p = 0.5$, and $np = j + g$ where j is the integer part of np and g is the fractional part of np. Then, the median y is

$$\begin{aligned} y &= x_j \quad \text{if} \quad g = 0 \\ y &= x_{j+1} \quad \text{if} \quad g > 0 \end{aligned}$$

Random Imputation

If you specify CONTINUOUSTECH=RANDOM in the INPUT statement, PROC VARIMPUTE replaces missing values with a random value that is drawn between the minimum and maximum of the variable. For variable x, assume that the input data table is $\{x_i\}$, where $i = 1, 2, \ldots, n$. Let $\min(x) = \min_{i \in \{1 \ldots n\}} \{x_i\}$, and let $\max(x) = \max_{i \in \{1 \ldots n\}} \{x_i\}$. The random value is calculated as $R = \min(x) + (\max(x) - \min(x)) \times \text{rand}('\text{uniform}')$, where $\text{rand}('\text{uniform}')$ is a function that takes a seed as input and returns a random value from a uniform distribution between 0 and 1. You can specify the seed as a nonnegative integer in the SEED= option in the PROC VARIMPUTE statement. If you do not specify the SEED= option, the seed is generated from reading the time of day from the computer's clock.

Displayed Output

The following sections describe the output that PROC VARIMPUTE produces. The output is organized into various tables, which are discussed in their order of appearance.

Imputation Requests

The first table displayed lists the imputation method, the number of variables that are imputed, and the seed value for the random imputation.

Imputed Values for Interval Variables

The "Imputed Values for Interval Variables" table lists the names of the interval variables, the imputation method, the numbers of nonmissing and missing values for each variable, and the imputation values that replace missing values for interval variables.

Imputed Values for Nominal Variables

The "Imputed Values for Nominal Variables" table lists the names of the nominal variables, the imputation method, the numbers of nonmissing and missing values for each variable, and the imputation values that replace missing values for nominal variables.

ODS Table Names

Each table that the VARIMPUTE procedure creates has a name associated with it. You must use this name to refer to the table when you use ODS statements. The name of each table and a short description of the contents are listed in Table 35.2.

Table 35.2 ODS Tables Produced by PROC VARIMPUTE

Table Name	Description	Statement	Option
ImputeInfo	Imputation method, number of variables that are imputed, and seed for the pseudorandom number generator	PROC VARIMPUTE	Default
VarImputeInfo (Interval Variables)	Variables, imputation method, number of nonmissing values, number of missing values, and imputation values	PROC VARIMPUTE	Default
VarImputeInfo (Nominal Variables)	Variables, imputation method, number of nonmissing values, number of missing values, and imputation values	PROC VARIMPUTE	Default

Examples: VARIMPUTE Procedure

The examples in this chapter are available in the GitHub repository located at https://github.com/sascommunities/doc-supplement-casstat.

NOTE: Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections "Using CAS Sessions and CAS Engine Librefs" on page 7 and "Loading a SAS Data Set onto a CAS Server" on page 7 in Chapter 2, "Shared Concepts."

Example 35.1: Imputation of Missing Values for HMEQ Data Table

(View the complete code for this example (varimpute_ex1.sas) in the example repository.)

This example demonstrates how to use PROC VARIMPUTE to perform imputation; it uses the same data table as in the section "Getting Started: VARIMPUTE Procedure" on page 1877.

Using DATA step, you can load the sampsio.hmeq data into the mylib.hmeq table:

```
data mylib.hmeq;
   set sampsio.hmeq;
   id = _N_;
run;
```

The following statements perform the imputation:

```
proc varimpute data=mylib.hmeq seed=12345;
   input derog clno/ctech=value cvalues=5,20;
   input value /ctech=mean;
   input job/ntech=mode;
   input mortdue /ctech=median;
   input ninq /ctech=random;
   input reason/ntech=value valuescharacter=unknown;
   output out=mylib.out1 copyvar=(id);
run;

data out2; set mylib.out1; run;
proc sort data=out2; by id; run;
proc print data=out2(firstobs=110 obs=124);
run;
```

Output 35.1.1 shows the number of variables for which the missing observations are imputed and the random seed value for the random imputation method.

Output 35.1.1 Imputation Requests

The VARIMPUTE Procedure

Imputation Method	Number of Variables	Seed
Mean	1	
Random	1	12345
Median	1	
Numeric Value	2	
Mode	1	
Character Value	1	

Output 35.1.2 shows the imputation results for interval variables.

Output 35.1.2 Imputed Values for Interval Variables

Imputed Values for Interval Variables

Variable	Imputation Method	Result Variable	N	Number of Missing	Imputed Value
VALUE	Mean	IM_VALUE	5848	112	101776
MORTDUE	Median	IM_MORTDUE	5442	518	65017
NINQ	Random	IM_NINQ	5450	510	
DEROG	Value	IM_DEROG	5252	708	5
CLNO	Value	IM_CLNO	5738	222	20

Output 35.1.3 shows the imputation results for nominal variables.

Output 35.1.3 Imputed Values for Nominal Variables

Imputed Values for Nominal Variables

Variable	Imputation Method	Result Variable	N	Number of Missing	Imputed Value
JOB	Mode	IM_JOB	5681	279	Other
REASON	Value	IM_REASON	5708	252	UNKNOWN

Output 35.1.4 shows 15 output observations with the new imputed values for those input variable values that were missing.

Output 35.1.4 Output Data Table

Obs	id	IM_VALUE	IM_MORTDUE	IM_NINQ	IM_DEROG	IM_CLNO	IM_JOB	IM_REASON
110	110	79852.00	68815	0.0000	0	14	Other	HomeImp
111	111	80027.00	72021	0.0000	2	5	Other	HomeImp
112	112	63869.00	53969	0.0000	0	30	ProfExe	DebtCon
113	113	60250.00	65017	10.9178	5	20	Other	UNKNOWN
114	114	64074.00	53091	0.0000	1	17	Office	DebtCon
115	115	79087.00	66127	0.0000	1	15	Other	HomeImp
116	116	77182.00	65017	13.8192	5	20	Other	UNKNOWN
117	117	66000.00	39000	1.0000	0	19	Other	HomeImp
118	118	79247.00	71082	0.0000	0	14	Other	HomeImp
119	119	49044.00	65017	0.0000	0	0	Other	HomeImp
120	120	101776.05	57000	1.0000	5	13	Other	HomeImp
121	121	46000.00	34184	2.0000	0	22	Mgr	HomeImp
122	122	87000.00	66517	0.0000	0	34	ProfExe	DebtCon
123	123	50000.00	45500	0.0000	0	18	Other	HomeImp
124	124	146870.00	118000	1.0000	2	20	ProfExe	HomeImp

Chapter 36
VARREDUCE Procedure

Contents

Overview: VARREDUCE Procedure **1888**
PROC VARREDUCE Features 1888
PROC VARREDUCE Compared to Other SAS Procedures 1889
Using CAS Sessions and CAS Engine Librefs 1889
Getting Started: VARREDUCE Procedure **1890**
Unsupervised Variable Selection 1893
Supervised Variable Selection 1894
Syntax: VARREDUCE Procedure **1895**
PROC VARREDUCE Statement 1895
CLASS Statement 1897
DISPLAY Statement 1898
DISPLAYOUT Statement 1899
FREQ Statement 1900
REDUCE Statement 1900
WEIGHT Statement 1902
Details: VARREDUCE Procedure **1902**
Missing Values 1902
Unsupervised Variable Selection 1903
Supervised Variable Selection 1904
Variable Selection for Regression 1904
Variable Selection for Classification 1904
Criteria Used in Model Selection 1905
Computational Method 1905
GLM Parameterization of Classification Variables and Effects 1907
Intercept 1907
Regression Effects 1908
Main Effects 1908
Interaction Effects 1908
Nested Effects 1909
Continuous-Nesting-Class Effects 1909
Continuous-by-Class Effects 1910
General Effects 1910
Displayed Output 1911
Number of Observations 1911
Selection Summary 1911
Selected Variables 1911

ODS Table Names 1912
Examples: VARREDUCE Procedure **1912**
Example 36.1: Analyzing Framingham Heart Study Data 1912
Example 36.2: Output a Correlation Matrix to a SAS Data File 1914
Example 36.3: Output the Correlation Matrix in LIL Format 1915
References **1917**

Overview: VARREDUCE Procedure

The VARREDUCE procedure performs both supervised and unsupervised variable selection in SAS Viya. You can use this procedure to read data in distributed form and perform variable selection in parallel.

The VARREDUCE procedure performs unsupervised variable selection by identifying a set of variables that jointly explain the maximum amount of data variance. Unlike principal component analysis (PCA), which reduces dimensionality by generating a set of new variables (variable extraction), PROC VARREDUCE reduces dimensionality by selecting a subset of the original variables (variable selection). Thus, this technique preserves model interpretation.

PROC VARREDUCE performs supervised variable selection by identifying a set of variables that jointly explain the maximum amount of variance contained in the response variables. It supports variable selection in both the regression setting and the classification (categorization) setting.

PROC VARREDUCE Features

The VARREDUCE procedure analyzes variance and reduces dimensionality by selecting the variables that contribute the most to the overall variance of the data in unsupervised variable selection (or to the overall variance of the response variables in supervised variable selection). The following list summarizes the basic features of this procedure:

- Variable selection is based on covariance analysis.
- Input data can be read in parallel.
- Computation of the Pearson correlation, covariance, or sums of squares and crossproducts matrix is distributed.
- Computation of the variable selection steps is distributed.
- All phases of analytic execution use of high degree of multithreading.
- Both supervised and unsupervised variable selection are supported.
- Multiple response variables are supported in variable selection for regression.
- The CLASS statement supports categorical inputs.
- The REDUCE statement supports main and interaction effects.

- The OUTCP option supports outputting a CORR, COV, or SSCP matrix.

PROC VARREDUCE Compared to Other SAS Procedures

This section compares the VARREDUCE procedure with the PCA and GENSELECT procedures in SAS Visual Statistics and with the FACTOR, PRINCOMP, GLMSELECT, and DISCRIM procedures in SAS/STAT software.

When PROC VARREDUCE performs unsupervised variable selection, it analyzes variance and reduces dimensionality by forward selection of the variables that contribute the most to the overall data variance. The output lists the variables in order of their contribution to data variance; the output can be used directly for reporting or for selecting variables for procedures that build models. In contrast, principal component analysis analyzes the variance and then projects the data space onto an orthogonal set of axes by a linear combination of the original variables. These new principal components best explain the data variance and can be used as input to procedures that build models. Principal component analysis can be done through the FACTOR, PRINCOMP, and PCA procedures, which reduce the number of inputs from the original set. The primary difference between principal component analysis (PCA) and PROC VARREDUCE is that PCA generates new variables, whereas PROC VARREDUCE reduces data dimensionality by selecting a subset of the original variables. This feature of PROC VARREDUCE is beneficial in applications where retaining the original variables is important for model exploration and interpretation.

When PROC VARREDUCE performs supervised variable selection, it analyzes the variance and reduces dimensionality by forward selection of the variables that contribute the most to explaining the overall variance of the response variables (targets). The output lists the variables in order of their contribution to explaining the response variance. The output can be used directly for reporting or for selecting variables for procedures that build models. When PROC VARREDUCE is used to perform supervised variable selection, it most resembles the GLMSELECT and GENSELECT procedures. However, PROC VARREDUCE allows multiple response variables, whereas PROC GLMSELECT and PROC GENSELECT do not. When the response variable is a classification variable and its levelization is done in a special format, PROC VARREDUCE analyzes the variance in the same way that linear discriminant analysis (LDA) does. LDA can be done through the DISCRIM procedure in SAS/STAT. Like principal component analysis, LDA generates new variables by linearly combining all original variables, whereas PROC VARREDUCE reduces data dimensionality by selecting a subset of the original variables.

Using CAS Sessions and CAS Engine Librefs

SAS Cloud Analytic Services (CAS) is the analytic server and associated cloud services in SAS Viya. This section describes how to create a CAS session and set up a CAS engine libref that you can use to connect to the CAS session. It assumes that you have a CAS server already available; contact your system administrator if you need help starting and terminating a server. This CAS server is identified by specifying the host on which it runs and the port on which it listens for communications. To simplify your interactions with this CAS server, the host information and port information for the server are stored as SAS option values that are retrieved automatically whenever this CAS server needs to be accessed. You can examine the host and port values for the server at your site by using the following statements:

```
proc options option=(CASHOST CASPORT);
run;
```

In addition to starting a CAS server, your system administrator might also have created a CAS session and a CAS engine libref for your use. You can define your own sessions and CAS engine librefs that connect to the CAS server as shown in the following statements:

```
cas mysess;
libname mylib cas sessref=mysess;
```

The CAS statement creates the CAS session named mysess, and the LIBNAME statement creates the mylib CAS engine libref that you use to connect to this session. It is not necessary to explicitly name the CASHOST and CASPORT of the CAS server in the CAS statement, because these values are retrieved from the corresponding SAS option values.

If you have created the mysess session, you can terminate it by using the TERMINATE option in the CAS statement as follows:

```
cas mysess terminate;
```

For more information about the CAS and LIBNAME statements, see the section "Introduction to Shared Concepts" on page 6 in Chapter 2, "Shared Concepts."

Getting Started: VARREDUCE Procedure

(View the complete code for this example at https://github.com/sascommunities/doc-supplement-casstat/tree/main/Examples/varreduce_gs.sas.)

NOTE: Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections "Using CAS Sessions and CAS Engine Librefs" on page 7 and "Loading a SAS Data Set onto a CAS Server" on page 7 in Chapter 2, "Shared Concepts."

This example uses a data table which contains 100 observations that contain one character variable (C), one classification variable (y), and 10 continuous variables (x1–x10). This data table is used for both of the getting-started examples in the following sections.

The following DATA step stores the variables in the mylib.getStarted data table and loads the table into your CAS session:

```
data mylib.getStarted;
   input C$ y x1-x10;
   datalines;
D  0  10.2  6  1.6  38  15  2.4  20  0.8  8.5  3.9
F  1  12.2  6  2.6  42  61  1.5  10  0.6  8.5  0.7
D  1   7.7  1  2.1  38  61    1  90  0.6  7.5  5.2
J  1  10.9  7  3.5  46  42  0.3   0  0.2    6  3.6
E  0  17.3  6  3.8  26  47  0.9  10  0.4  1.5  4.7
A  0  18.7  4  1.8   2  34  1.7  80    1  9.5  2.2
B  0   7.2  1  0.3  48  61  1.1  10  0.8  3.5    4
D  0   0.1  3  2.4   0  65  1.6  70  0.8  3.5  0.7
```

```
H  1   2.4  4  0.7  38  22  0.2  20    0    3  4.2
J  0  15.6  7  1.4   0  98  0.3   0    1    5  5.2
J  0  11.1  3  2.4  42  55  2.2  60  0.6  4.5  0.7
F  0     4  6  0.9   4  36  2.1  30  0.8    9  4.6
A  0   6.2  2  1.8  14  79  1.1  70  0.2    0  5.1
H  0   3.7  3  0.8  12  66  1.3  40  0.4  0.5  3.3
A  1   9.2  3  2.3  48  51  2.3  50    0    6  5.4
G  0    14  3    2  18  12  2.2   0    0    3  3.4
E  1  19.5  6  3.7  26  81  0.1  30  0.6    5  4.8
C  0    11  3  2.8  38   9  1.7  50  0.8  6.5  0.9
I  0  15.3  7  2.2  20  98  2.7 100  0.4    7  0.8
H  1   7.4  4  0.5  28  65  1.3  60  0.2  9.5  5.4
F  0  11.4  2  1.4  42  12  2.4  10  0.4    1  4.5
C  1  19.4  1  0.4  42   4  2.4  10    0  6.5  0.1
G  0   5.9  4  2.6  12  57  0.8  50  0.4    2  5.8
G  1  15.8  6  3.7  34   8  1.3  90  0.6  2.5  5.7
I  0    10  3  1.9  16  80    3  90  0.4  9.5  1.9
E  0  15.7  1  2.7  32  25  1.7  20  0.2  8.5    6
G  0    11  5  2.9  48  53  0.1  50    1  3.5  1.2
J  1  16.8  0  0.9  14  86  1.4  40  0.8    9    5
D  1    11  4  3.2  48  63  2.8  90  0.6    0  2.2
J  1   4.8  7  3.6  24   1  2.2  20    1  8.5  0.5
J  1  10.4  5    2  42  56    1  20    0  3.5  4.2
G  0  12.7  7  3.6   8  56  2.1  70    1  4.5  1.5
G  0   6.8  1  3.2  30  27  0.6   0  0.8    2  5.6
E  0   8.8  0  3.2   2  67  0.7  10  0.4    1    5
I  1   0.2  0  2.9  10  41  2.3  60  0.2    9  0.3
J  1   4.6  7  3.9  50  61  2.1  50  0.4    3  4.9
J  1   2.3  2  3.2  36  98  0.1  40  0.6  4.5  4.3
I  0  10.8  3  2.7  28  58  0.8  80  0.8    3    6
B  0   9.3  2  3.3  44  44  0.3  50  0.8  5.5  0.4
F  0   9.2  6  0.6   4  64  0.1   0  0.6  4.5  3.9
D  0   7.4  0  2.9  14   0  0.2  30  0.8  7.5  4.5
G  0  18.3  3  3.1   8  60  0.3  60  0.2    7  1.9
F  0   5.3  4  0.2  48  63  2.3  80  0.2    8  5.2
C  0   2.6  5  2.2  24   4  1.3  20    0    2  1.4
F  0  13.8  4  3.6   4   7  1.1  10  0.4  3.5  1.9
B  1  12.4  6  1.7  30  44  1.1  60  0.2    6  1.5
I  0   1.3  1  1.3   8  53  1.1  70  0.6    7  0.8
F  0  18.2  7  1.7  26  92  2.2  30    1  8.5  4.8
J  0   5.2  2  2.2  18  12  1.4  90  0.8    4  4.9
G  1   9.4  2  0.8  22  86  0.4  30  0.4    1  5.9
J  1  10.4  2  1.7  26  31  2.4  10  0.2    7  1.6
J  0    13  1  1.8  14  11  2.3  50  0.6  5.5  2.6
A  0  17.9  4  3.1  46  58  2.6  90  0.6  1.5  3.2
D  1  19.4  6    3  20  50  2.8 100  0.2    9  1.2
I  0  19.6  3  3.6  22  19  1.2   0  0.6    5  4.1
I  1     6  2  1.5  30  30  2.2  20  0.4  8.5  5.3
G  0  13.8  1  2.7   0  52  2.4  20  0.8    6    2
B  0  14.3  4  2.9  30  11  0.6  90  0.6  0.5  4.9
E  0  15.6  0  0.4  38  79  0.4  80  0.4    1  3.3
D  0    14  2    1  22  61    3  90  0.6    2  0.1
C  1   9.4  5  0.4  12  53  1.7  40    0    3  1.1
H  0  13.2  1  1.6  40  15  0.7  40  0.2    9  5.5
```

```
A  0  13.5  5  2.4  18  89  1.6  20  0.4  9.5  4.7
E  0   2.6  4  2.3  38   6  0.8  20  0.4    5  5.3
E  0  12.4  3  1.3  26   8  2.8  10  0.8    6  5.8
D  0   7.6  2  0.9  44  89  1.3  50  0.8    6  0.4
I  0  12.7  1  2.3  42   6  2.4  10  0.4    1    3
C  1  10.7  4  3.2  28  23  2.2  90  0.8  5.5  2.8
H  0  10.1  2  2.3  10  62  0.9  50  0.4  2.5  3.7
C  1  16.6  1  0.5  12  88  0.1  20  0.6  5.5  1.8
I  1   0.2  3  2.2   8  71  1.7  80  0.4  0.5  5.5
C  0  10.8  4  3.5  30  70  2.3  60  0.4  4.5  5.9
F  0   7.1  4    3  14  63  2.4  70    0    7  3.1
D  0  16.5  1  3.3  30  80  1.6  40    0  3.5  2.7
H  0  17.1  7  2.1  30  45  1.5  60  0.6  0.5  2.8
D  0   4.3  1  1.5  24  44    0  70    0    5  0.5
H  0    15  2  0.2  14  87  1.8  50    0  4.5  4.7
G  0  19.7  3  1.9  36  99  1.5  10  0.6    3  1.7
H  1   2.8  6  0.6  34  21    2  60    1    9  4.7
G  0  16.6  3  3.3  46   1  1.4  70  0.6  1.5  5.3
E  0  11.7  5  2.7  48   4  0.9  60  0.8  4.5  1.6
F  0  15.6  3  0.2   4  79  0.5   0  0.8  1.5  2.9
C  1   5.3  6  1.4   8  64    2  80  0.4    9  4.2
B  1   8.1  7  1.7  40  36  1.4  60  0.6    6  3.9
I  0  14.8  2  3.2   8  37  0.4  10    0  4.5    3
D  0   7.4  4    3  12   3  0.6  60  0.6    7  0.7
D  0   4.8  3  2.3  44  41  1.9  60  0.2    3  3.1
A  0   4.5  0  0.2   4  48  1.7  80  0.8    9  4.2
D  0   6.9  6  3.3  14  92  0.5  40  0.4  7.5    5
B  0   4.7  4  0.9  14  99  2.4  80    1  0.5  0.7
I  1   7.5  4  2.1  20  79  0.4  40  0.4  2.5  0.7
C  0   6.1  0  1.4  38  18  2.3  60  0.8  4.5  0.7
C  0  18.3  1    1  26  98  2.7  20    1  8.5  0.5
F  0  16.4  7  1.2  32  94  2.9  40  0.4  5.5  2.1
I  0   9.4  2  2.3  32  42  0.2  70  0.4  8.5  0.3
F  1  17.9  4  1.3  32  42    2  40  0.2    1  5.4
H  0  14.9  3  1.6  36  74  2.6  60  0.2    1  2.3
C  0  12.7  0  2.6   0  88  1.1  80  0.8  0.5  2.1
F  0   5.4  4  1.5   2   1  1.8  70  0.4  5.5  3.6
J  1  12.1  4  1.8  20  59  1.3  60  0.4    3  3.8
;
```

These statements assume that your CAS engine libref is named mylib, but you can substitute any appropriately defined CAS engine libref.

Unsupervised Variable Selection

The following statements use PROC VARREDUCE for unsupervised variable selection:

```
proc varreduce data=mylib.getStarted technique=VarianceAnalysis;
   class C;
   reduce unsupervised C x1-x10 / maxeffects=5 varexp=0.99;
run;
```

The TECHNIQUE= option in the PROC VARREDUCE statement specifies variance analysis as the technique to be used for variable selection. In the REDUCE statement, the MAXEFFECTS= option specifies 5 as the maximum number of variables to select, and the VAREXP= option specifies 99% as the maximum percentage of the total variance to explain. The procedure stops when either of these two conditions is satisfied.

The output from this analysis is presented in Figure 36.1 through Figure 36.3.

Figure 36.1 displays the "Number of Observations" tables. This table shows that all 100 observations in the data table are used in the analysis.

Figure 36.1 Number of Observations

The VARREDUCE Procedure

Number of Observations Read	100
Number of Observations Used	100

Figure 36.2 and Figure 36.3 show the "Selection Summary" and "Selected Effects" tables. The "Selection Summary" table shows which variable (or level for CLASS variables) is selected in each step, in addition to showing the total variance that is explained by the variables selected so far. The "Selected Effects" table presents all the selected variables and their corresponding variable types.

Figure 36.2 Selection Summary

Selection Summary

Iteration	Parameter	Proportion of Variance Explained	SSE	MSE	AIC	AICC	BIC
1	x3	0.068102	18.637953	0.188262	7.105200	27.216339	2.971252
2	x7	0.132579	17.348424	0.177025	6.993502	26.094008	2.945605
3	x10	0.193293	16.134131	0.166331	6.860937	24.945494	2.919092
4	C I	0.252883	14.942348	0.155649	6.704199	23.767490	2.888406
5	C F	0.311964	13.760728	0.144850	6.521819	22.558528	2.852077
6	C J	0.368951	12.620982	0.134266	6.315361	21.320171	2.811671
7	C B	0.425720	11.485609	0.123501	6.081095	20.048690	2.763457
8	C G	0.481257	10.374861	0.112770	5.819386	18.744449	2.707799
9	C D	0.535098	9.298037	0.102176	5.529803	17.407018	2.644269
10	C H	0.588910	8.221792	0.091353	5.206788	16.030839	2.567305
11	C A	0.640665	7.186698	0.080749	4.852232	14.617801	2.478801
12	x5	0.691609	6.167815	0.070089	4.459345	13.161117	2.371965

Figure 36.3 Selected Effects

Selected Effects

Number	Selected Variable	Variable Type
1	x3	INTERVAL
2	x7	INTERVAL
3	x10	INTERVAL
4	C	CLASS
5	x5	INTERVAL

Supervised Variable Selection

The following statements use PROC VARREDUCE for supervised variable selection. The CLASS statement specifies that y is a categorical response variable. The TECHNIQUE= option specifies discriminant analysis as the technique to be used for variable selection. The MAXEFFECTS= option in the REDUCE statement specifies 5 as the maximum number of variables to select.

```
proc varreduce data=mylib.getStarted technique=DiscriminantAnalysis;
   class C y;
   reduce supervised y = C x1-x10 / maxeffects=5;
run;
```

The output from this analysis is presented in Figure 36.4 through Figure 36.6.

Figure 36.4 shows that all 100 observations in the data table are used in the analysis.

Figure 36.4 Number of Observations

The VARREDUCE Procedure

Number of Observations Read	100
Number of Observations Used	100

Figure 36.5 and Figure 36.6 show the "Selection Summary" and "Selected Effects" tables. The "Selection Summary" table shows which variable (or level for CLASS variables) is selected in each step, in addition to showing the total variance that is explained by the variables selected so far. The "Selected Effects" table presents all the selected variables and their corresponding variable types.

Figure 36.5 Selection Summary

Selection Summary

Iteration	Parameter	Proportion of Variance Explained	SSE	MSE	AIC	AICC	BIC
1	C J	0.081099	0.918901	0.009282	0.015424	2.019590	-0.038525
2	x8	0.132303	0.867697	0.008854	-0.001913	2.005456	-0.049809
3	x2	0.168747	0.831253	0.008570	-0.004821	2.006669	-0.046665
4	C C	0.199152	0.800848	0.008342	-0.002084	2.014475	-0.037877
5	x4	0.218366	0.781634	0.008228	0.013631	2.036240	-0.016110
6	x9	0.236868	0.763132	0.008118	0.029676	2.059346	0.005986

Figure 36.6 Selected Effects

Selected Effects		
Number	Selected Variable	Variable Type
1	C	CLASS
2	x8	INTERVAL
3	x2	INTERVAL
4	x4	INTERVAL
5	x9	INTERVAL

Syntax: VARREDUCE Procedure

The following statements are available in the VARREDUCE procedure:

PROC VARREDUCE < *options* > **;**
CLASS *variable* < *(options)* >. . .< *variable* < *(options)* > > < */ global-options* > **;**
DISPLAY < *table-list* > < */options* > **;**
DISPLAYOUT *table-spec-list* < */options* > **;**
FREQ *variable* **;**
REDUCE UNSUPERVISED *effects* < */ reduce-options* > **;**
REDUCE SUPERVISED *response* . . .< *response* > **=** *effects* < */ reduce-options* > **;**
WEIGHT *variable* **;**

The PROC VARREDUCE statement and one REDUCE statement are required. The CLASS statement can appear multiple times. If a CLASS statement is specified, it must precede the REDUCE statement.

PROC VARREDUCE Statement

PROC VARREDUCE < *options* > **;**

The PROC VARREDUCE statement invokes the procedure. Table 36.1 summarizes the important options in the PROC VARREDUCE statement by function. The options are then described fully in alphabetical order.

Table 36.1 PROC VARREDUCE Statement Options

Option	Description
Basic Options	
DATA=	Specifies the input data table
Options Related to Variable Selection	
NOPRINT	Suppresses ODS output
MATRIX=	Specifies the matrix to use to select variables
TECHNIQUE=	Selects the variable selection technique
OUTCP=	Outputs the CORR, COV, or SSCP matrix, which is specified in the MATRIX= option

You can specify the following *options*:

DATA=*libref.data-table*
names the input data table for PROC VARREDUCE to use. The default is the most recently created data table. *libref.data-table* is a two-level name, where

libref refers to a collection of information that is defined in the LIBNAME statement and includes the library, which includes a path to the data, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *libref*, see the section "Using CAS Sessions and CAS Engine Librefs" on page 1889.

data-table specifies the name of the input data table.

MATRIX=CORR | COV | SSCP
specifies which matrix to use to select variables.

You can specify the following values:

CORR selects variables based on the Pearson correlation matrix. Assuming that X and Y are two variables, the correlation between X and Y is computed by

$$\mathrm{Corr}(X,Y) = \frac{E\left((X - E\left(X\right))\left(Y - E\left(Y\right)\right)\right)}{\sqrt{E\left(X - E\left(X\right)\right)^2 E\left(Y - E\left(Y\right)\right)^2}}$$

COV selects variables based on the covariance matrix. Assuming that X and Y are two variables, the covariance between X and Y is computed by

$$\mathrm{Cov}(X,Y) = E\left(\left(X - E\left(X\right)\right)\left(Y - E\left(Y\right)\right)\right)$$

SSCP selects variables based on the sums of squares and crossproducts matrix. Assuming that X and Y are two variables and that $\mathbf{x}$ and $\mathbf{y}$ are their corresponding variable vectors, the SSCP between X and Y is computed by

$$\mathrm{SSCP}(X,Y) = \mathbf{x}^{\top}\mathbf{y}$$

By default, MATRIX=CORR.

NOPRINT
suppresses the generation of ODS output.

OUTCP=*libref.data-table* **< / LIST < EPS =** *number* **> >**
creates a data table that contains a symmetric matrix that depicts the relationships among variables and also creates a set of statistics about the input data table and variables. *libref.data-table* is a two-level name, where *libref* refers to the library, and *data-table* specifies the name of the output data table. For more information about this two-level name, see the DATA= option and the section "Using CAS Sessions and CAS Engine Librefs" on page 1889. Depending on the value of the MATRIX= option, the symmetric matrix can be a correlation (CORR) matrix, a covariance (COV) matrix, or a sums of squares and crossproducts (SSCP) matrix.

When you specify the LIST option, the symmetric matrix is output in the list-of-list (LIL) format. In this format, the matrix is represented as a set of tuples $(i,\ j,\ x)$, where x is an entry in the matrix and i and j denote its row and column indices, respectively. LIL format can be used when the output contains too many columns to fit in a data table. For example, in most database systems, the maximum number of columns in a table is usually limited to several thousand. If an output matrix contains more columns than the limit, you must use the LIST option to avoid errors that would arise from writing too many columns to the table. When LIL format is used, all 0 entries in the matrix are ignored in the output.

When you specify EPS= *number* in the LIST suboption, matrix entries whose absolute value is smaller than *number* are ignored in the output. This feature helps omit unreliable estimations and generate a compact representation of the matrix. When you do not specify the EPS= option, only the 0 entries in the matrix are ignored in the output.

TECHNIQUE=*keyword*

TECH=*keyword*

specifies the variable selection technique.

You can specify the following *keyword*s:

VARIANCEANALYSIS | VAR performs variance analysis for variable selection.

DISCRIMINANTANALYSIS | DSC performs discriminant analysis for variable selection.

By default, TECHNIQUE=VAR.

You can use variance analysis for both supervised and unsupervised variable selection. You can use discriminant analysis only for supervised variable selection that uses one classification variable as the response. For more information, see the section "Variable Selection for Classification" on page 1904.

CLASS Statement

CLASS *variable* < *(options)* >. . . < *variable* < *(options)* > > < / *global-options* > **;**

The CLASS statement names the classification variables to be used as explanatory variables in the analysis. The CLASS statement must precede the REDUCE statement.

The VARREDUCE procedure supports the GLM method of CLASS variable parameterization; for more information, see the section "GLM Parameterization of Classification Variables and Effects" on page 1907.

You can specify options either as individual variable *options*, by enclosing the options in parentheses after the variable name, or as *global-options*, by placing them after a slash (/). *Global-options* are applied to all variables that are specified in the CLASS statement. If you specify more than one CLASS statement, the *global-options* that are specified in any one CLASS statement apply to all CLASS statements. However, individual CLASS variable *options* override the *global-options*.

Table 36.2 summarizes the values you can use for either an *option* or a *global-option*. The options are described in detail in the list that follows Table 36.2.

Table 36.2 CLASS Statement Options

Option	Description
DESCENDING	Reverses the sort order
MISSING	Treats missing values as valid levels

DESCENDING

DESC

reverses the sort order of the classification variable.

MISSING

treats missing values (“.”, “.A”, . . . , “.Z” for numeric variables and blanks for character variables) as valid values for the CLASS variable.

If you do not specify the MISSING option, observations that have missing values for CLASS variables are removed from the analysis.

DISPLAY Statement

DISPLAY < *table-list* > < / *options* > **;**

The DISPLAY statement enables you to specify a list of display tables to display or exclude. This statement is similar to the ODS SELECT, ODS EXCLUDE, and ODS TRACE statements. However, the DISPLAY statement can improve performance when a large number of tables could be generated (such as in BY-group processing). The procedure processes the DISPLAY statement on a CAS server and thus sends only a subset of ODS tables to the SAS client. Because ODS statements are processed on a SAS client, first all the generated display tables are sent to the client, and then the client creates a subset.

If you use both DISPLAY and ODS statements together, the DISPLAY statement takes precedence over the ODS statements. Note that the ODS EXCLUDE statement processes tables that are sent to the client after they have been filtered by the DISPLAY statement. In some cases, it might appear that the ODS EXCLUDE statement is taking precedence because it can further filter the tables. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

You can specify the *table-list* as a list of table names, paths, partial pathnames, and regular expressions.

The table names that you can specify are listed in the section “ODS Table Names” on page 1912. A path is a table name that is prefixed with dot-separated grouping information. For example, a SelectionSummary table that a procedure produces during a selection routine might have the path *Bygroup1.Summary.SelectionSummary*. A partial pathname does not include all groups; for example, *SelectionSummary* and *Summary.SelectionSummary* are partial pathnames for *Bygroup1.Summary.SelectionSummary*.

When you specify a table name or partial pathname, all display tables whose paths end in the specified name are selected for display or exclusion. For example, both *SelectionSummary* and *Summary.SelectionSummary* select *Bygroup1.Summary.SelectionSummary*.

A regular expression is enclosed in forward slashes (/). For example, specifying “/tions/” selects all pathnames that contain the substring “tions”; in particular, the *Bygroup1.Summary.SelectionSummary* table is selected.

Specifying "!/tions/" selects all pathnames that do not contain the substring "tions"; in particular, the *Bygroup1.Summary.SelectionSummary* table is not selected.

You can specify the following *options* after a slash (/):

CASESENSITIVE
performs a case-sensitive comparison of table names in the *table-list* to display table names when tables are subsetted for display. To preserve case, you must enclose table names in the *table-list* in quotation marks.

EXCLUDE
displays all display tables except those that you specify in the *table-list*.

EXCLUDEALL
suppresses display of all tables. This option takes precedence over the other options.

TRACE
displays the display table names, labels, and paths.

DISPLAYOUT Statement

DISPLAYOUT *table-spec-list* < / *options* > **;**

The DISPLAYOUT statement enables you to create output tables from your displayed output. This statement is similar to the ODS OUTPUT statement. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

The *table-spec-list* specifies a list of output tables to create. Each entry in the list has either a *key*=*value* format or a *key* format:

key=*value*	specifies *key* as the ODS table name, path, or partial pathname, and specifies *value* as the output table name.
key	specifies *key* as the ODS table name and also as the output table name.

The ODS table names that you can specify are listed in the section "ODS Table Names" on page 1912. You cannot specify the ODS table named OutputCasTables in the *table-spec-list*.

Table names and partial pathnames are discussed under the DISPLAY statement. The DISPLAYOUT statement does not support regular expressions.

You can specify the following *options* after a slash (/):

INCLUDEALL
creates output tables for all display tables. The name of the created output table is the same as the name of the corresponding display table. If you specify this option, the *table-spec-list* specification is ignored.

NOREPLACE
does not replace any existing output table of the same name.

REPEATED
replicates all output tables on all nodes.

FREQ Statement

FREQ *variable* **;**

The *variable* in the FREQ statement identifies a numeric variable in the input data table that contains the frequency of occurrence of each observation. PROC VARREDUCE treats each observation as if it appears f times, where f is the value of the FREQ *variable* for the observation. If f is not an integer, it is truncated to an integer. If f is less than 1 or missing, the observation is not used in the analysis. When you omit the FREQ statement, each observation is assigned a frequency of 1.

REDUCE Statement

REDUCE UNSUPERVISED *effects* < / *reduce-options* > **;**

REDUCE SUPERVISED *response-variable* < *response-variable* > . . . = *effects* < / *reduce-options* > **;**

PROC VARREDUCE can be used for either of the following types of variable selection:

- For unsupervised variable selection, you specify the *effects* to be considered in the variable selection process. An *effect* can be an original variable in the input data table or a variable that is constructed from the original variables.
- For supervised variable selection, you specify both the *effects* and one or more *response-variables*. A *response-variable* can be an original variable in the input data table or a variable that is constructed from the original variables. PROC VARREDUCE supports variable selection both in a classification (categorization) context, in which you specify one *response-variable*, and in a regression context, in which you can specify more than one *response-variable*.

Table 36.3 summarizes the *reduce-options*, which control the number of variables to be selected.

Table 36.3 *reduce-options*

Option	Description
AIC	Uses Akaike's information criterion to perform model selection
AICC	Uses the corrected Akaike's information criterion to perform model selection
BIC	Uses Schwarz Bayesian information criterion to perform model selection

Table 36.3 *continued*

Option	Description
MAXITER= | MAXSTEPS=	Specifies the maximum number of steps to take; the number must be greater than or equal to 1
MAXEFFECTS=	Specifies the number of effects to select; the number must be greater than or equal to 1.
VARIANCEEXPLAINED= | VAREXP=	Specifies the fraction of the total variance to be explained; the value must be between 0 and 1.
MINVARIANCEINCREMENT= | VARINC=	Specifies the minimum increment of explained variance (in a fraction of the total variance); the value must be between 0 and 1.

The *reduce-options* determine the number of variables to be selected. You can specify the following *reduce-options* as stopping criteria for the VARREDUCE procedure. When you specify more than one option, PROC VARREDUCE stops whenever one of the specified options is satisfied, or when the explained variance equals the total variance. In the latter case, the procedure prints the following message in the log: "Early stop: the proportion of the explained variance to the total variance equals 1."

You can specify the following *reduce-options* only for supervised variable selection:

AIC
stops PROC VARREDUCE if the Akaike's information criterion (AIC) value fails to decrease in three contiguous steps.

AICC
stops PROC VARREDUCE if the corrected Akaike's information (AICC) value fails to decrease in three contiguous steps.

BIC
stops PROC VARREDUCE if the Schwarz Bayesian information criterion (BIC) value fails to decrease in three contiguous steps.

You can specify the following *reduce-options* for both supervised and unsupervised variable selection:

MAXITER=*n*

MAXSTEPS=*n*
stops PROC VARREDUCE after it runs *n* steps.

By default, *n* is the number of input variables.

MAXEFFECTS=*n*
stops PROC VARREDUCE after *n* effects have been selected. Because individual levels of one classification variable can be selected in different steps of the variable selection process, selecting *n* effects might require more steps than are specified by the value of the MAXITER= option.

By default, MAXEFFECTS=200.

VARIANCEEXPLAINED=*fraction*

VAREXP=*fraction*

stops PROC VARREDUCE when the *fraction* of the total variance can be explained by the selected variables.

By default, VARIANCEEXPLAINED=0.9.

MINVARIANCEINCREMENT=*fraction*

VARINC=*fraction*

stops PROC VARREDUCE when the minimum increment of the explained variance is less than *fraction* of the total variance.

By default, MINVARIANCEINCREMENT=0.

WEIGHT Statement

WEIGHT *variable* **;**

The *variable* in the WEIGHT statement is used as a weight to perform a weighted analysis of the data. Observations that have nonpositive or missing weights are not included in the analysis. The values of the WEIGHT variable can be nonintegral and are not truncated. When you omit the WEIGHT statement, all observations that are used in the analysis are assigned a weight of 1.

Details: VARREDUCE Procedure

The performance of a learning model usually decreases in terms of accuracy and efficiency when the dimensionality of the input data is high. The problem is known as the "curse of dimensionality." Variable selection techniques can reduce the dimensionality of data by removing irrelevant and redundant variables (Liu and Motoda 1998).

The VARREDUCE procedure performs both supervised and unsupervised variable selection. It selects variables by identifying a set of variables that can jointly explain the maximum amount of data variance.

Missing Values

Any observation that has missing values for the responses, frequency, or effects is excluded from the analysis; however, missing values are valid for responses and effects if the MISSING option is specified in the CLASS statement. Observations that have a frequency less than 1 are also excluded.

Unsupervised Variable Selection

When no response variable is specified, PROC VARREDUCE conducts unsupervised variable selection. Assume that k variables need to be selected. Let $\mathbf{X} \in \mathbb{R}^{n \times m}$ be a data table that contains n samples and m variables; let $\mathbf{X} = (\mathbf{X}_1, \mathbf{X}_2)$, where $\mathbf{X}_1 \in \mathbb{R}^{n \times k}$ contains the k selected variables and $\mathbf{X}_2 \in \mathbb{R}^{n \times (m-k)}$ contains the remaining $m - k$ variables. PROC VARREDUCE selects the variables by minimizing the following equation:

$$\min \text{Trace}\left(\mathbf{X}_2^\top \left(\mathbf{I} - \mathbf{X}_1 \left(\mathbf{X}_1^\top \mathbf{X}_1\right)^{-1} \mathbf{X}_1^\top\right) \mathbf{X}_2\right)$$

$\left(\mathbf{I} - \mathbf{X}_1 \left(\mathbf{X}_1^\top \mathbf{X}_1\right)^{-1} \mathbf{X}_1^\top\right)^{\frac{1}{2}} \mathbf{X}_2$ projects $\mathbf{X}_2$ to the null space of $\mathbf{X}_1$. Therefore, the preceding equation measures the data variance that resides in the null space of $\mathbf{X}_1$, which is the data variance that cannot be explained by the variables in $\mathbf{X}_1$. Minimizing this equation leads to the selection of the variables that jointly explain the maximum amount of the variance in the original data.

Let $\mathbf{C}_{11} = \mathbf{X}_1^\top \mathbf{X}_1$, $\mathbf{C}_{12} = \mathbf{X}_1^\top \mathbf{X}_2$, and $\mathbf{C}_{21} = \mathbf{X}_2^\top \mathbf{X}_1$. The following equations hold:

$$\mathbf{C} = \mathbf{X}^\top \mathbf{X} = \begin{pmatrix} \mathbf{C}_{11} & \mathbf{C}_{12} \\ \mathbf{C}_{21} & \mathbf{C}_{22} \end{pmatrix}$$

$$\mathbf{X}_2^\top \left(\mathbf{I} - \mathbf{X}_1 \left(\mathbf{X}_1^\top \mathbf{X}_1\right)^{-1} \mathbf{X}_1^\top\right) \mathbf{X}_2 = \mathbf{C}_{22} - \mathbf{C}_{21}\mathbf{C}_{11}^{-1}\mathbf{C}_{12}$$

When all the variables are centralized to have a zero mean, $\mathbf{C}$ is the covariance matrix. This corresponds to specifying MATRIX=COV in the PROC VARREDUCE statement, which requests that the covariance matrix be used for variable selection. Similarly, if variables need to be both centralized and normalized to have unit length, you should specify MATRIX=CORR in the PROC VARREDUCE statement, which requests that the correlation matrix be used for variable selection. If neither centralization nor normalization should be applied, specify MATRIX=SSCP in the PROC VARREDUCE statement.

Principal component analysis (PCA) (Jolliffe 2002) also reduces dimensionality by preserving data variance. The key difference between PCA and PROC VARREDUCE is that PCA generates a small set of new variables (variable extraction) by linearly combining the original variables, whereas PROC VARREDUCE selects a small set of the original variables (variable selection). The variables returned by PROC VARREDUCE are the original variables. This feature is very important in applications where retaining the original variables is important for model exploration or interpretation (for example, genetic analysis and text mining).

Supervised Variable Selection

When response variables are specified in a REDUCE statement, PROC VARREDUCE conducts supervised variable selection, either in a regression context or in a classification (categorization) context.

Variable Selection for Regression

In a regression context, all response variables should be numerical. When the response is a classification variable, it needs to be levelized to multiple dummy variables, where each dummy variable corresponds to a level of the classification variable. You can achieve this levelization by adding this variable to the variable list in the CLASS statement.

Let $\mathbf{Y} \in \mathbb{R}^{n \times t}$ be the response data that contain t response variables. Assume that k variables need to be selected. Let $\mathbf{X} \in \mathbb{R}^{n \times m}$ be a data table that contains n samples and m variables; let $\mathbf{X} = (\mathbf{X}_1, \mathbf{X}_2)$, where $\mathbf{X}_1 \in \mathbb{R}^{n \times k}$ contains the k selected variables and $\mathbf{X}_2 \in \mathbb{R}^{n \times (m-k)}$ contains the remaining $m - k$ variables. PROC VARREDUCE selects the variables by minimizing the following equation:

$$\min \text{Trace}\left(\mathbf{Y}^\top \left(\mathbf{I} - \mathbf{X}_1 \left(\mathbf{X}_1^\top \mathbf{X}_1\right)^{-1} \mathbf{X}_1^\top\right) \mathbf{Y}\right)$$

$\left(\mathbf{I} - \mathbf{X}_1 \left(\mathbf{X}_1^\top \mathbf{X}_1\right)^{-1} \mathbf{X}_1^\top\right)^{\frac{1}{2}} \mathbf{Y}$ projects $\mathbf{Y}$ onto the null space of $\mathbf{X}_1$. Therefore, the equation measures the response variance that resides in the null space of $\mathbf{X}_1$, which is the variance of the response variables that cannot be explained by the variables in $\mathbf{X}_1$. Minimizing the equation leads to the selection of the variables that jointly explain the maximum amount of the variance of the response variables.

Variable Selection for Classification

In a classification context, one classification variable is specified as the response, and each of its levels corresponds to a category of the classification problem. Let the classification variable be $\mathbf{y}$ with c levels $\{1, \ldots, c\}$. Then $\mathbf{y}$ can be levelized in a special way to generate response data $\mathbf{Y} \in \mathbb{R}^{n \times c}$ as:

$$\mathbf{Y}_{i,j} = \begin{cases} \frac{1}{\sqrt{n}} \left(\sqrt{\frac{n}{n_j}} - \sqrt{\frac{n_j}{n}}\right), & \mathbf{y}_i = j \\ -\frac{1}{\sqrt{n}} \sqrt{\frac{n_j}{n}}, & \mathbf{y}_i \neq j \end{cases}$$

By using this $\mathbf{Y}$ in the variance analysis, PROC VARREDUCE selects variables by using the discriminant criterion that is specified in linear discriminant analysis (LDA) (Fisher 1936; Cooley and Lohnes 1971). LDA also reduces dimensionality. The key difference between LDA and PROC VARREDUCE is that LDA generates a small set of new variables (variable extraction) by linearly combining the original variables, whereas PROC VARREDUCE selects a small set of the original variables (variable selection).

Criteria Used in Model Selection

The VARREDUCE procedure supports the following three fit statistics that you can specify as stopping criteria in the REDUCE statement. These statistics are produced only for the supervised variable selection.

AIC	Akaike's information criterion (Akaike 1969; Judge et al. 1985)
AICC	Corrected Akaike's information criterion (Hurvich and Tsai 1989)
BIC	Schwarz Bayesian information criterion (Schwarz 1978; Judge et al. 1985)

The VARREDUCE procedure supports multiple response variables; therefore, it computes the AIC, AICC, and BIC that are defined for multivariate regression. Besides the three criteria, it also computes the error sum of squares (SSE) and residual mean square error (MSE).

Table 36.4 provides formulas and definitions for these fit statistics.

Table 36.4 Formulas and Definitions for Model Fit Summary Statistics

Statistic	Definition or Formula
n	Number of observations
p	Number of parameters
t	Number of response variables in supervised selection; number of unselected variables in unsupervised selection
SSE	Error sum of squares
MSE	$\frac{\text{SSE}}{n-p}$
AIC	$\ln(\text{SSE}) + \frac{2pt + t(t+1)}{n}$
AICC	$\ln(\text{SSE}) + \frac{(n+p)t}{n-p-t-1}$
BIC	$\ln(\text{SSE}) + \frac{p\ln(n)}{n}$

Computational Method

Given m variables, finding the k variables that minimize the proposed equations is a combinatorial problem, which is NP-hard (nondeterministic polynomial-time hard). To select k variables, PROC VARREDUCE applies k steps of a greedy search to generate a suboptimal solution for the problem.

Assume that q features have been selected, that $\mathbf{X}_1$ contains the q selected variables, and that $\mathbf{X}_2$ contains the remaining variables. PROC VARREDUCE selects the $q+1$ variable, F, by minimizing the equation

$$\arg\min_{F} \text{Trace}\left(\hat{\mathbf{X}}_2^{\top}\left(\mathbf{I} - \hat{\mathbf{X}}_1\left(\hat{\mathbf{X}}_1^{\top}\hat{\mathbf{X}}_1\right)^{-1}\hat{\mathbf{X}}_1^{\top}\right)\hat{\mathbf{X}}_2\right)$$

where $\hat{\mathbf{X}}_1$ is the data table that contains the feature F and the q selected variables, and $\hat{\mathbf{X}}_2$ is the data table that contains the remaining variables. Minimizing the preceding problem is equivalent to maximizing the following problem:

$$\frac{\left\| \mathbf{X}_2^\top \left(\mathbf{I} - \mathbf{X}_1 \left(\mathbf{X}_1^\top \mathbf{X}_1 \right)^{-1} \mathbf{X}_1^\top \right) \mathbf{f} \right\|_2^2}{\left\| \left(\mathbf{I} - \mathbf{X}_1 \left(\mathbf{X}_1^\top \mathbf{X}_1 \right)^{-1} \mathbf{X}_1^\top \right)^{\frac{1}{2}} \mathbf{f} \right\|_2^2}$$

In the preceding equation, $\left\| \mathbf{X}_2^\top \left(\mathbf{I} - \mathbf{X}_1 \left(\mathbf{X}_1^\top \mathbf{X}_1 \right)^{-1} \mathbf{X}_1^\top \right) \mathbf{f} \right\|_2^2$ is the summation of the squares of the covariance between the variable $\mathbf{f}$ and all the unselected variables in the null space of $\mathbf{X}_1$. And $\left\| \left(\mathbf{I} - \mathbf{X}_1 \left(\mathbf{X}_1^\top \mathbf{X}_1 \right)^{-1} \mathbf{X}_1^\top \right)^{\frac{1}{2}} \mathbf{f} \right\|_2^2$ is the square of the variance of $\mathbf{f}$ in the null space of $\mathbf{X}_1$, which is used as a normalization factor.

This problem can be solved efficiently. Assuming that $m \gg k$, the time complexity for solving it is

$$O\left(m^2 \left(n + k^2\right)\right)$$

where m is the number of variables, n is the number of samples, and k is the number of selected variables. In the equation, $m^2 n$ corresponds to the time for computing the covariance (or correlation or SSCP) matrix. And $m^2 k^2$ corresponds to the time for selecting k variables out of m.

Similar analysis also applies to supervised variable selection with PROC VARREDUCE. The following problem is maximized for supervised variable selection:

$$\frac{\left\| \mathbf{Y}^\top \left(\mathbf{I} - \mathbf{X}_1 \left(\mathbf{X}_1^\top \mathbf{X}_1 \right)^{-1} \mathbf{X}_1^\top \right) \mathbf{f} \right\|_2^2}{\left\| \left(\mathbf{I} - \mathbf{X}_1 \left(\mathbf{X}_1^\top \mathbf{X}_1 \right)^{-1} \mathbf{X}_1^\top \right)^{\frac{1}{2}} \mathbf{f} \right\|_2^2}$$

Here, $\mathbf{Y}$ is the response data table. Let c be the number of columns in $\mathbf{Y}$. The time complexity for selecting k variables by solving the preceding problem is

$$O\left(k^2 \left(c + k\right) m + m^2 n\right)$$

Note that for most data of very high dimensionality, $c + k \ll m$.

PROC VARREDUCE is fully threaded and distributed. When there are p machines used for computing, the time complexity for unsupervised variable selection is

$$\mathrm{CPU}\left(\frac{m^2 \left(n + k^2\right)}{p} + m^2 \log p\right) + \mathrm{NET}\left(m^2 \log p\right)$$

and the time complexity for supervised variable selection is

$$\mathrm{CPU}\left(\frac{k^2 \left(c + k\right) m + m^2 n}{p} + m^2 \log p\right) + \mathrm{NET}\left(m^2 \log p\right)$$

where CPU corresponds to the time used for computing and NET corresponds to the time used for communication among computers.

GLM Parameterization of Classification Variables and Effects

Table 36.5 shows the types of effects that are available in the VARREDUCE procedure; they are discussed in more detail in the following subsections. Let A, B, and C represent classification variables, and let X and Z represent continuous variables.

Table 36.5 Available Types of Effects

Effect	Example	Description
Intercept	Default	Intercept (unless the NOINT option is specified)
Regression	X Z	Continuous variables
Polynomial	X*Z	Interaction of continuous variables
Main	A B	CLASS variables
Interaction	A*B	Crossing of CLASS variables
Nested	A(B)	Main effect A nested within CLASS effect B
Continuous-by-class	X*A	Crossing of continuous and CLASS variables
Continuous-nesting-class	X(A)	Continuous variable X1 nested within CLASS variable A
General	X*Z*A(B)	Combinations of different types of effects

Table 36.6 shows some examples of MODEL statements that use various types of effects.

Table 36.6 Model Statement Effect Examples

Specification	Type of Model
`model Y=X;`	Simple regression
`model Y=X Z;`	Multiple regression
`model Y=X X*X;`	Polynomial regression
`model Y=A;`	One-way analysis of variance (ANOVA)
`model Y=A B C;`	Main-effects ANOVA
`model Y=A B A*B;`	Factorial ANOVA with interaction
`model y=A B(A) C(B A);`	Nested ANOVA
`model Y=A X;`	Analysis of covariance (ANCOVA)
`model Y=A X(A);`	Separate-slopes regression
`model Y=A X X*A;`	Homogeneity-of-slopes regression

Intercept

By default, linear models that are created by the VARREDUCE procedure automatically include a column of 1s in **X**. This column corresponds to an intercept parameter. In many procedures, you can use the NOINT option in the MODEL statement to suppress this intercept. For example, the NOINT option is useful when the MODEL statement contains a classification effect and you want the parameter estimates to be in terms of the mean response for each level of that effect.

Regression Effects

Numeric variables or polynomial terms that involve them can be included in the model as regression effects (covariates). The actual values of such terms are included as columns of the relevant model matrices. You can use the bar operator along with a regression effect to generate polynomial effects. For example, X | X | X expands to X X*X X*X*X, which is a cubic model.

Main Effects

If a classification variable has *m* levels, the GLM parameterization generates *m* columns for its main effect in the model matrix. Each column is an indicator variable for a particular level. The order of the columns is the sort order of the values of their levels and can be controlled by the ORDER= option in the CLASS statement.

Table 36.7 is an example where β_0 denotes the intercept and A and B are classification variables that have two and three levels, respectively.

Table 36.7 Example of Main Effects

Data		I	A		B		
A	B	β_0	A1	A2	B1	B2	B3
1	1	1	1	0	1	0	0
1	2	1	1	0	0	1	0
1	3	1	1	0	0	0	1
2	1	1	0	1	1	0	0
2	2	1	0	1	0	1	0
2	3	1	0	1	0	0	1

There are usually more columns for these effects than there are degrees of freedom to estimate them. In other words, the GLM parameterization of main effects is *singular*.

Interaction Effects

Often a model includes interaction (crossed) effects to account for how the effect of a variable changes along with the values of other variables. With an interaction, the terms are first reordered to correspond to the order of the variables in the CLASS statement. Thus, B*A becomes A*B if A precedes B in the CLASS statement. Then, the GLM parameterization generates columns for all combinations of levels that occur in the data. The order of the columns is such that the rightmost variables in the interaction change faster than the leftmost variables (Table 36.8).

Table 36.8 Example of Interaction Effects

Data		I	A		B			A*B					
A	B	β_0	A1	A2	B1	B2	B3	A1B1	A1B2	A1B3	A2B1	A2B2	A2B3
1	1	1	1	0	1	0	0	1	0	0	0	0	0
1	2	1	1	0	0	1	0	0	1	0	0	0	0
1	3	1	1	0	0	0	1	0	0	1	0	0	0
2	1	1	0	1	1	0	0	0	0	0	1	0	0
2	2	1	0	1	0	1	0	0	0	0	0	1	0
2	3	1	0	1	0	0	1	0	0	0	0	0	1

In the preceding matrix, main-effects columns are not linearly independent of crossed-effects columns. In fact, the column space for the crossed effects contains the space of the main effect.

When your model contains many interaction effects, you might be able to code them more parsimoniously by using the bar operator (|). The bar operator generates all possible interaction effects. For example, A | B | C expands to A B A*B C A*C B*C A*B*C. To eliminate higher-order interaction effects, use the at sign (@) in conjunction with the bar operator. For example, A | B | C | D@2 expands to A B A*B C A*C B*C D A*D B*D C*D.

Nested Effects

Nested effects are generated in the same manner as crossed effects. Hence, the design columns that are generated by the following two statements are the same (but the ordering of the columns is different):

```
model Y=A B(A);

model Y=A A*B;
```

The nesting operator in the VARREDUCE procedure is more of a notational convenience than an operation that is distinct from crossing. Nested effects are typically characterized by the property that the nested variables do not appear as main effects. The order of the variables within nesting parentheses is made to correspond to the order of these variables in the CLASS statement. The order of the columns is such that variables outside the parentheses index faster than those inside the parentheses, and the rightmost nested variables index faster than the leftmost variables (Table 36.9).

Table 36.9 Example of Nested Effects

Data		I	A		B(A)					
A	B	β_0	A1	A2	B1A1	B2A1	B3A1	B1A2	B2A2	B3A2
1	1	1	1	0	1	0	0	0	0	0
1	2	1	1	0	0	1	0	0	0	0
1	3	1	1	0	0	0	1	0	0	0
2	1	1	0	1	0	0	0	1	0	0
2	2	1	0	1	0	0	0	0	1	0
2	3	1	0	1	0	0	0	0	0	1

Continuous-Nesting-Class Effects

When a continuous variable nests or crosses with a classification variable, the design columns are constructed by multiplying the continuous values into the design columns for the classification effect (Table 36.10).

Table 36.10 Example of Continuous-Nesting-Class Effects

Data		I	A		X(A)	
X	A	β_0	A1	A2	X(A1)	X(A2)
21	1	1	1	0	21	0
24	1	1	1	0	24	0

Table 36.10 *continued*

Data		I	A		X(A)	
22	1	1	1	0	22	0
28	2	1	0	1	0	28
19	2	1	0	1	0	19
23	2	1	0	1	0	23

This model estimates a separate intercept and a separate slope for X within each level of A.

Continuous-by-Class Effects

Continuous-by-class effects generate the same design columns as continuous-nesting-class effects. Table 36.11 shows the construction of the X*A effect. The two columns for this effect are the same as the columns for the X(A) effect in Table 36.10.

Table 36.11 Example of Continuous-by-Class Effects

Data		I	X	A		X*A	
X	A	β_0	X	A1	A2	X*A1	X*A2
21	1	1	21	1	0	21	0
24	1	1	24	1	0	24	0
22	1	1	22	1	0	22	0
28	2	1	28	0	1	0	28
19	2	1	19	0	1	0	19
23	2	1	23	0	1	0	23

You can use continuous-by-class effects together with pure continuous effects to test for homogeneity of slopes.

General Effects

An example that combines all the effects is X1*X2*A*B*C(D E). The continuous list comes first, followed by the crossed list, followed by the nested list in parentheses.

Effects might be renamed by the procedure to correspond to ordering rules. For example, B*A(E D) might be renamed A*B(D E) to satisfy the following:

- Classification variables that occur outside parentheses (crossed effects) are sorted in the order in which they appear in the CLASS statement.
- Variables within parentheses (nested effects) are sorted in the order in which they appear in the CLASS statement.

The sequencing of the parameters that are generated by an effect is determined by the variables whose levels are indexed faster:

- Variables in the crossed list index faster than variables in the nested list.
- Within a crossed or nested list, variables to the right index faster than variables to the left.

For example, suppose that a model includes four effects—A, B, C, and D—each having two levels, 1 and 2, and that the CLASS statement is as follows:

```
class A B C D;
```

Then the order of the parameters for the effect B*A(C D), which is renamed A*B(C D), is as follows:

$$\begin{array}{llll} A_1B_1C_1D_1 \rightarrow & A_1B_2C_1D_1 \rightarrow & A_2B_1C_1D_1 \rightarrow & A_2B_2C_1D_1 \rightarrow \\ A_1B_1C_1D_2 \rightarrow & A_1B_2C_1D_2 \rightarrow & A_2B_1C_1D_2 \rightarrow & A_2B_2C_1D_2 \rightarrow \\ A_1B_1C_2D_1 \rightarrow & A_1B_2C_2D_1 \rightarrow & A_2B_1C_2D_1 \rightarrow & A_2B_2C_2D_1 \rightarrow \\ A_1B_1C_2D_2 \rightarrow & A_1B_2C_2D_2 \rightarrow & A_2B_1C_2D_2 \rightarrow & A_2B_2C_2D_2 \end{array}$$

Note that first the crossed effects B and A are sorted in the order in which they appear in the CLASS statement so that A precedes B in the parameter list. Then, for each combination of the nested effects in turn, combinations of A and B appear. The B effect changes fastest because it is rightmost in the cross list. Then A changes next fastest, and D changes next fastest. The C effect changes most slowly because it is leftmost in the nested list.

Displayed Output

The following sections describe the output that PROC VARREDUCE produces by default. The output is organized into various tables, which are discussed in the order of appearance.

Number of Observations

The "Number of Observations" table displays the number of observations read from the input data table and the number of observations used in the analysis.

Selection Summary

The "Selection Summary" table displays for each iteration the name of the selected effect, the name of the selected level, and the total variance explained after the iteration.

Selected Variables

The "Selected Variables" table summarizes which variables were selected in the selection process. It also provides information about the variable type of each selected variable.

ODS Table Names

Each table that the VARREDUCE procedure creates has a name associated with it. You must use this name to refer to the table when you use ODS statements. The name of each table and a short description of the contents are listed in Table 36.12.

Table 36.12 ODS Tables Produced by PROC VARREDUCE

Table Name	Description	Statement	Option
NObs	Number of observations read and used; number of events and trials, if applicable	PROC VARREDUCE	Default
SelectedEffects	Summary of selected variables	PROC VARREDUCE	Default
SelectionSummary	Selection summary	PROC VARREDUCE	Default

Examples: VARREDUCE Procedure

The examples in this chapter are available in the GitHub repository located at `https://github.com/sascommunities/doc-supplement-casstat`.

NOTE: Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections "Using CAS Sessions and CAS Engine Librefs" on page 7 and "Loading a SAS Data Set onto a CAS Server" on page 7 in Chapter 2, "Shared Concepts."

Example 36.1: Analyzing Framingham Heart Study Data

(View the complete code for this example (varreduce_ex1.sas) in the example repository.)

This example analyzes the data from the Framingham Heart Study for feature selection, which are available in a data set named Heart in the Sashelp library.

You can load the Sashelp.Heart data set into your CAS session by specifying your CAS engine libref in the first statement in the following DATA step:

```
data mylib.heart;
   set sashelp.heart;
run;
```

These statements assume that your CAS engine libref is named mylib, but you can substitute any appropriately defined CAS engine libref.

The following statements invoke the VARREDUCE procedure:

```
proc varreduce data=mylib.heart matrix=COV tech=DSC;
   ods output SelectionSummary=Summary;
   class Status Sex Chol_Status BP_Status Weight_Status Smoking_Status;
   reduce supervised Status = Sex AgeAtStart Height Weight Diastolic Systolic MRW
                              Smoking Cholesterol Chol_Status BP_Status Weight_Status
                              Smoking_Status/ maxiter=15 BIC;
   display 'SelectionSummary' 'SelectedEffects';
run;

proc sgplot data=Summary;
   series x=Iteration  y=BIC;
run;
```

The TECH=DSC option in the PROC VARREDUCE statement requests a discriminant analysis of the Heart data table for feature selection. The ODS OUTPUT statement stores the "Selection Summary" table as a local file named *Summary*. The MATRIX=COV option in the PROC VARREDUCE statement requests that selections be done based on the covariance matrix. The BIC option specifies the stop criterion, and the MAXITER= option specifies 15 as the maximum number of iterations. The selection process terminates when the BIC statistic increases in the last three consecutive steps.

Output 36.1.1 and Output 36.1.2 display the selection summary from each iteration and the selected effects.

Output 36.1.1 Selection Summary

The VARREDUCE Procedure

Selection Summary

Iteration	Parameter	Proportion of Variance Explained	SSE	MSE	AIC	AICC	BIC
1	AgeAtStart	0.191310	0.808690	0.00016052	-0.210355	1.789647	-0.210648
2	Smoking	0.220923	0.779077	0.00015467	-0.246867	1.753136	-0.246262
3	Systolic	0.245405	0.754595	0.00014984	-0.278002	1.722002	-0.276499
4	Sex Female	0.255479	0.744521	0.00014787	-0.290648	1.709358	-0.288247
5	Weight_Status Underweight	0.256772	0.743228	0.00014764	-0.291593	1.708415	-0.288294
6	BP_Status Normal	0.257880	0.742120	0.00014745	-0.292291	1.707720	-0.288093
7	Smoking_Status Non-smoker	0.258962	0.741038	0.00014727	-0.292957	1.707057	-0.287861
8	Chol_Status Borderline	0.259547	0.740453	0.00014718	-0.292952	1.707065	-0.286958

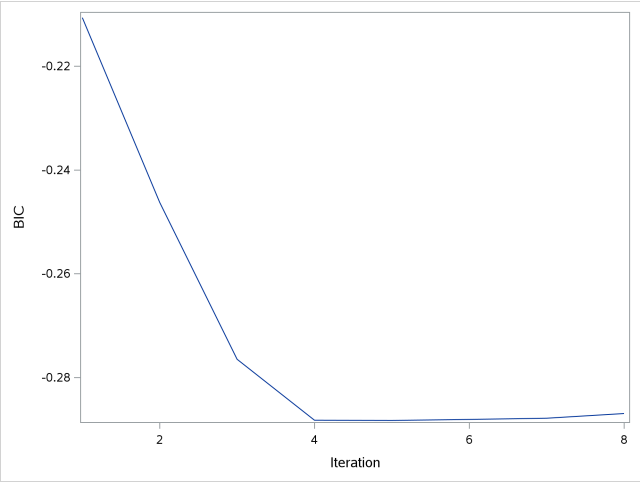
Output 36.1.2 Selected Effects

Selected Effects

Number	Selected Variable	Variable Type
1	AgeAtStart	INTERVAL
2	Smoking	INTERVAL
3	Systolic	INTERVAL
4	Sex	CLASS
5	Weight_Status	CLASS
6	BP_Status	CLASS
7	Smoking_Status	CLASS
8	Chol_Status	CLASS

Output 36.1.3 shows the BIC curve change throughout iterations.

Output 36.1.3 BIC Curve



Example 36.2: Output a Correlation Matrix to a SAS Data File

(View the complete code for this example (varreduce_ex2.sas) in the example repository.)

This example shows how to output a correlation matrix to a SAS data file. The OUTCP= option creates an output data table named mylib.corr.

The following DATA step generates a data table that has 2,000 observations and contains both interval variables (x1–x2) and CLASS variables (a, c1, and c2):

```
data mylib.data1;
   array x{2};
   array c{2};
   do i=1 to 2000;
      a=int(ranuni(1)*2);
      do j=1 to 2;
         x{j}=ranuni(1);
         c{j}=int(ranuni(1)*2);
      end;
      output;
   end;
run;
```

These statements assume that your CAS engine libref is named mylib, but you can substitute any appropriately defined CAS engine libref.

The following statements invoke the VARREDUCE procedure:

```
title  "Output the Correlation Matrix";

proc varreduce data=mylib.data1 matrix=corr outcp=mylib.corr;
   class a;
   reduce unsupervised a x1-x2 /maxsteps=4;
```

```
run;

proc print data=mylib.corr;
run;
```

Output 36.2.1 shows the content of the data file that PROC VARREDUCE generates.

Output 36.2.1 Output the Correlation Matrix

Output the Correlation Matrix

Obs	_ID_	_TYPE_	_VAR_	_vID_	v1	v2	v3	v4
1	1	MEAN/FREQ			1015.00	985.00	0.50	0.50
2	2	N			2000.00	2000.00	2000.00	2000.00
3	3	CORR	a 0	v1	1.00	-1.00	-0.00	-0.02
4	4	CORR	a 1	v2	-1.00	1.00	0.00	0.02
5	5	CORR	x1	v3	-0.00	0.00	1.00	-0.01
6	6	CORR	x2	v4	-0.02	0.02	-0.01	1.00

The _VAR_ column displays the names of all variables and the levels of the CLASS variables. Assuming that you have *n* effects (the total number of interval variables and the levels of CLASS variables), the _vID_ column contains *n* markers, v1 to v*n*, where v*i* denotes the *i*th effect. The column _TYPE_ defines the role of each row. When the _TYPE_ column displays MEAN/FREQ, the corresponding row contains either the mean for an interval variable or the frequency for a level of a CLASS variable. When the _TYPE_ column displays N, the corresponding row contains the number of samples. And when the _TYPE_ column displays CORR, COV, or SSCP, the corresponding row contains a row of the CORR, COV, or SSCP matrix. In this example, the CORR matrix is 4×4, and it resides in the table in rows 3–6 and columns 7–10.

Example 36.3: Output the Correlation Matrix in LIL Format

(View the complete code for this example (varreduce_ex3.sas) in the example repository.)

This example shows how to output a correlation matrix in list-of-list (LIL) format. The OUTCP= option creates an output data table named mylib.corr_lil.

The following DATA step generates a data table that has 2,000 observations and contains both interval variables (x1–x2) and a CLASS variable (a):

```
data mylib.data2;
   array x{2};
   do i=1 to 2000;
      a=int(ranuni(1)*2);
      do j=1 to 2;
         x{j}=ranuni(1);
      end;
      output;
   end;
run;
```

These statements assume that your CAS engine libref is named mylib, but you can substitute any appropriately defined CAS engine libref.

The following statements invoke the VARREDUCE procedure:

```
title  "Output the Correlation Matrix in LIL Format";

proc varreduce data=mylib.data2 matrix=corr outcp=mylib.corr_lil/list eps=0.01;
   class a;
   reduce unsupervised a x1-x2 /maxsteps=4;
run;

proc print data=mylib.corr_lil;
run;
```

Output 36.3.1 shows the correlation matrix in LIL format.

Output 36.3.1 Output the Correlation Matrix in LIL Format

Output the Correlation Matrix in LIL Format

Obs	_TYPE_	_ID_	_NAME1_	_NAME2_	_VAL_
1	S	1	samples		2000.00
2	S	2	nVar		3.00
3	S	3	nEff		4.00
4	F	1	a 0		979.00
5	F	2	a 1		1021.00
6	M	3	x1		0.49
7	M	4	x2		0.50
8	R	1	1	1	1.00
9	R	2	2	1	-1.00
10	R	3	2	2	1.00
11	R	4	3	1	0.03
12	R	5	3	2	-0.03
13	R	6	3	3	1.00
14	R	10	4	4	1.00

The column _TYPE_ defines the type of each row:

- When the _TYPE_ column displays S, the corresponding row contains the statistics of the data table. More specifically, when the _TYPE_ column displays S and the _NAME1_ column displays samples, the _VAL_ column in the corresponding row contains the number of samples in the data table. Similarly, when the _TYPE_ column displays S and the _NAME1_ column displays nVar, the _VAL_ column contains the number of variables. And when the _TYPE_ column displays S and the _NAME1_ column displays nEff, the _VAL_ column in the corresponding row contains the number of effects.
- When the _TYPE_ column displays F, the row contains the frequency of a level of a CLASS variable. In this case, the _NAME1_ column contains the name and level of the CLASS variable.
- When the _TYPE_ column displays M, the row contains the mean of an interval variable. In this case, the _NAME1_ column contains the name of the variable and the _NAME2_ column is empty.
- When the _TYPE_ column displays R, the row contains an entry in the correlation matrix. In this case, the _NAME1_ column contains the row ID, the _NAME2_ column contains the column ID, and the _VAL_ column contains the value.

- When the _TYPE_ column displays V or P, the corresponding row contains an entry of a COV matrix or an SSCP matrix, respectively.

Only entries in the lower triangle of the correlation matrix are written to the file, because the correlation matrix is symmetric. Also, any entry of the matrix whose value is smaller than 0.01 is ignored in the output (EPS = 0.01); this saves storage space.

References

Akaike, H. (1969). "Fitting Autoregressive Models for Prediction." *Annals of the Institute of Statistical Mathematics* 21:243–247.

Cooley, W. W., and Lohnes, P. R. (1971). *Multivariate Data Analysis*. New York: John Wiley & Sons.

Fisher, R. A. (1936). "The Use of Multiple Measurements in Taxonomic Problems." *Annals of Eugenics* 7:179–188.

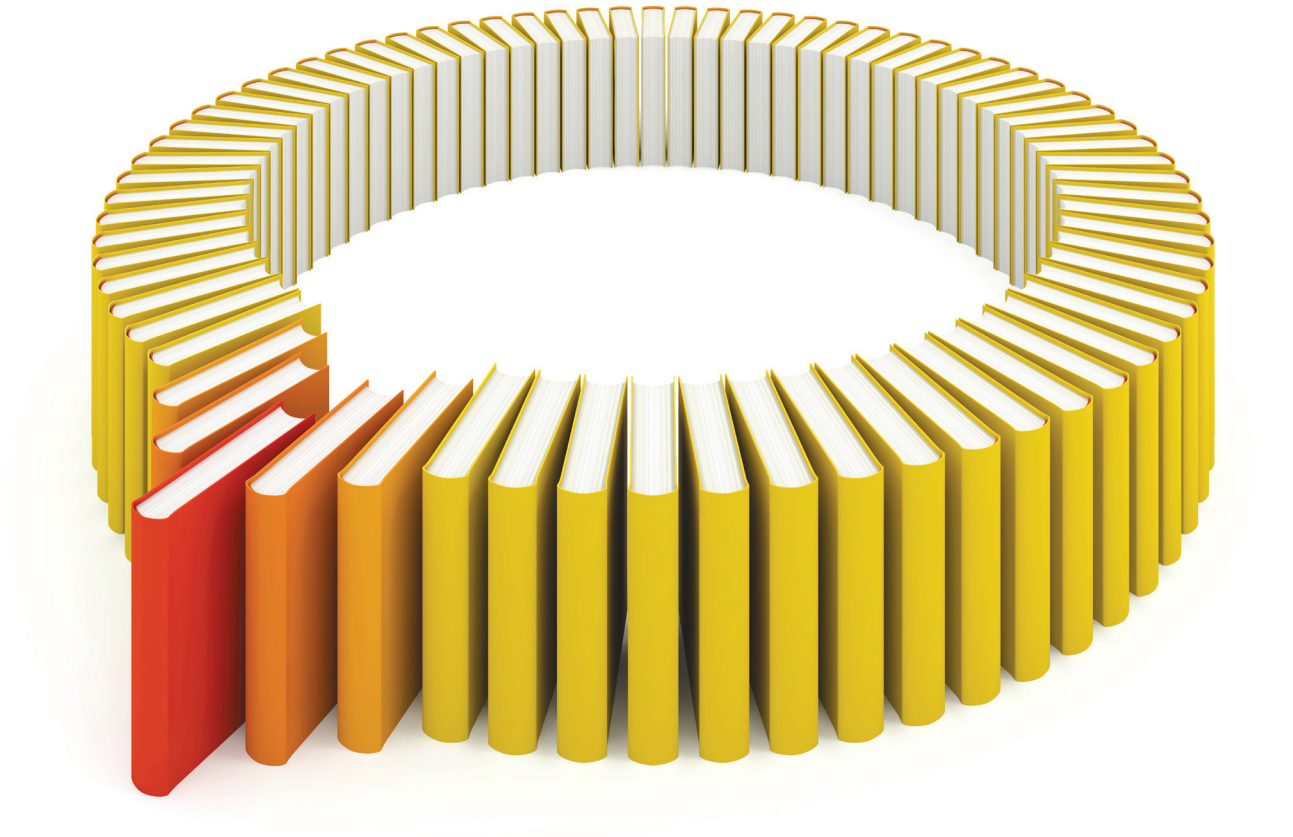
Hurvich, C. M., and Tsai, C.-L. (1989). "Regression and Time Series Model Selection in Small Samples." *Biometrika* 76:297–307.

Jolliffe, I. T. (2002). *Principal Component Analysis*. New York: Springer-Verlag.

Judge, G. G., Griffiths, W. E., Hill, R. C., Lütkepohl, H., and Lee, T.-C. (1985). *The Theory and Practice of Econometrics*. 2nd ed. New York: John Wiley & Sons.

Liu, H., and Motoda, H. (1998). *Feature Selection for Knowledge Discovery and Data Mining*. Norwell, MA: Kluwer Academic.

Schwarz, G. (1978). "Estimating the Dimension of a Model." *Annals of Statistics* 6:461–464.



Gain Greater Insight into Your SAS® Software with SAS Books.

Discover all that you need on your journey to knowledge and empowerment.

support.sas.com/bookstore
for additional books and resources.

sas
THE POWER TO KNOW®

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other brand and product names are trademarks of their respective companies. © 2013 SAS Institute Inc. All rights reserved. S107969US.0613