SAS® Visual Data Mining and Machine Learning 8.1
Statistical Procedures
## Contents

### I General

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>Shared Concepts</td>
<td>7</td>
</tr>
</tbody>
</table>

### II Statistics

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>The GENSELECT Procedure</td>
<td>77</td>
</tr>
<tr>
<td>4</td>
<td>The KCLUS Procedure</td>
<td>135</td>
</tr>
<tr>
<td>5</td>
<td>The LOGSELECT Procedure</td>
<td>167</td>
</tr>
<tr>
<td>6</td>
<td>The NLMOD Procedure</td>
<td>225</td>
</tr>
<tr>
<td>7</td>
<td>The PCA Procedure</td>
<td>255</td>
</tr>
<tr>
<td>8</td>
<td>The PLSMOD Procedure</td>
<td>287</td>
</tr>
<tr>
<td>9</td>
<td>The QTRSELECT Procedure</td>
<td>327</td>
</tr>
<tr>
<td>10</td>
<td>The REGSELECT Procedure</td>
<td>371</td>
</tr>
<tr>
<td>11</td>
<td>The TREESPLIT Procedure</td>
<td>413</td>
</tr>
</tbody>
</table>

### III Utility

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>The ASSESS Procedure</td>
<td>467</td>
</tr>
<tr>
<td>13</td>
<td>The BINNING Procedure</td>
<td>483</td>
</tr>
<tr>
<td>14</td>
<td>The CARDINALITY Procedure</td>
<td>503</td>
</tr>
<tr>
<td>15</td>
<td>The PARTITION Procedure</td>
<td>521</td>
</tr>
<tr>
<td>16</td>
<td>The VARIMPUTE Procedure</td>
<td>535</td>
</tr>
<tr>
<td>17</td>
<td>The VARREDUCE Procedure</td>
<td>545</td>
</tr>
</tbody>
</table>

### Subject Index

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>575</td>
</tr>
</tbody>
</table>

### Syntax Index

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>587</td>
</tr>
</tbody>
</table>
Part I

General
Overview of SAS Visual Data Mining and Machine Learning Statistical Procedures

SAS Visual Data Mining and Machine Learning statistical procedures provide predictive modeling tools that have been specially developed to take advantage of the distributed environment that the SAS Viya platform provides. Methods include linear regression, logistic regression, quantile regression, generalized linear 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 either statistical procedures or data mining and machine learning procedures in SAS Visual Data Mining and Machine Learning.
Chapter 1, this chapter, provides an overview of SAS Visual Data Mining and Machine Learning statistical 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. This chapter also documents the following statements that are used across a number of procedures: CLASS, CODE, EFFECT, PARTITION, and SELECTION.

Part II describes the statistical procedures and Part III describes the utility procedures in SAS Visual Data Mining and Machine Learning. In each of these parts, the chapters 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** 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 in an ODS destination statement as follows:

```sas
ods html style=HTMLBlue;
ods html close;
ods pdf style=PearlJ;
ods pdf close;
```

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

```sas
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](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](http://support.sas.com/techsup) for more information.
Chapter 2
Shared Concepts

Contents

Introduction to Shared Concepts ........................................... 8
Using CAS Sessions and CAS Engine Librefs .......................... 8
Loading a SAS Data Set onto a CAS Server ............................ 9
Syntax Common to SAS Visual Data Mining and Machine Learning Statistical Procedures .......................... 10
CLASS Statement ............................................................. 10
CODE Statement ................................................................ 14
DISPLAY Statement ........................................................... 15
DISPLAYOUT Statement ....................................................... 16
EFFECT Statement .............................................................. 17
  Collection Effects .......................................................... 19
  Multimember Effects ....................................................... 19
  Polynomial Effects .......................................................... 21
  Spline Effects ................................................................ 24
  Splines and Spline Bases ................................................... 27
  ODS Table Names .......................................................... 31
PARTITION Statement ......................................................... 32
SELECTION Statement ........................................................ 32
Optimization Options .......................................................... 38
Details for SAS Visual Statistics Procedures ......................... 43
  Levelization of Classification Variables ................................ 43
  Specification and Parameterization of Model Effects ............... 45
    Effect Operators .......................................................... 46
    GLM Parameterization of Classification Variables and Effects .. 48
    Nonsingular Parameterization .......................................... 52
    Class Variable Parameterization with Unbalanced Designs ....... 55
Model Selection Methods ...................................................... 56
  Full Model Fitted ............................................................ 56
  Forward Selection .......................................................... 56
  Backward Elimination ...................................................... 58
  Stepwise Selection .......................................................... 59
  Forward-Swap Selection .................................................... 61
  Least Angle Regression ..................................................... 61
  LASSO Selection ............................................................ 62
  Adaptive LASSO Selection ................................................ 63
  Group LASSO Selection ..................................................... 63
Introduction to Shared Concepts

SAS Visual Data Mining and Machine Learning statistical 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 Data Mining and Machine Learning Statistical Procedures” on page 10 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 43 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 mycas cas sessref=mysess;

The CAS statement creates the CAS session named mysess, and the LIBNAME statement creates the mycas 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: Language Reference. 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, mycas 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:

```sas
data mycas.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:

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

- You can use the CASUTIL procedure as follows:
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: Language Reference.

The mycas 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 mycas caslib:

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

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

```
proc delete data = mycas.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: Accessing and Manipulating Data.

---

**Syntax Common to SAS Visual Data Mining and Machine Learning Statistical Procedures**

**CLASS Statement**

```
CLASS variable <(options)> ... < variable <(options)> >> < global-options > ;
```

This section applies to the following procedures: GENSELECT, LOGSELECT, PLSMOD, REGSELECT, 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 43.

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>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DESCENDING</td>
<td>Reverses the sort order</td>
</tr>
<tr>
<td>MISSING</td>
<td>Treats missing values as valid levels</td>
</tr>
<tr>
<td>ORDER=</td>
<td>Specifies the sort order for the levels</td>
</tr>
<tr>
<td>PARAM=</td>
<td>Specifies the parameterization of the variable</td>
</tr>
<tr>
<td>REF=</td>
<td>Specifies the reference level of the variable</td>
</tr>
<tr>
<td>SPLIT</td>
<td>Allows design columns for a variable to enter or leave the model independently</td>
</tr>
</tbody>
</table>

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

specifies the sort order for the levels of classification variables. This ordering determines which parameters in the model correspond to each level in the data.

The following table shows how values of the ORDER= option are interpreted.

<table>
<thead>
<tr>
<th>Value of ORDER=</th>
<th>Levels Sorted By</th>
</tr>
</thead>
<tbody>
<tr>
<td>FORMATTED</td>
<td>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.</td>
</tr>
<tr>
<td>FREQ</td>
<td>Descending frequency count (levels that have more observations come earlier in the order)</td>
</tr>
<tr>
<td>INTERNAL</td>
<td>Unformatted value. The sort order is machine-dependent.</td>
</tr>
</tbody>
</table>
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*. 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>
<thead>
<tr>
<th>Value of PARAM=</th>
<th>Coding</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>EFFECT</strong></td>
<td>Effect coding. The REF= option in the CLASS statement determines the reference level.</td>
</tr>
<tr>
<td><strong>GLM</strong></td>
<td>Less-than-full-rank reference cell coding. This <em>keyword</em> 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.</td>
</tr>
<tr>
<td>**ORDINAL</td>
<td>THERMOMETER**</td>
</tr>
<tr>
<td>**POLYNOMIAL</td>
<td>POLY**</td>
</tr>
<tr>
<td>**REFERENCE</td>
<td>REF**</td>
</tr>
<tr>
<td><strong>ORTHEFFECT</strong></td>
<td>Orthogonalizes PARAM=EFFECT coding. The REF= option in the CLASS statement determines the reference level.</td>
</tr>
<tr>
<td>**ORTHORDINAL</td>
<td>ORTHOTHERM**</td>
</tr>
<tr>
<td><strong>ORTHPOLY</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>ORTHREF</strong></td>
<td>Orthogonalizes PARAM=REFERENCE coding. The REF= option in the CLASS statement determines the reference level.</td>
</tr>
</tbody>
</table>

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 45.
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.

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:

```plaintext
proc regselect;
   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:

```plaintext
proc regselect;
   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;
  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;
  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, REGSELECT, and TREESPLIT.

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

**Table 2.3  CODE Statement Options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMMENT</td>
<td>Adds comments to the generated code</td>
</tr>
<tr>
<td>FILE=</td>
<td>Names the file where the generated code is saved</td>
</tr>
<tr>
<td>FORMATWIDTH=</td>
<td>Specifies the numeric format width for the regression coefficients</td>
</tr>
<tr>
<td>INDENTSIZE=</td>
<td>Specifies the number of spaces to indent the generated code</td>
</tr>
<tr>
<td>LABELID=</td>
<td>Specifies a number used to construct names and labels</td>
</tr>
<tr>
<td>LINESIZE=</td>
<td>Specifies the line size for the generated code</td>
</tr>
<tr>
<td>NOTRIM</td>
<td>Compares formatted values, including blank padding</td>
</tr>
<tr>
<td>PCATALL</td>
<td>Generates probabilities for all levels of categorical response</td>
</tr>
</tbody>
</table>

If you do not specify the FILE= 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.
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 specified in the FILENAME statement is opened; otherwise, if your SAS client has a default path, an external file named filename is created.

FORMATWIDTH=width
specifies the width to use in formatting derived numbers such as parameter estimates. You can specify a value in the range 4 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.

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.

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.

PCATALL
generates probabilities for all levels of categorical response variables.

---

DISPLAY Statement

DISPLAY < table-list > < / options > ;

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

The DISPLAY statement enables you to specify a list of ODS 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, all ODS tables are sent to the client and then the client creates a subset. If both DISPLAY and ODS statements are used
together, the DISPLAY statement takes precedence over the ODS statements. For more information about ODS, see SAS Output Delivery System: Procedures Guide.

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

**CASESENSITIVE**
performs a case-sensitive comparison of table names in the table-list to ODS 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 ODS tables except those specified in the table-list.

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

**TRACE**
displays the ODS table names, labels, and paths.

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

A path is a table name that is prefixed with dot-separated grouping information. For example, a SelectionSummary table that is produced by a procedure 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 ODS 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 starts with a “/” or a “!”. 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.

---

**DISPLAYOUT Statement**

DISPLAYOUT table-spec-list < / options > ;

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

The DISPLAYOUT statement enables you to create CAS 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 CAS output tables to create. Each entry in the list has either a key or a key=value format:
EFFECT Statement

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

This section applies to the following procedures: GENSELECT, LOGSELECT, PLSMOD, 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 48. 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:

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

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 (/):

- **NOREPLACE**: does not replace an existing CAS output table of the same name.
- **REPEATED**: replicates the CAS output tables on all nodes.
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 19.

- **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 19.

- **POLYNOMIAL | POLY** specifies a multivariate polynomial effect in the specified numeric variables. For more information, see the section “Polynomial Effects” on page 21.

- **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 24.

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 2.4 after a slash following the var-list.

<table>
<thead>
<tr>
<th>Table 2.4</th>
<th>EFFECT Statement Options</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Option</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>Collection Effects Options</td>
<td></td>
</tr>
<tr>
<td>DETAILS</td>
<td>Displays the constituents of the collection effect</td>
</tr>
<tr>
<td>Multimember Effects Options</td>
<td></td>
</tr>
<tr>
<td>DETAILS</td>
<td>Displays the levels of the multimember effect</td>
</tr>
<tr>
<td>NOEFFECT</td>
<td>Specifies that observations whose levels are all missing for the multimember variables should have 0 values in the corresponding design matrix columns</td>
</tr>
<tr>
<td>STDIZE</td>
<td>Standardizes the design matrix entries so that each observation has a sum of 1</td>
</tr>
<tr>
<td>WEIGHT=</td>
<td>Specifies the weight variable for the contributions of each classification effect</td>
</tr>
<tr>
<td>Polynomial Effects Options</td>
<td></td>
</tr>
<tr>
<td>DEGREE=</td>
<td>Specifies the degree of the polynomial</td>
</tr>
<tr>
<td>DETAILS</td>
<td>Displays details of the specified polynomial</td>
</tr>
<tr>
<td>MDEGREE=</td>
<td>Specifies the maximum degree of any variable in a term of the polynomial</td>
</tr>
</tbody>
</table>
Table 2.4  continued

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOSEPARATE</td>
<td>Treats the polynomial as a single effect with multiple degrees of freedom</td>
</tr>
<tr>
<td>STANDARDIZE=</td>
<td>Specifies centering and scaling suboptions for the variables that define the polynomial</td>
</tr>
</tbody>
</table>

**Spline Effects Options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BASIS=</td>
<td>Specifies the type of basis (B-spline basis or truncated power function basis) for the spline effect</td>
</tr>
<tr>
<td>DATABOUNDARY</td>
<td>Uses the extremes of the data as boundary knots for a B-spline basis</td>
</tr>
<tr>
<td>DEGREE=</td>
<td>Specifies the degree of the spline effect</td>
</tr>
<tr>
<td>DETAILS</td>
<td>Displays the knots and locations for each spline basis function</td>
</tr>
<tr>
<td>KNOTMAX=</td>
<td>Requests equally spaced right-side boundary knots starting at the variables’ maximum and ending at the KNOTMAX= value</td>
</tr>
<tr>
<td>KNOTMETHOD=</td>
<td>Specifies how to construct the knots for the spline effect</td>
</tr>
<tr>
<td>KNOTMIN=</td>
<td>Requests equally spaced left-side boundary knots starting at the KNOTMAX= value and ending at the variables’ minimum value</td>
</tr>
<tr>
<td>NATURALCUBIC</td>
<td>Specifies a natural cubic spline basis for the spline effect</td>
</tr>
<tr>
<td>SEPARATE</td>
<td>Treats the spline basis for each variable as a separate effect when multiple variables are specified</td>
</tr>
<tr>
<td>SPLIT</td>
<td>Treats each design matrix column as a separate effect for selection methods</td>
</tr>
</tbody>
</table>

**Collection Effects**

\[ \text{EFFECT } \text{effect-name}=\text{COLLECTION} (\text{var-list} < / \text{DETAILS} >) ; \]

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 \text{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**

\[ \text{EFFECT } \text{effect-name}=\text{MULTIMEMBER} (\text{var-list} < / \text{mm-options} >) ; \]

\[ \text{EFFECT } \text{effect-name}=\text{MM} (\text{var-list} < / \text{mm-options} >) ; \]

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:

<table>
<thead>
<tr>
<th>Student</th>
<th>Score</th>
<th>Teacher1</th>
<th>Teacher2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mary</td>
<td>87</td>
<td>Tobias</td>
<td>Cohen</td>
</tr>
<tr>
<td>Tom</td>
<td>89</td>
<td>Rodriguez</td>
<td>Tobias</td>
</tr>
<tr>
<td>Fred</td>
<td>82</td>
<td>Cohen</td>
<td>Cohen</td>
</tr>
<tr>
<td>Jane</td>
<td>88</td>
<td>Tobias</td>
<td>.</td>
</tr>
<tr>
<td>Jack</td>
<td>99</td>
<td>.</td>
<td>.</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Student</th>
<th>Cohen</th>
<th>Rodriguez</th>
<th>Tobias</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mary</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Tom</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Fred</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Jane</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Jack</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
</tbody>
</table>

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 steps yield the identical analysis:

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

```plaintext
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 \( x_1, x_2, x_1^2, x_1 x_2, x_2^2, x_1 x_2^2, x_1^2 x_2 \) and \( x_1^2 x_2^2 \):

```
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

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

Table 2.5 shows how the center and scale are computed for each of the supported methods.

**Table 2.5** Center and Scale Estimates by Method

<table>
<thead>
<tr>
<th>Method</th>
<th>Center</th>
<th>Scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>RANGE</td>
<td>((x_{(n)} + x_{(1)})/2)</td>
<td>((x_{(n)} - x_{(1)})/2)</td>
</tr>
<tr>
<td>MOMENTS</td>
<td>(\bar{x} = \sum_{i=1}^{n} f_i x_i / F)</td>
<td>(\sqrt{\sum_{i=1}^{n} f_i (x_i - \bar{x})^2 / (F - 1)})</td>
</tr>
<tr>
<td>WMOMENTS</td>
<td>(\bar{x}<em>w = \sum</em>{i=1}^{n} w_i f_i x_i / WF)</td>
<td>(\sqrt{\sum_{i=1}^{n} w_i f_i (x_i - \bar{x}_w)^2 / (F - 1)})</td>
</tr>
</tbody>
</table>

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

**CENTER**

centers but does not scale the variables. For a variable \(x\),

\[ s_x = x - \text{center} \]

**CENTERSCALE**

centers and scales the variables. This is the default if you do not specify a `standardization-opt`. For a variable \(x\),

\[ s_x = \frac{x - \text{center}}{\text{scale}} \]
NONE
performs no standardization.

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

$$s_x = \frac{x}{\text{scale}}$$

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 27. 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 _Si 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 _Si 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, \(\text{STARTSCALE}=0\).
- **ENDSCALE=**\(n\) specifies the end scale, where \(n\) is a positive integer. By default, \(\text{ENDSCALE}=7\).

**RANGEFRACTIONS(\(fraction-list\))** places internal knots at each fraction of the ranges of the variables in the EFFECT statement. For example, if variable \(x_1\) ranges between 1 and 3 and variable \(x_2\) ranges between 0 and 20, then the following EFFECT statement uses internal knots 1.2, 2, and 2.5 for variable \(x_1\) and internal knots 2, 10, and 15 for variable \(x_2\):

```
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 \(i\)th function, \(i = 1, 2, \ldots, n-2\), is 0 to the left of the \(i\)th knot, which is called the “break knot.” For more information about this basis, see the section “Splines and Spline Bases” on page 27. 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 REGSELECT 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
are simply polynomials of degree .

More formally, suppose you specify knots . Then a spline of degree is a function
with 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 is a polynomial of degree . The requirement that has continuous derivatives is
satisfied by requiring that the function values and all derivatives up to order of the adjacent polynomials
at each knot match.

A counting argument yields the number of parameters that define a spline with knots. There are polynomials of degree , producing coefficients. However, there are restrictions at each
of the knots, so the number of free parameters is . In mathematical
terminology this says that the dimension of the vector space of splines of degree on distinct knots is . If you have 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 and a repeated knot that has multiplicity , the piecewise polynomials that join such a knot
are required to have only matching derivatives. Note that this increases the number of free parameters
by but also decreases the number of distinct knots by . Hence the dimension of the vector space
of splines of degree with knots is still , provided that any repeated knot has a multiplicity less
than or equal to .
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 $x = 1$](chart)

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](image)

**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 Boundary Knots at 0 and 1](image)
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

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>BSplineDetails</td>
<td>B-spline basis details</td>
<td>BASIS=BSPLINE</td>
</tr>
<tr>
<td>CollectionLevelInfo</td>
<td>Levels of collection effects</td>
<td></td>
</tr>
<tr>
<td>MMLLevelInfo</td>
<td>Levels of multimember effects</td>
<td></td>
</tr>
<tr>
<td>PolyDetails</td>
<td>Number of variables and columns, polynomial\degree, and standardization method</td>
<td></td>
</tr>
<tr>
<td>PolyScaling</td>
<td>Centering and scaling details</td>
<td>STANDARDIZE</td>
</tr>
<tr>
<td>SplineKnots</td>
<td>knot and boundary knot values</td>
<td></td>
</tr>
<tr>
<td>TPFSplineDetails</td>
<td>Truncated power function spline basis details</td>
<td>BASIS=TPF or NATURALCUBIC</td>
</tr>
</tbody>
</table>
PARTITION Statement

```
PARTITION partition-option ;
```

This section applies to the following procedures: GENSELECT, KCLUS, LOGSELECT, NLMOD, PCA, 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 66. 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 of the observations 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 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, 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 used 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 56. By default, METHOD=STEPWISE.

- **NONE** specifies no model selection.
- **BACKWARD** specifies backward elimination. This method starts with all effects in the model and deletes effects.
- **FORWARD** specifies forward selection. This method starts with no effects in the model and adds effects.
- **STEPWISE** specifies stepwise regression. This method is similar to the FORWARD method except that effects already in the model do not necessarily stay there.
- **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 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** 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.

Table 2.7 lists the applicable `method-options` for each of these methods.
Table 2.7  Applicable method-options by method

<table>
<thead>
<tr>
<th>method-option</th>
<th>FORWARD</th>
<th>BACKWARD</th>
<th>STEPWISE</th>
<th>LAR</th>
<th>LASSO</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADAPTIVE</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHOOSE =</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>COMPETITIVE</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>CRITERION =</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FAST</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>LSCEFFS</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAXEFFECTS =</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>MAXSTEPS =</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>MINEFFECTS =</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>SELECT =</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>SENTRY =</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>SLSTAY =</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>STOP =</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

The syntax of the method-options that you can specify in parentheses after the SELECTION= option method follows. As described in Table 2.7, not all selection method-options are applicable to every method.

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

**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 do not specify the CHOOSE= 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.

**COMPETITIVE**  
is applicable as a method-option only when METHOD=STEPWISE and the SELECT= criterion is not SL. If you specify the COMPETITIVE 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= option.
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 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 option is available only when METHOD=BACKWARD is specified. It is computationally efficient because the model is not fit after removal of each effect.

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. This option is ignored when METHOD=BACKWARD is specified. If, at some step of the selection process, the model contains the specified maximum number of effects, then no candidates for addition are considered.

MAXSTEPS=n

specifies the maximum number of selection steps that are performed. 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.

MINEFFECTS=n

specifies the minimum number of effects in any model that is considered during backward selection. This 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.

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 option is not valid when METHOD=LAR or METHOD=LASSO.

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.
**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 do not specify the STOP= option but do specify the SELECT= option, then the criterion specified in the SELECT= option is also used as the STOP= criterion.

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= 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= 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.

HIERARCHY=NONE | SINGLE | SINGLECLASS
specifies whether and how the model hierarchy requirement is applied. 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.

SELECTION=NONE | BACKWARD | FORWARD | STEPWISE | LAR | LASSO
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, METHOD=LAR, or METHOD=LASSO, then \( n \) is ignored and STOPHORIZON=1 is used. By default, STOPHORIZON=3.

### Optimization Options

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

The following options are typically available in the PROC statement of the procedures in this book that perform optimizations:

**ABSCONV=\( r \)**

**ABSTOL=\( r \)**

specifies an absolute function convergence criterion. For minimization, termination requires \( f(\psi^{(k)}) \leq r \), where \( \psi \) 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=\( r < n \)**

**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:

\[
| f(\psi^{(k-1)}) - f(\psi^{(k)}) | \leq r
\]

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

\[
\max_j | g_j(\psi^{(k)}) | \leq r
\]

Here, \( \psi \) 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 \( j \)th 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,
\[ \| \psi^{(k)} - \psi^{(k-1)} \|_2 \leq r \]
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 \( \xi^{(k)} \) that has the smallest function value to the other \( p \) simplex points \( \psi_l^{(k)} \neq \xi^{(k)} \):
\[ \delta^{(k)} = \sum_{\psi_l \neq \psi} \| \psi_l^{(k)} - \xi^{(k)} \|_1 \]
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.

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,
\[ \frac{|f(\psi^{(k)}) - f(\psi^{(k-1)})|}{|f(\psi^{(k-1)})|} \leq r \]
Here, \( \psi \) denotes the vector of parameters that participate in the optimization, and \( f(\cdot) \) is the objective function. The same formula is used for the NMSIMP technique, but \( \psi^{(k)} \) is defined as the vertex that has the lowest function value and \( \psi^{(k-1)} \) 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 \( \epsilon \) is the machine precision.

FCONV2=r < n>
FTOL2=r < n>

specifies a second function convergence criterion: For all techniques except NMSIMP, termination requires a small predicted reduction,
\[ df^{(k)} \approx f(\psi^{(k)}) - f(\psi^{(k)} + s^{(k)}) \]
of the objective function. The predicted reduction

\[ df^{(k)} = -g^{(k)'}s^{(k)} - \frac{1}{2}s^{(k)'}H^{(k)}s^{(k)} \]

\[ = -\frac{1}{2}s^{(k)'}g^{(k)} \leq r \]

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

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

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

\[ \sqrt{\frac{1}{n + 1} \sum_l \left[ f(\psi_l^{(k)}) - \bar{f}(\psi^{(k)}) \right]^2} \leq r \]

where \( \bar{f}(\psi^{(k)}) = \frac{1}{p+1} \sum_l f(\psi_l^{(k)}) \). If there are \( p_{\text{act}} \) boundary constraints active at \( \psi^{(k)} \), the mean and standard deviation are computed only for the \( n + 1 - p_{\text{act}} \).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 \( r = 1 \times 10^{-6} \) 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:

\[ \frac{g(\psi^{(k)})' [H^{(k)}]^{-1}g(\psi^{(k)})}{|f(\psi^{(k)})|} \leq r \]

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

\[ \frac{\| g(\psi^{(k)}) \|_2^2}{\| g(\psi^{(k)}) - g(\psi^{(k-1)}) \|_2} \frac{\| s(\psi^{(k)}) \|_2}{|f(\psi^{(k)})|} \leq r \]

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:

$$\max_j \frac{|g_j(\psi^{(k)})|}{\sqrt{f(\psi^{(k)}) H_{j,j}^{(k)}}} \leq r$$

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
- 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
- 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.
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 68.

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

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

and $\hat{\beta}_j^{(i)}$ is the estimate of the $j$th parameter at iteration $i$. The default value is $r = 1E-8$ for the NMSIMP technique and $r = 0$ otherwise.
This section applies to the following procedures: GENSELECT, LOGSELECT, NLMOD, PLSMOD, REGSELECT, 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 2.8. 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 2.8 apply two different formats to the variable X.

<table>
<thead>
<tr>
<th>Obs</th>
<th>A</th>
<th>x</th>
<th>FORMAT x 3.0</th>
<th>FORMAT x 3.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1.09</td>
<td>1</td>
<td>1.1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1.13</td>
<td>1</td>
<td>1.1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1.27</td>
<td>1</td>
<td>1.3</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>2.26</td>
<td>2</td>
<td>2.3</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>2.48</td>
<td>2</td>
<td>2.5</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>3.34</td>
<td>3</td>
<td>3.3</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>3.34</td>
<td>3</td>
<td>3.3</td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td>3.14</td>
<td>3</td>
<td>3.1</td>
</tr>
</tbody>
</table>

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 2.8 leads to the level assignment in Table 2.9.

<table>
<thead>
<tr>
<th>Obs</th>
<th>A Value</th>
<th>X Value</th>
<th>FORMAT x 3.0 Value</th>
<th>FORMAT x 3.1 Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1.09</td>
<td>1</td>
<td>1.1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1.13</td>
<td>1</td>
<td>1.1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1.27</td>
<td>1</td>
<td>1.3</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>2.26</td>
<td>2</td>
<td>2.3</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>2.48</td>
<td>2</td>
<td>2.5</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>3.34</td>
<td>3</td>
<td>3.3</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>3.34</td>
<td>3</td>
<td>3.3</td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td>3.14</td>
<td>3</td>
<td>3.1</td>
</tr>
</tbody>
</table>
Chapter 2: Shared Concepts

Table 2.9 continued

<table>
<thead>
<tr>
<th>Obs</th>
<th>A Value</th>
<th>X FORMAT</th>
<th>x 3.0 FORMAT</th>
<th>x 3.1 FORMAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>1.13</td>
<td>1</td>
<td>1.1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1.27</td>
<td>1</td>
<td>1.3</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>2.26</td>
<td>.</td>
<td>2.3</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>2.48</td>
<td>2</td>
<td>2.5</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>3.34</td>
<td>3</td>
<td>3.3</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>3.14</td>
<td>3</td>
<td>3.1</td>
</tr>
</tbody>
</table>

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 2.10 shows how values of the ORDER= option are interpreted.

<table>
<thead>
<tr>
<th>Value of ORDER=</th>
<th>Levels Sorted By</th>
</tr>
</thead>
<tbody>
<tr>
<td>FORMATTED</td>
<td>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.</td>
</tr>
<tr>
<td>FREQ</td>
<td>Descending frequency count (levels that have the most observations come first in the order)</td>
</tr>
<tr>
<td>INTERNAL</td>
<td>Unformatted value. The sort order is machine-dependent.</td>
</tr>
</tbody>
</table>

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 SAS Language Reference: Concepts.

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 2.11 displays the results of levelizing the values in Table 2.8 when the MISSING option is in effect.

Table 2.11 Values and Levels When the MISSING Option Is Specified

<table>
<thead>
<tr>
<th>Obs</th>
<th>A Value</th>
<th>X Value</th>
<th>FORMAT x 3.0 Value</th>
<th>FORMAT x 3.1 Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1.09</td>
<td>1</td>
<td>1.1</td>
</tr>
</tbody>
</table>
Specification and Parameterization of Model Effects

Table 2.11  continued

<table>
<thead>
<tr>
<th>Obs</th>
<th>Value</th>
<th>Level</th>
<th>X Value</th>
<th>Level</th>
<th>FORMAT x 3.0 Value</th>
<th>Level</th>
<th>FORMAT x 3.1 Value</th>
<th>Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1.13</td>
<td>3</td>
<td>1.1</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1.27</td>
<td>4</td>
<td>1.3</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2.26</td>
<td>5</td>
<td>2.3</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>2</td>
<td>2.48</td>
<td>6</td>
<td>2.5</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>3</td>
<td>3.34</td>
<td>8</td>
<td>3.3</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>3</td>
<td>3.34</td>
<td>8</td>
<td>3.3</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>3</td>
<td>3.14</td>
<td>7</td>
<td>3.1</td>
<td>6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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: GENSELECT, LOGSELECT, NLMOD, PLSMOD, and REGSELECT.

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

\[ X\beta \]

or

\[ X\beta + Z\gamma \]

The model matrices \( X \) and \( Z \) 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 17.

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 43. 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 46.

Procedures in this book that have a CLASS statement support a general linear model (GLM) parameterization and a reference parameterization 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 48 and “Nonsingular Parameterization” on page 52.

Effect Operators

Table 2.12 summarizes the operators that are available for selecting and constructing effects. These operators are discussed in the following sections.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interaction</td>
<td>A*B</td>
<td>Crosses the levels of the effects</td>
</tr>
<tr>
<td>Nesting</td>
<td>A(B)</td>
<td>Nests A levels within B levels</td>
</tr>
<tr>
<td>Bar operator</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>At sign operator</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>Dash operator</td>
<td>A1-A10</td>
<td>Specifies sequentially numbered variables</td>
</tr>
<tr>
<td>Colon operator</td>
<td>A:</td>
<td>Specifies variables with common prefix</td>
</tr>
<tr>
<td>Double dash operator</td>
<td>A- -C</td>
<td>Specifies sequential variables in data set order</td>
</tr>
</tbody>
</table>

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:

```plaintext
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:
• Crossed and nested groups of variables are combined. For example, \( A(B) \mid C(D) \) generates \( A*C(B \mid D) \), among other terms.

• Duplicate variables are removed. For example, \( A(C) \mid B(C) \) generates \( A*B(C \mid C) \), 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, \( A(B) \mid B(D \mid E) \) generates \( A*B(B \mid D \mid E) \), 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:

\[
\text{model } Y = A|B|C@2;
\]

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

\[
\text{model } Y = A B C A*B A*C B*C;
\]

More examples of using the bar and at operators follow:

\[
\begin{align*}
A \mid C(B) & \quad \text{is equivalent to} \quad A \ C(B) \ A*C(B) \\
A(B) \mid C(B) & \quad \text{is equivalent to} \quad A(B) \ C(B) \ A*C(B) \\
A(B) \mid B(D \ E) & \quad \text{is equivalent to} \quad A(B) \ B(D \ E) \\
A \mid B(A) \mid C & \quad \text{is equivalent to} \quad A \ B(A) \ C \ A*C \ B*C(A) \\
A \mid B(A) \mid C@2 & \quad \text{is equivalent to} \quad A \ B(A) \ C \ A*C \\
A \mid B \mid C \mid D@2 & \quad \text{is equivalent to} \quad A \ B \ A*B \ C \ A*C \ B*C \ D \ A*D \ B*D \ C*D \\
A*B(C*D) & \quad \text{is equivalent to} \quad A*B(C \ D)
\end{align*}
\]

**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:

\[
\begin{align*}
\text{model } Y &= X1 \ X2 \ X3 \ X4 \ X5 \ X6 \ X7 \ X8 \ X9; \\
\text{model } Y &= X1-X9; \\
\text{model } Y &= X:;
\end{align*}
\]

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 \( X_3 \) through \( X_8 \) 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 2.13 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>
<thead>
<tr>
<th>Effect</th>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intercept</td>
<td>Default</td>
<td>Intercept (unless the NOINT option is specified)</td>
</tr>
<tr>
<td>Regression</td>
<td>( X ) ( Z )</td>
<td>Continuous variables</td>
</tr>
<tr>
<td>Polynomial</td>
<td>( X*Z )</td>
<td>Interaction of continuous variables</td>
</tr>
<tr>
<td>Main</td>
<td>( A ) ( B )</td>
<td>CLASS variables</td>
</tr>
<tr>
<td>Interaction</td>
<td>( A*B )</td>
<td>Crossing of CLASS variables</td>
</tr>
<tr>
<td>Nested</td>
<td>( A(B) )</td>
<td>Main effect ( A ) nested within CLASS effect ( B )</td>
</tr>
<tr>
<td>Continuous-by-class</td>
<td>( X*A )</td>
<td>Crossing of continuous and CLASS variables</td>
</tr>
<tr>
<td>Continuous-nesting-class</td>
<td>( X(A) )</td>
<td>Continuous variable ( X_1 ) nested within CLASS variable ( A )</td>
</tr>
<tr>
<td>General</td>
<td>( X<em>Z</em>A(B) )</td>
<td>Combinations of different types of effects</td>
</tr>
</tbody>
</table>

Table 2.14 shows some examples of MODEL statements that use various types of effects.

<table>
<thead>
<tr>
<th>Specification</th>
<th>Type of Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>model ( Y=X; )</td>
<td>Simple regression</td>
</tr>
<tr>
<td>model ( Y=X \ Z; )</td>
<td>Multiple regression</td>
</tr>
<tr>
<td>model ( Y=X \ X*X; )</td>
<td>Polynomial regression</td>
</tr>
<tr>
<td>model ( Y=A; )</td>
<td>One-way analysis of variance (ANOVA)</td>
</tr>
<tr>
<td>model ( Y=A \ B \ C; )</td>
<td>Main-effects ANOVA</td>
</tr>
<tr>
<td>model ( Y=A \ B A*B; )</td>
<td>Factorial ANOVA with interaction</td>
</tr>
<tr>
<td>model ( y=A \ B(A) \ C(B \ A); )</td>
<td>Nested ANOVA</td>
</tr>
</tbody>
</table>
Table 2.14  continued

<table>
<thead>
<tr>
<th>Specification</th>
<th>Type of Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>model Y=A X;</td>
<td>Analysis of covariance (ANCOVA)</td>
</tr>
<tr>
<td>model Y=A X(A);</td>
<td>Separate-slopes regression</td>
</tr>
<tr>
<td>model Y=A X X*A;</td>
<td>Homogeneity-of-slopes regression</td>
</tr>
</tbody>
</table>

**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, $X \mid X \mid X$ 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 2.15 is an example where $\beta_0$ denotes the intercept and $A$ and $B$ are classification variables that have two and three levels, respectively.

Table 2.15  Example of Main Effects

<table>
<thead>
<tr>
<th>Data</th>
<th>I</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>$\beta_0$</td>
<td>A1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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 2.16).

Table 2.16 Example of Interaction Effects

<table>
<thead>
<tr>
<th>Data</th>
<th>I</th>
<th>A</th>
<th>B</th>
<th>A*B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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, \( A \mid B \mid C \) 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, \( A \mid B \mid C \mid D @2 \) 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):

```plaintext
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 2.17).

Table 2.17 Example of Nested Effects

<table>
<thead>
<tr>
<th>Data</th>
<th>I</th>
<th>A</th>
<th>B(A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>( \beta_0 )</td>
<td>A1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Table 2.17  continued

<table>
<thead>
<tr>
<th>Data</th>
<th>I</th>
<th>A</th>
<th>B(A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**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 2.18).

Table 2.18  Example of Continuous-Nesting-Class Effects

<table>
<thead>
<tr>
<th>Data</th>
<th>I</th>
<th>A</th>
<th>X</th>
<th>( \beta_0 )</th>
<th>A1</th>
<th>A2</th>
<th>X(A1)</th>
<th>X(A2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>21</td>
<td>0</td>
</tr>
<tr>
<td>24</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>24</td>
<td>0</td>
</tr>
<tr>
<td>22</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>22</td>
<td>0</td>
</tr>
<tr>
<td>28</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>28</td>
</tr>
<tr>
<td>19</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>19</td>
</tr>
<tr>
<td>23</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>23</td>
</tr>
</tbody>
</table>

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 2.19 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 2.18.

Table 2.19  Example of Continuous-by-Class Effects

<table>
<thead>
<tr>
<th>Data</th>
<th>I</th>
<th>X</th>
<th>A</th>
<th>X*A</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>A</td>
<td>( \beta_0 )</td>
<td>X</td>
<td>A1</td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>1</td>
<td>21</td>
<td>1</td>
</tr>
<tr>
<td>24</td>
<td>1</td>
<td>1</td>
<td>24</td>
<td>1</td>
</tr>
<tr>
<td>22</td>
<td>1</td>
<td>1</td>
<td>22</td>
<td>1</td>
</tr>
<tr>
<td>28</td>
<td>2</td>
<td>1</td>
<td>28</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>2</td>
<td>1</td>
<td>19</td>
<td>0</td>
</tr>
<tr>
<td>23</td>
<td>2</td>
<td>1</td>
<td>23</td>
<td>0</td>
</tr>
</tbody>
</table>

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 $X_1 \times X_2 \times A \times B \times C( D \ E)$. 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, $B \times A( E \ D)$ might be renamed $A \times B( D \ E)$ 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 $B \times A( C \ D)$, which is renamed $A \times B( C \ D)$, is as follows:

```
A_1 B_1 C_1 D_1 \rightarrow A_1 B_2 C_1 D_1 \rightarrow A_2 B_1 C_1 D_1 \rightarrow A_2 B_2 C_1 D_1 \rightarrow 
A_1 B_1 C_2 D_1 \rightarrow A_1 B_2 C_2 D_1 \rightarrow A_2 B_1 C_2 D_1 \rightarrow A_2 B_2 C_2 D_1 \rightarrow 
A_1 B_1 C_2 D_2 \rightarrow A_1 B_2 C_2 D_2 \rightarrow A_2 B_1 C_2 D_2 \rightarrow A_2 B_2 C_2 D_2 
```

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.
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

<table>
<thead>
<tr>
<th>Effect Coding</th>
<th>Design Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A)</td>
<td>A1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>-1</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Effect Coding</th>
<th>Design Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A)</td>
<td>A1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>-1</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Ordinal Coding</th>
<th>Design Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A)</td>
<td>A2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
</tr>
</tbody>
</table>

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, \ldots according to their sort order. The design matrix columns for \( A \) are as follows.

<table>
<thead>
<tr>
<th>Polynomial Coding</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Design Matrix</strong></td>
</tr>
<tr>
<td>( A )</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>7</td>
</tr>
</tbody>
</table>

**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.

<table>
<thead>
<tr>
<th>Orthogonal Effect Coding</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Design Matrix</strong></td>
</tr>
<tr>
<td>( A )</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>7</td>
</tr>
</tbody>
</table>

**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.

<table>
<thead>
<tr>
<th>Orthogonal Ordinal Coding</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Design Matrix</strong></td>
</tr>
<tr>
<td>( A )</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>7</td>
</tr>
</tbody>
</table>

**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

<table>
<thead>
<tr>
<th>Design Matrix</th>
<th>AOPOLY1</th>
<th>AOPOLY2</th>
<th>AOPOLY5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>–1.15311</td>
<td>0.90712</td>
<td>–0.92058</td>
</tr>
<tr>
<td>2</td>
<td>–0.73380</td>
<td>–0.54041</td>
<td>1.47292</td>
</tr>
<tr>
<td>5</td>
<td>0.52414</td>
<td>–1.37034</td>
<td>–0.92058</td>
</tr>
<tr>
<td>7</td>
<td>1.36277</td>
<td>1.00363</td>
<td>0.36823</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Design Matrix</th>
<th>AOREF1</th>
<th>AOREF2</th>
<th>AOREF3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.73205</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>–0.57735</td>
<td>1.63299</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>–0.57735</td>
<td>–0.81650</td>
<td>1.41421</td>
</tr>
<tr>
<td>7</td>
<td>–0.57735</td>
<td>–0.81650</td>
<td>–1.41421</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Design Matrix</th>
<th>A(B=1)</th>
<th>A(B=2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>B A</td>
<td>A1 A2</td>
<td>A1 A2</td>
</tr>
<tr>
<td>1 1</td>
<td>1 0</td>
<td>0 0</td>
</tr>
<tr>
<td>1 2</td>
<td>0 1</td>
<td>0 0</td>
</tr>
<tr>
<td>1 3</td>
<td>–1 –1</td>
<td>0 0</td>
</tr>
<tr>
<td>2 1</td>
<td>0 0</td>
<td>1 0</td>
</tr>
<tr>
<td>2 2</td>
<td>0 0</td>
<td>0 1</td>
</tr>
</tbody>
</table>

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.
Chapter 2: Shared Concepts

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, 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. These criteria fall into two groups—information criteria and criteria that are based on out-of-sample prediction performance.

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);
```

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.

You can use the SELECT= option together with the CHOOSE= and STOP= options. If you specify only the SELECT= criterion, then this criterion is also used as the stopping criterion. In the previous example, where
only the selection criterion is specified, 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 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:

\[
\text{selection method=forward stophorizon=1;}
\]

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:

\[
\text{selection=forward(select=SL);}
\]

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:

\[
\text{selection=forward(select=SL stop=validation) stophorizon=1;}
\]

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:

\[
\text{selection=forward(select=AIC);}
\]

The following statement adds effects that at each step produce the largest value of the adjusted \( R^2 \) statistic and stops at the step where the significance level that corresponds to the addition of this effect is greater than 0.2:

\[
\text{selection=forward(select=ADJRSQ stop=SL SLE=0.2);}
\]

**Backward Elimination**

This section applies to the following procedures: GENSELECT, LOGSELECT, 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 56.
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 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: 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 56.
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:
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 following procedures: REGSELECT.

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 $R^2$ square, this method is the same as the maximum $R^2$-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 following procedures: REGSELECT.

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 \texttt{CHOOSE=} option to specify a criterion for choosing among the models at each step. You can also use the \texttt{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 \texttt{LSCOEFFS} suboption of \texttt{METHOD=LAR}.

**LASSO Selection**

This section applies to the following procedures: \texttt{GENSELECT}, \texttt{LOGSELECT}, and \texttt{REGSELECT}.

\texttt{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 $\hat{\beta} = (\hat{\beta}_1, \hat{\beta}_2, \ldots, \hat{\beta}_m)$ are the solution to the following constrained optimization problem:

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

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 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 56. 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 following procedures: REGSELECT.

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 \( \beta = (\beta_1, \beta_2, \ldots, \beta_m) \) are the solution to the following constrained optimization problem:

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

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.

### Group LASSO Selection

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

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

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

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 $\beta_{G_j}$ 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

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

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

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

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:

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

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 $\lambda$. Nesterov’s method is known to have an optimal convergence rate for first-order black-box optimization. Because the optimal $\lambda$ is usually unknown, a series of regularization parameters $\rho, \rho^2, \rho^3, \ldots$ is used, where $\rho$ is a positive value less than 1. You can specify $\rho$ by using the LASSORHO= option in the procedure; the default value is $\rho = 0.8$. In the $i$th step of group LASSO selection, the value that is used for $\lambda$ is $\rho^i$.

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 32.

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 $\phi$ 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.

Informative Missingness

This section applies to the following procedures: GENSELECT, LOGSELECT, 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 $\text{AgeCHDdiag}$. 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:

$$y = \beta_0 + \beta_1 x^* + \beta_2 x_{\text{miss}} + \epsilon_1$$

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 $x_{\text{miss}}$ is a dummy variable whose value is 1 when $x$ is missing, and 0 otherwise:

$$x_{\text{miss}} = \begin{cases} 1 & \text{if } x \text{ is missing} \\ 0 & \text{otherwise} \end{cases}$$

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 $x_{\text{miss}}$ 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 $\beta_2$ 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: GENSELECT, LOGSELECT, 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:

```plaintext
proc logselect data=mycas.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.

```plaintext
proc logselect data=mycas.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

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

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: GENSELECT, LOGSELECT, and NLMOD.

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
Choosing an Optimization Algorithm

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 2.20 shows which derivatives are required for each optimization technique.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>First-Order</th>
<th>Second-Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUREG</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>NEWRAP</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>NRRIDG</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>QUANEW</td>
<td>x</td>
<td>-</td>
</tr>
<tr>
<td>DBLDOG</td>
<td>x</td>
<td>-</td>
</tr>
<tr>
<td>CONGRA</td>
<td>x</td>
<td>-</td>
</tr>
<tr>
<td>LEVMAR</td>
<td>x</td>
<td>-</td>
</tr>
<tr>
<td>NMSIMP</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

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 method CONGRA is 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 algorithm requires more iterations than QUANEW or DBLDOG, but each iteration can be much faster. Because CONGRA requires only a factor of \( p \) double-word memory, many large applications can be solved only by CONGRA.

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 \( \text{ABSGCONV} < 1 \times 10^{-5} \), \( \text{FCONV} < 2 \times \epsilon \), or \( \text{GCONV} < 1 \times 10^{-8} \).

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 \text{TECHNIQUE=}QUANEW option, 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 2.20.

_Trust Region Optimization (TRUREG)_

The trust region method uses the gradient $g(\psi^{(k)})$ and the Hessian matrix $H(\psi^{(k)})$; thus, it requires that the objective function $f(\psi)$ 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 $\Delta$ 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 $g(\psi^{(k)})$ and the Hessian matrix $H(\psi^{(k)})$; 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 $g(\psi^{(k)})$ and the Hessian matrix $H(\psi^{(k)})$; 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)_

The dual quasi-Newton method uses the gradient $g(\psi^{(k)})$, 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 $\alpha$ 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 $s^{(k)}$ as the linear combination of the steepest descent or ascent search direction $s_1^{(k)}$ and a quasi-Newton search direction $s_2^{(k)}$:

$$s^{(k)} = \alpha_1 s_1^{(k)} + \alpha_2 s_2^{(k)}$$

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 $\alpha$ 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.

**References**


Part II

Statistics
Chapter 3
The GENSELECT Procedure

Contents

<table>
<thead>
<tr>
<th>Overview: GENSELECT Procedure</th>
<th>78</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROC GENSELECT Features</td>
<td>78</td>
</tr>
<tr>
<td>PROC GENSELECT Compared with Other SAS Procedures</td>
<td>79</td>
</tr>
<tr>
<td>Using CAS Sessions and CAS Engine Librefs</td>
<td>80</td>
</tr>
<tr>
<td>Getting Started: GENSELECT Procedure</td>
<td>80</td>
</tr>
<tr>
<td>Poisson Regression for Count Data</td>
<td>80</td>
</tr>
<tr>
<td>Syntax: GENSELECT Procedure</td>
<td>85</td>
</tr>
<tr>
<td>PROC GENSELECT Statement</td>
<td>86</td>
</tr>
<tr>
<td>BY Statement</td>
<td>88</td>
</tr>
<tr>
<td>CLASS Statement</td>
<td>88</td>
</tr>
<tr>
<td>CODE Statement</td>
<td>89</td>
</tr>
<tr>
<td>DISPLAY Statement</td>
<td>89</td>
</tr>
<tr>
<td>DISPLAYOUT Statement</td>
<td>90</td>
</tr>
<tr>
<td>EFFECT Statement</td>
<td>91</td>
</tr>
<tr>
<td>FREQ Statement</td>
<td>92</td>
</tr>
<tr>
<td>MODEL Statement</td>
<td>92</td>
</tr>
<tr>
<td>OUTPUT Statement</td>
<td>98</td>
</tr>
<tr>
<td>PARTITION Statement</td>
<td>102</td>
</tr>
<tr>
<td>SELECTION Statement</td>
<td>103</td>
</tr>
<tr>
<td>WEIGHT Statement</td>
<td>105</td>
</tr>
<tr>
<td>Details: GENSELECT Procedure</td>
<td>105</td>
</tr>
<tr>
<td>Missing Values</td>
<td>105</td>
</tr>
<tr>
<td>Exponential Family Distributions</td>
<td>105</td>
</tr>
<tr>
<td>Response Distributions</td>
<td>106</td>
</tr>
<tr>
<td>Response Probability Distribution Functions</td>
<td>108</td>
</tr>
<tr>
<td>Log-Likelihood Functions</td>
<td>112</td>
</tr>
<tr>
<td>Existence of Maximum Likelihood Estimates</td>
<td>116</td>
</tr>
<tr>
<td>The LASSO Method of Model Selection</td>
<td>117</td>
</tr>
<tr>
<td>Model Fit and Assessment Statistics</td>
<td>118</td>
</tr>
<tr>
<td>Predicted Values and Regression Diagnostics</td>
<td>119</td>
</tr>
<tr>
<td>Joint Tests and Type 3 Tests</td>
<td>122</td>
</tr>
<tr>
<td>Multithreading</td>
<td>122</td>
</tr>
<tr>
<td>Optimization Algorithms</td>
<td>123</td>
</tr>
<tr>
<td>Displayed Output</td>
<td>123</td>
</tr>
<tr>
<td>ODS Table Names</td>
<td>127</td>
</tr>
<tr>
<td>Examples: GENSELECT Procedure</td>
<td>128</td>
</tr>
</tbody>
</table>
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 and modern LASSO methods. The procedure 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, and Tweedie distributions. In addition, PROC GENSELECT fits multinomial models for ordinal responses, and it fits models for the beta, generalized Poisson, and negative binomial distributions.

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 response-variable options for binary and multinomial data as in the LOGISTIC procedure
- 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

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 68 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.

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.

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:

```sas
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:

```sas
cas mysess;
libname mycas cas sessref=mysess;
```

The CAS statement creates the CAS session named `mysess`, and the LIBNAME statement creates the `mycas` 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:

```sas
cas mysess terminate;
```

For more information about the CAS and LIBNAME statements, see the section “Introduction to Shared Concepts” on page 8 in Chapter 2, “Shared Concepts.”

---

### Getting Started: GENSELECT Procedure

---

### Poisson Regression for Count Data

This example illustrates how you can use PROC GENSELECT to perform Poisson regression for count data. 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 8 and “Loading a SAS Data Set onto a CAS Server” on page 9 in Chapter 2, “Shared Concepts.” 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, in your CAS session:

```plaintext
data mycas.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 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 1 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 CAS engine libref is named mycas, but you can substitute any appropriately
defined CAS engine libref.

The following statements fit a log-linked Poisson model to these data by using classification effects for the
variables C1–C5:

```plaintext
proc genselect data=mycas.getStarted;
  class C1-C5;
  model Y = C1-C5 / Distribution=Poisson Link=Log;
run;
```

The output from this analysis is presented in Figure 3.1 through Figure 3.7.

**Figure 3.1** 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 3.1 Model Information**

<table>
<thead>
<tr>
<th>Model Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Source: GETSTARTED</td>
</tr>
<tr>
<td>Response Variable: Y</td>
</tr>
<tr>
<td>Distribution: Poisson</td>
</tr>
<tr>
<td>Link Function: Log</td>
</tr>
<tr>
<td>Optimization Technique:</td>
</tr>
<tr>
<td>Newton-Raphson with Ridging</td>
</tr>
</tbody>
</table>

Each of the CLASS variables C1–C5 has four unique formatted levels, which are displayed in the “Class
Level Information” table in Figure 3.2.

**Figure 3.2 Class Level Information**

<table>
<thead>
<tr>
<th>Class Level Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td>C1</td>
</tr>
<tr>
<td>C2</td>
</tr>
<tr>
<td>C3</td>
</tr>
<tr>
<td>C4</td>
</tr>
<tr>
<td>C5</td>
</tr>
</tbody>
</table>

**Figure 3.3** displays the “Number of Observations” table. All 100 observations in the data set are used in the
analysis.

**Figure 3.3 Number of Observations**

<table>
<thead>
<tr>
<th>Number of Observations Read</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Observations Used</td>
<td>100</td>
</tr>
</tbody>
</table>
Figure 3.4 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 $X$: 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 3.4 Dimensions in Poisson Regression**

<table>
<thead>
<tr>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Columns in Design</td>
</tr>
<tr>
<td>Number of Effects</td>
</tr>
<tr>
<td>Max Effect Columns</td>
</tr>
<tr>
<td>Rank of Design</td>
</tr>
<tr>
<td>Parameters in Optimization</td>
</tr>
</tbody>
</table>

Figure 3.5 displays the final convergence status of the Newton-Raphson algorithm. The FCONV= convergence criterion is satisfied.

**Figure 3.5 Convergence Status**

Convergence criterion (FCONV=1E-7) satisfied.

The “Fit Statistics” table is shown in Figure 3.6. 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 3.6 Fit Statistics**

<table>
<thead>
<tr>
<th>Fit Statistics</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2 Log Likelihood</td>
<td>290.16169</td>
</tr>
<tr>
<td>AIC (smaller is better)</td>
<td>322.16169</td>
</tr>
<tr>
<td>AICC (smaller is better)</td>
<td>328.71590</td>
</tr>
<tr>
<td>SBC (smaller is better)</td>
<td>363.84441</td>
</tr>
</tbody>
</table>

The “Parameter Estimates” table in Figure 3.7 shows that many parameters have fairly large $p$-values, indicating that one or more of the model effects might not be necessary.
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 > ;
   OUTPUT OUT=CAS-libref.data-table < keyword < =name > > . . . < keyword < =name > > < options > ;
   PARTITION < partition-options > ;
   SELECTION < METHOD=method < (method-options) > > < options > ;
   WEIGHT variable ;
```

The PROC GENSELECT statement and at least one MODEL statement are required. The CLASS statement can appear multiple times. If a CLASS statement is specified, it must precede the MODEL statements.
**PROC GENSELECT Statement**

```
PROC GENSELECT < options >;
```

The PROC GENSELECT statement invokes the procedure. Table 3.1 summarizes the available options in the PROC GENSELECT statement by function. They are then described fully in alphabetical order.

### Table 3.1  PROC GENSELECT Statement Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Basic Options</strong></td>
<td></td>
</tr>
<tr>
<td>ALPHA=</td>
<td>Specifies a global significance level</td>
</tr>
<tr>
<td>DATA=</td>
<td>Specifies the CAS input data table</td>
</tr>
<tr>
<td>MAXOPTBATCH=</td>
<td>Specifies the maximum number of observations to be computed in each batch</td>
</tr>
<tr>
<td><strong>Options Related to Output</strong></td>
<td></td>
</tr>
<tr>
<td>CORRB</td>
<td>Displays the “Parameter Estimates Correlation Matrix” table</td>
</tr>
<tr>
<td>COVB</td>
<td>Displays the “Parameter Estimates Covariance Matrix” table</td>
</tr>
<tr>
<td>ITHIST</td>
<td>Displays the “Iteration History” table</td>
</tr>
<tr>
<td>NOCHECK</td>
<td>Disables checking for infinite parameters</td>
</tr>
<tr>
<td>NOCLPRINT</td>
<td>Limits or suppresses the display of classification variable levels</td>
</tr>
<tr>
<td>NOSTDERR</td>
<td>Suppresses computation of the covariance matrix and standard errors</td>
</tr>
<tr>
<td>PARTFIT</td>
<td>Displays the fit statistics that are produced when your data are partitioned</td>
</tr>
<tr>
<td><strong>Options Related to Optimization</strong></td>
<td></td>
</tr>
<tr>
<td>ABSCONV=</td>
<td>Tunes the absolute function convergence criterion</td>
</tr>
<tr>
<td>ABSFCONV=</td>
<td>Tunes the absolute function difference convergence criterion</td>
</tr>
<tr>
<td>ABSSGCONV=</td>
<td>Tunes the absolute gradient convergence criterion</td>
</tr>
<tr>
<td>ABSXCONV=</td>
<td>Tunes the absolute parameter convergence criterion</td>
</tr>
<tr>
<td>FCONV=</td>
<td>Tunes the relative function difference convergence criterion</td>
</tr>
<tr>
<td>FCONV2=</td>
<td>Tunes the second relative function difference convergence criterion</td>
</tr>
<tr>
<td>GCONV=</td>
<td>Tunes the relative gradient convergence criterion</td>
</tr>
<tr>
<td>GCONV2=</td>
<td>Tunes the second relative gradient convergence criterion</td>
</tr>
<tr>
<td>XCONV=</td>
<td>Tunes the relative gradient convergence criterion</td>
</tr>
<tr>
<td>MAXFUNC=</td>
<td>Specifies the maximum number of function evaluations in any optimization</td>
</tr>
<tr>
<td>MAXITER=</td>
<td>Chooses the maximum number of iterations in any optimization</td>
</tr>
<tr>
<td>MAXTIME=</td>
<td>Specifies the upper limit of CPU time (in seconds) for any optimization</td>
</tr>
<tr>
<td>MINITER=</td>
<td>Specifies the minimum number of iterations in any optimization</td>
</tr>
<tr>
<td>NORMALIZE=</td>
<td>Specifies whether the objective function is normalized during optimization</td>
</tr>
<tr>
<td>TECHNIQUE=</td>
<td>Selects the optimization technique</td>
</tr>
<tr>
<td><strong>LASSO Options</strong></td>
<td></td>
</tr>
<tr>
<td>LASSORHO=</td>
<td>Specifies the base regularization parameter for the LASSO method</td>
</tr>
<tr>
<td>LASSOSTEPS=</td>
<td>Specifies the maximum number of steps for the LASSO method</td>
</tr>
<tr>
<td>LASSOTOL=</td>
<td>Specifies the convergence criterion for the LASSO method</td>
</tr>
</tbody>
</table>

The optimization options are fully described in the section “Optimization Options” on page 38 in Chapter 2, “Shared Concepts.” The following list describes the other options available in the PROC GENSELECT statement:
**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.

**MAXOPTBATCH=number**
specifies the maximum number of observations to be included in a batch. 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 MAXOPTBATCH value decreases memory usage but might lead to longer computation times, whereas a larger MAXOPTBATCH value might lead to shorter computation times but increases memory usage. By default, MAXOPTBATCH=256.

**CORRB**
creates the “Parameter Estimates Correlation Matrix” table. The correlation matrix is computed by normalizing the covariance matrix \( \Sigma \). That is, if \( \sigma_{ij} \) is an element of \( \Sigma \), then the corresponding element of the correlation matrix is \( \rho_{ij} = \frac{\sigma_{ij}}{\sigma_i \sigma_j} \), 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 of the matrix of second derivatives of the log-likelihood function with respect to the model parameters (the Hessian matrix).

**DATA=cas-libref.data-table**
names the input data table for PROC GENSELECT to use. The default is the most recently created data table. cas-libref specifies the CAS library where the input data table resides, and data-table specifies the name of the input data table.

**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.

**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 116.
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 116.

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).

**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 *SAS Language Reference: Concepts*.

**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 3.2 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 10 in Chapter 2, “Shared Concepts.”

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DESCENDING</td>
<td>Reverses the sort order</td>
</tr>
<tr>
<td>MISSING</td>
<td>Treats missing values as valid levels</td>
</tr>
<tr>
<td>ORDER=</td>
<td>Specifies the sort order for the levels</td>
</tr>
<tr>
<td>PARAM=</td>
<td>Specifies the parameterization of the variable</td>
</tr>
<tr>
<td>REF=</td>
<td>Specifies the reference level of the variable</td>
</tr>
<tr>
<td>SPLIT</td>
<td>Allows design columns for a variable to enter or leave the model independently</td>
</tr>
</tbody>
</table>
**CODE Statement**

```sas
CODE <options> ;
```

The CODE statement writes SAS DATA step code for computing predicted values of the fitted model either to a file or to a catalog entry. This code can then be included in a DATA step to score new data.

Table 3.3 summarizes the *options* available in the CODE statement.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMMENT</td>
<td>Adds comments to the generated code</td>
</tr>
<tr>
<td>FILE=</td>
<td>Names the file where the generated code is saved</td>
</tr>
<tr>
<td>FORMATWIDTH=</td>
<td>Specifies the numeric format width for the regression coefficients</td>
</tr>
<tr>
<td>INDENTSIZE=</td>
<td>Specifies the number of spaces to indent the generated code</td>
</tr>
<tr>
<td>LABELID=</td>
<td>Specifies a number used to construct names and labels</td>
</tr>
<tr>
<td>LINESIZE=</td>
<td>Specifies the line size for the generated code</td>
</tr>
<tr>
<td>NOTRIM</td>
<td>Compares formatted values, including blank padding</td>
</tr>
<tr>
<td>PCATALL</td>
<td>Generates probabilities for all levels of categorical response variables</td>
</tr>
</tbody>
</table>

For more information about the syntax of the CODE statement, see the section “CODE Statement” on page 14 in Chapter 2, “Shared Concepts.”

**DISPLAY Statement**

```sas
DISPLAY <table-list> < /options> ;
```

The DISPLAY statement enables you to specify a list of ODS 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, all ODS tables are sent to the client and then the client creates a subset. If both DISPLAY and ODS statements are used together, the DISPLAY statement takes precedence over the ODS statements. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

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

**CASESENSITIVE**

performs a case-sensitive comparison of table names in the *table-list* to ODS 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 ODS tables except those specified in the *table-list*. 

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

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

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

A path is a table name that is prefixed with dot-separated grouping information. For example, a SelectionSummary table that is produced by a procedure 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 ODS 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 starts with a “/” or a “!”. 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.

DISPLAYOUT Statement

DISPLAYOUT table-spec-list < / options > ;

The DISPLAYOUT statement enables you to create CAS 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 CAS output tables to create. Each entry in the list has either a key or a key=value format:

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

key specifies key as the ODS table name and also as the CAS output table name.

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 (/):

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

REPEATED replicates the CAS 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 48 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 3.4 summarizes the *options* available in the EFFECT statement.

### Table 3.4  EFFECT Statement Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Collection Effects Options</strong></td>
<td></td>
</tr>
<tr>
<td>DETAILS</td>
<td>Displays the constituents of the collection effect</td>
</tr>
<tr>
<td><strong>Multimember Effects Options</strong></td>
<td></td>
</tr>
<tr>
<td>DETAILS</td>
<td>Displays the levels of the multimember effect</td>
</tr>
<tr>
<td>NOEFFECT</td>
<td>Specifies that observations whose levels are all missing for the multimember variables should have 0 values in the corresponding design matrix columns</td>
</tr>
<tr>
<td>STDIZE</td>
<td>Standardizes the design matrix entries so that each observation has a sum of 1</td>
</tr>
<tr>
<td>WEIGHT=</td>
<td>Specifies the weight variable for the contributions of each classification effect</td>
</tr>
<tr>
<td><strong>Polynomial Effects Options</strong></td>
<td></td>
</tr>
<tr>
<td>DEGREE=</td>
<td>Specifies the degree of the polynomial</td>
</tr>
<tr>
<td>DETAILS</td>
<td>Displays details of the specified polynomial</td>
</tr>
<tr>
<td>MDEGREE=</td>
<td>Specifies the maximum degree of any variable in a term of the polynomial</td>
</tr>
<tr>
<td>NOSEPARATE</td>
<td>Treats the polynomial as a single effect with multiple degrees of freedom</td>
</tr>
<tr>
<td>STANDARDIZE=</td>
<td>Specifies centering and scaling suboptions for the variables that define the polynomial</td>
</tr>
</tbody>
</table>
Table 3.4  continued

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Spline Effects Options</strong></td>
<td></td>
</tr>
<tr>
<td>BASIS=</td>
<td>Specifies the type of basis (B-spline basis or truncated power function basis) for the spline effect</td>
</tr>
<tr>
<td>DATABOUNDARY</td>
<td>Uses the extremes of the data as boundary knots for a B-spline basis</td>
</tr>
<tr>
<td>DEGREE=</td>
<td>Specifies the degree of the spline effect</td>
</tr>
<tr>
<td>DETAILS</td>
<td>Displays the knots and locations for each spline basis function</td>
</tr>
<tr>
<td>KNOTMAX=</td>
<td>Requests equally spaced right-side boundary knots starting at the variables’ maximum and ending at the KNOTMAX= value</td>
</tr>
<tr>
<td>KNOTMETHOD=</td>
<td>Specifies how to construct the knots for the spline effect</td>
</tr>
<tr>
<td>KNOTMIN=</td>
<td>Requests equally spaced left-side boundary knots starting at the variables’ minimum value</td>
</tr>
<tr>
<td>NATURALCUBIC</td>
<td>Specifies a natural cubic spline basis for the spline effect</td>
</tr>
<tr>
<td>SEPARATE</td>
<td>Treats the spline basis for each variable as a separate effect when multiple variables are specified</td>
</tr>
<tr>
<td>SPLIT</td>
<td>Treats each design matrix column as a separate effect for selection methods</td>
</tr>
</tbody>
</table>

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 17 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 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 the FREQ statement is not specified, each observation is assigned a frequency of 1.

---

**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 \((\text{trials} - \text{events})\) 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 45 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 3.5 summarizes these options.

### Table 3.5 MODEL Statement Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DESCENDING</td>
<td>Reverses the response categories</td>
</tr>
<tr>
<td>EVENT=’category’</td>
<td>Specifies the event category</td>
</tr>
<tr>
<td>ORDER=</td>
<td>Specifies the sort order</td>
</tr>
<tr>
<td>REF=</td>
<td>Specifies the reference category</td>
</tr>
<tr>
<td>CLB</td>
<td>Requests confidence limits</td>
</tr>
<tr>
<td>INCLUDE=</td>
<td>Includes effects in all models for model selection</td>
</tr>
<tr>
<td>INFORMATIVE</td>
<td>Models missing values by using extra indicator variables</td>
</tr>
<tr>
<td>LINK=</td>
<td>Specifies the link function</td>
</tr>
<tr>
<td>NOINT</td>
<td>Suppresses the intercept</td>
</tr>
<tr>
<td>OFFSET=</td>
<td>Specifies the offset variable</td>
</tr>
<tr>
<td>PHI=</td>
<td>Specifies a fixed dispersion parameter</td>
</tr>
<tr>
<td>START=</td>
<td>Includes effects in the initial model for model selection</td>
</tr>
<tr>
<td>TYPE3</td>
<td>Displays the Type 3 or joint tests of effects</td>
</tr>
</tbody>
</table>

**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:
Chapter 3: The GENSELECT Procedure

'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.

```sas
proc genselect data=mycas.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. When ORDER=FORMATTED (the default) 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 following table shows how the ORDER= option is interpreted:

<table>
<thead>
<tr>
<th>ORDER=</th>
<th>Levels Sorted By</th>
</tr>
</thead>
<tbody>
<tr>
<td>FORMATTED</td>
<td>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.</td>
</tr>
<tr>
<td>FREQ</td>
<td>Descending frequency count (levels that have the most observations come first in the order)</td>
</tr>
<tr>
<td>INTERNAL</td>
<td>Unformatted value; the sort order is machine-dependent.</td>
</tr>
</tbody>
</table>

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 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. You can specify one of the following:

'category'
specifies the value (formatted, if a format is applied) of the reference category in quotation marks.
**MODEL Statement**

**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 (/):

- **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.

- **DISTRIBUTION=** *keyword*
  specifies the response distribution for the model. The *keywords* and the associated distributions are shown in Table 3.6. For information about default and commonly used link functions for each distribution function, see Table 3.8.

<table>
<thead>
<tr>
<th>Table 3.6 Built-In Distribution Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Keyword</strong></td>
</tr>
<tr>
<td>BETA</td>
</tr>
<tr>
<td>BINARY</td>
</tr>
<tr>
<td>BINOMIAL</td>
</tr>
<tr>
<td>EXPONENTIAL</td>
</tr>
<tr>
<td>GAMMA</td>
</tr>
<tr>
<td>GENPOISSON</td>
</tr>
<tr>
<td>GEOMETRIC</td>
</tr>
<tr>
<td>INVERSEGAUSSIAN</td>
</tr>
<tr>
<td>MULTINOMIAL</td>
</tr>
<tr>
<td>NEGATIVEBINOMIAL</td>
</tr>
<tr>
<td>NORMAL</td>
</tr>
<tr>
<td>POISSON</td>
</tr>
<tr>
<td>T&lt;ν&gt;</td>
</tr>
<tr>
<td>TWEEDIE&lt;(Tweedie-options)&gt;</td>
</tr>
<tr>
<td>WEIBULL</td>
</tr>
</tbody>
</table>

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.

P=value
specifies a value to be used as a fixed Tweedie power parameter.

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 (/).

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 65 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 3.7. Default and commonly used link functions for the available distributions are shown in Table 3.8.

Table 3.7  Built-In Link Functions

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Link Function</th>
<th>g(μ) = η =</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLOGLOG</td>
<td>Complementary log-log or cumulative complementary log-log</td>
<td>log(− log(1 − μ))</td>
</tr>
<tr>
<td>IDENTITY</td>
<td>Identity</td>
<td>μ</td>
</tr>
<tr>
<td>INVERSE</td>
<td>Reciprocal</td>
<td>1/μ</td>
</tr>
<tr>
<td>POWERMINUS2</td>
<td>Reciprocal square</td>
<td>1/μ²</td>
</tr>
<tr>
<td>POWER(p)</td>
<td>Power</td>
<td>μᵖ</td>
</tr>
<tr>
<td>LOG</td>
<td>Logarithm</td>
<td>log(μ)</td>
</tr>
<tr>
<td>LOGIT</td>
<td>Logit or cumulative logit</td>
<td>log(μ/(1 − μ))</td>
</tr>
<tr>
<td>LOGLOG</td>
<td>Log-log or cumulative log-log</td>
<td>− log(− log(μ))</td>
</tr>
<tr>
<td>PROBIT</td>
<td>Probit or cumulative probit</td>
<td>Φ⁻¹(μ)</td>
</tr>
</tbody>
</table>
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 3.8. For binary or multinomial distributions, only the link functions shown in Table 3.8 are available. For the other distributions, you can use any link function shown in Table 3.7 by specifying the LINK= option. Other commonly used link functions for each distribution are shown in Table 3.8.

<table>
<thead>
<tr>
<th>Value of the DISTRIBUTION= Option</th>
<th>Default Link Function</th>
<th>Other Commonly Used Link Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>BETA</td>
<td>Logit</td>
<td>Probit, complementary log-log, log-log</td>
</tr>
<tr>
<td>BINARY</td>
<td>Logit</td>
<td>Probit, complementary log-log, log-log</td>
</tr>
<tr>
<td>BINOMIAL</td>
<td>Logit</td>
<td>Probit, complementary log-log, log-log</td>
</tr>
<tr>
<td>EXPONENTIAL</td>
<td>Log</td>
<td>Probit, complementary log-log, log-log</td>
</tr>
<tr>
<td>GAMMA</td>
<td>Log</td>
<td>Probit, complementary log-log, log-log</td>
</tr>
<tr>
<td>GENPOISSON</td>
<td>GPOISSON</td>
<td>Log</td>
</tr>
<tr>
<td>GEOMETRIC</td>
<td>Log</td>
<td>Cumulative probit, cumulative complementary log-log, cumulative log-log</td>
</tr>
<tr>
<td>INVERSEGAUSSIAN</td>
<td>IG</td>
<td>Log</td>
</tr>
<tr>
<td>MULTINOMIAL</td>
<td>Cumulative logit</td>
<td>Cumulative probit, cumulative complementary log-log, cumulative log-log</td>
</tr>
<tr>
<td>NEGATIVEBINOMIAL</td>
<td>NB</td>
<td>Log</td>
</tr>
<tr>
<td>NORMAL</td>
<td>GAUSSIAN</td>
<td>Log</td>
</tr>
<tr>
<td>POISSON</td>
<td>Log</td>
<td>Log</td>
</tr>
<tr>
<td>T</td>
<td>Identity</td>
<td>Log</td>
</tr>
<tr>
<td>TWEEDIE</td>
<td>Log</td>
<td>Log</td>
</tr>
<tr>
<td>WEIBULL</td>
<td>Log</td>
<td>Log</td>
</tr>
</tbody>
</table>

**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.

**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.
\textbf{START} = \textit{n}  \\
\textbf{START} = \textit{single-effect}  \\
\textbf{START} = (\textit{effects}) begins the selection process from the designated initial model for the forward selection method. If you specify \textbf{START} = \textit{n}, then the starting model includes the first \textit{n} effects that are listed in the \textbf{MODEL} statement. If you specify \textbf{START} = \textit{single-effect} or \textbf{START} = (\textit{effects}), then the starting model includes those specified effects. The effects that you specify in the \textbf{START} = option must be explanatory effects that are specified in the \textbf{MODEL} statement before the slash (/). This option is not available when you specify \textbf{METHOD} = BACKWARD in the \textbf{SELECTION} statement.

\textbf{TYPE3} requests that Wald statistics for Type 3 contrasts be computed for each effect that is specified in the \textbf{MODEL} statement. For more information, see the section “Joint Tests and Type 3 Tests” on page 122.

\textbf{OUTPUT Statement}

\begin{verbatim}
OUTPUT OUT=\textit{CAS-libref.data-table}  \\
\quad <\textit{ALL}> <\textit{ALPHA}=\textit{number}> <\textit{COPYVARS}=(\textit{variables})>  \\
\quad <\textit{keyword} <\textit{name}> >...<\textit{keyword} <\textit{name}> > ;
\end{verbatim}

The \textbf{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 \textbf{COPYVAR} = option.

If the response variable has more than two categories, you can request the “Statistic Options” listed in Table 3.9; 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 \textbf{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.

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:

\begin{verbatim}
OUT=\textit{CAS-libref.data-table}
\end{verbatim}

names the output data table for PROC GENSELECT to use. You must specify this option before any other options. \textit{CAS-libref.data-table} is a two-level name, where

\textit{CAS-libref} refers to a collection of information that is defined in the \textbf{LIBNAME} statement and includes the \textit{caslib}, 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 \textbf{LIBNAME} statement. For more information about \textit{CAS-libref}, see the section “Using CAS Sessions and CAS Engine Librefs” on page 80.
**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 the **OBSCAT** option and the ordinal response variable has *J* levels, then *J*–1 records are output for every observation in the input data that corresponds to the *J*–1 lower-ordered 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.

Table 3.9 summarizes the *keywords* available in the **OUTPUT** statement.

**Table 3.9** **OUTPUT** Statement Keywords

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
<th>Default Names</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Statistic Options</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INDIVIDUAL</td>
<td>Specifies the individual predicted probabilities</td>
<td><em>IPRED</em></td>
</tr>
<tr>
<td>LCL</td>
<td>Specifies the lower confidence limit for the linear predictor</td>
<td><em>LCL</em></td>
</tr>
<tr>
<td>LCLM</td>
<td>Specifies the lower confidence limit for the event probability</td>
<td><em>LCLM</em></td>
</tr>
<tr>
<td>PREDICTED</td>
<td>Specifies the predicted probabilities</td>
<td><em>PRED</em></td>
</tr>
<tr>
<td>STDXBETA</td>
<td>Specifies the standard error estimate of the linear predictor</td>
<td><em>STDXBETA</em></td>
</tr>
<tr>
<td>UCL</td>
<td>Specifies the upper confidence limit for the linear predictor</td>
<td><em>UCL</em></td>
</tr>
<tr>
<td>UCLM</td>
<td>Specifies the upper confidence limit for the event probability</td>
<td><em>UCLM</em></td>
</tr>
<tr>
<td>XBETA</td>
<td>Specifies the linear predictor</td>
<td><em>XBETA</em></td>
</tr>
</tbody>
</table>
Table 3.9 continued

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
<th>Default Names</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Diagnostic Options</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CBAR</td>
<td>Specifies the confidence interval displacement</td>
<td><em>CBAR</em></td>
</tr>
<tr>
<td>DIFCHISQ</td>
<td>Specifies the deletion chi-square goodness-of-fit change</td>
<td><em>DIFCHISQUARE</em></td>
</tr>
<tr>
<td>DIFDEV</td>
<td>Specifies the deletion deviance change</td>
<td><em>DIFDEVIANCE</em></td>
</tr>
<tr>
<td>H</td>
<td>Specifies the leverage</td>
<td><em>HATDIAG</em></td>
</tr>
<tr>
<td>RESCHI</td>
<td>Specifies the Pearson chi-square residual</td>
<td><em>RESCHI</em></td>
</tr>
<tr>
<td>RESDEV</td>
<td>Specifies the deviance residual</td>
<td><em>RESDEV</em></td>
</tr>
<tr>
<td>RESLIK</td>
<td>Specifies the likelihood residual</td>
<td><em>RESLIK</em></td>
</tr>
<tr>
<td>RESRAW</td>
<td>Specifies the raw residual</td>
<td><em>RESRAW</em></td>
</tr>
<tr>
<td>RESWORK</td>
<td>Specifies the working residual</td>
<td><em>RESWORK</em></td>
</tr>
<tr>
<td>STDRESCHI</td>
<td>Specifies the standardized Pearson chi-square residual</td>
<td><em>STDRESCHI</em></td>
</tr>
<tr>
<td>STDRESDEV</td>
<td>Specifies the standardized deviance residual</td>
<td><em>STDRESDEV</em></td>
</tr>
</tbody>
</table>

The following list describes these *keywords*. For more information, see the section “Predicted Values and Regression Diagnostics” on page 119.

**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_.

**H**  
specifies the diagonal element of the hat matrix (leverage) for detecting extreme points in the design space. The default name is _HATDIAG_.

**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 $P(Y = 1)$, $P(Y = 2)$, and $P(Y = 3)$. The default name is _IPRED_.

**IP**
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.

PREDICTED
PRED
PROB
P
  specifies the predicted values (or predicted probabilities of events) for the response variable 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 ≤ 1) and Pr(Y ≤ 2), but by default the last level, Pr(Y ≤ 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_.

RESLIK
  specifies the likelihood residual for identifying poorly fitted observations. The default name is _RESLIK_.

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 3.10 shows how this variable is interpreted for each observation.
Chapter 3: The GENSELECT Procedure

Table 3.10  Role Interpretation

<table>
<thead>
<tr>
<th>Value</th>
<th>Observation Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not used</td>
</tr>
<tr>
<td>1</td>
<td>Training</td>
</tr>
<tr>
<td>2</td>
<td>Validation</td>
</tr>
<tr>
<td>3</td>
<td>Testing</td>
</tr>
</tbody>
</table>

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_.

**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_.

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 66 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 of the observations 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 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 118.

**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 32 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.
- **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 63 in Chapter 2, “Shared Concepts.”
- **NONE** results in no model selection. This method fits the full model.

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:
Chapter 3: The GENSELECT Procedure

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 118. 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.

For forward selection in models other than the binary or binomial where you also specify the NOINT option, you must use significance level (SL) as the SELECT= criterion.

If you specify METHOD=LASSO and do not specify the CHOOSE= 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 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. 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.

The GENSELECT procedure does not support the ORDERSELECT option in this release.
**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, all observations that are used in the analysis are assigned a weight of 1.

---

**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 65 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 $\phi$, 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:

$$E(Y) = b'(\theta)$$
$$\text{Var}(Y) = b''(\theta)\phi$$
In generalized linear models, the mean $\mu$ of the response distribution is related to linear regression parameters through a link function,

$$g(\mu_i) = x_i^T \beta$$

for the $i$th observation, where $x_i$ is a fixed known vector of explanatory variables and $\beta$ is a vector of regression parameters. The GENSELECT procedure parameterizes models in terms of the regression parameters $\beta$ and either the dispersion parameter $\phi$ or a parameter that is related to $\phi$, depending on the model. For exponential family models, the distribution variance is $\text{Var}(Y) = \phi V(\mu)$, where $V(\mu)$ is a variance function that depends only on $\mu$.

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
- multinomial for ordinal 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 108. The expressions for the log-likelihood functions of these distributions are presented in the section “Log-Likelihood Functions” on page 112.

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 $i$th 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:

```plaintext
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.
Response Probability Distribution Functions

Beta Distribution

\[ f(y) = \frac{\Gamma(\phi)}{\Gamma(\mu\phi)\Gamma((1-\mu)\phi)} y^{\mu\phi-1}(1-y)^{(1-\mu)\phi-1} \quad \text{for } 0 < y < 1 \]

\[ E(Y) = \mu \]

\[ \text{Var}(Y) = \frac{\mu(1-\mu)}{1 + \phi}, \quad \phi > 0 \]

This parameterization of the beta distribution comes from Ferrari and Cribari-Neto (2004).

Binary Distribution

\[ f(y) = \begin{cases} 
  p & \text{for } y = 1 \\
  1 - p & \text{for } y = 0 
\end{cases} \]

\[ E(Y) = p \]

\[ \text{Var}(Y) = p(1-p) \]

Binomial Distribution

\[ f(y) = \binom{n}{r} \mu^r (1-\mu)^{n-r} \quad \text{for } y = \frac{r}{n}, \quad r = 0, 1, 2, \ldots, n \]

\[ E(Y) = \mu \]

\[ \text{Var}(Y) = \frac{\mu(1-\mu)}{n} \]

Exponential Distribution

\[ f(y) = \frac{1}{\mu} \exp\left(-\frac{y}{\mu}\right) \quad \text{for } 0 < y < \infty \]

\[ E(Y) = \mu \]

\[ \text{Var}(Y) = \mu^2 \]

The exponential distribution is a special case of the gamma where \( \nu = 1 \).
Response Probability Distribution Functions

Gamma Distribution

\[ f(y) = \frac{1}{\Gamma(v)y} \left( \frac{y}{\mu} \right)^v \exp \left( -\frac{y}{\mu} \right) \quad \text{for } 0 < y < \infty \]

\[ \phi = \frac{1}{v} \]

\[ E(Y) = \mu \]

\[ \text{Var}(Y) = \frac{\mu^2}{v}, \quad v > 0 \]

For the gamma distribution, \( v = \frac{1}{\phi} \) is the estimated dispersion parameter that is displayed in the output. The parameter \( v \) is sometimes called the gamma index parameter.

Generalized Poisson Distribution

\[ \xi = 1 - \exp(-\phi), \quad \phi \geq 0 \]

\[ \mu^* = \mu - \xi(\mu - y) \]

\[ f(y) = (\mu^*)^{y-1} \frac{((1 - \xi)\mu)}{\Gamma(y + 1)} \exp(-\mu^*) \quad \text{for } y = 0, 1, 2, \ldots \]

\[ E(Y) = \mu \]

\[ \text{Var}(Y) = \frac{\mu}{(1 - \xi)^2} \]

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, \( \phi \) is the estimated dispersion parameter that is displayed in the output.

Geometric Distribution

\[ f(y) = \frac{\mu^y}{(1 + \mu)^{y+1}} \quad \text{for } y = 0, 1, 2, \ldots \]

\[ E(Y) = \mu \]

\[ \text{Var}(Y) = \mu + \mu^2 \]

The geometric distribution is a special case of the negative binomial where \( \phi = 1 \).
Inverse Gaussian Distribution

\[ f(y) = \frac{1}{\sqrt{2\pi y^2}} \exp \left( -\frac{1}{2y} \left( \frac{y - \mu}{\mu \sigma} \right)^2 \right) \text{ for } 0 < y < \infty \]

\[ \phi = \sigma^2 \]

\[ \text{Var}(Y) = \phi \mu^3 \]

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

\[ f(y) = \frac{\Gamma(y + 1/k)}{\Gamma(y + 1) \Gamma(1/k)} \frac{(k\mu)^y}{(1 + k\mu)^{y + 1/k}} \text{ for } y = 0, 1, 2, \ldots \]

\[ \phi = k \]

\[ \text{E}(Y) = \mu \]

\[ \text{Var}(Y) = \mu + \phi \mu^2 \]

For the negative binomial distribution, \( k \) is the estimated dispersion parameter that is displayed in the output.

Normal Distribution

\[ f(y) = \frac{1}{\sqrt{2\pi \sigma^2}} \exp \left( -\frac{1}{2} \left( \frac{y - \mu}{\sigma} \right)^2 \right) \text{ for } -\infty < y < \infty \]

\[ \phi = \sigma^2 \]

\[ \text{E}(Y) = \mu \]

\[ \text{Var}(Y) = \phi \]
Poisson Distribution

\[ f(y) = \frac{\mu^y e^{-\mu}}{y!} \quad \text{for } y = 0, 1, 2, \ldots \]
\[ E(Y) = \mu \]
\[ \text{Var}(Y) = \mu \]

(Shifted) t Distribution

\[ z = \frac{\Gamma(0.5(v + 1))}{\sqrt{\pi v\phi} \Gamma(0.5v)} \]
\[ f(y) = z \left(1 + \frac{(y - \mu)^2}{\nu\phi}\right)^{-0.5(v+1)} \quad \text{for } -\infty < y < \infty \]
\[ E(Y) = \mu \]
\[ \text{Var}(Y) = \phi\nu/(\nu - 2), \quad \nu \geq 3 \]

You can specify the degrees of freedom parameter \( \nu \) in the DISTRIBUTION=td(\nu) 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 \( \mu \), the dispersion \( \phi \), 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 \( Y_i \)s have independent, identical gamma distributions, each of which has an expected value \( E(Y_i) = \phi(2-p)\mu^{p-1} \) 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 \( 1.1 \leq p \) 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:

\[
\begin{align*}
E(Y) &= \mu \\
Var(Y) &= \phi \mu^p
\end{align*}
\]

**Weibull Distribution**

\[
f(y) = \frac{1}{\mu \phi} \left( \frac{y}{\mu} \right)^{\frac{1}{\phi} - 1} \exp \left( -\frac{y}{\mu} \right)^{\frac{1}{\phi}} \quad \text{for } 0 < y < \infty
\]

\[
E(Y) = \mu \Gamma(1 + \phi)
\]

\[
Var(Y) = \mu^2 [\Gamma(1 + 2\phi) - \Gamma^2(1 + \phi)], \quad \phi > 0
\]

**Log-Likelihood Functions**

The GENSELECT procedure forms the log-likelihood functions of the various models as

\[
L(\mu; y) = \sum_{i=1}^{n} f_i l(\mu_i; y_i, w_i)
\]

where \(l(\mu_i; y_i, w_i)\) is the log-likelihood contribution of the \(i\)th 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 \(\mu_i\) for each observation \(i\) is related to the regression parameters \(\beta_i\) through the linear predictor \(\eta_i = x_i' \beta\) by

\[
\mu_i = g^{-1}(\eta_i)
\]

where \(g\) is the link function.

**Beta Distribution**

\[
l(\mu_i; y_i, w_i) = \log \left\{ \frac{\Gamma(\phi / w_i)}{\Gamma(\mu_i \phi / w_i) \Gamma((1 - \mu_i) \phi / w_i)} \right\}
\]

\[
+ (\mu_i \phi / w_i - 1) \log \{y_i\}
\]

\[
+ ((1 - \mu_i) \phi / w_i - 1) \log \{1 - y_i\}
\]

where \(\phi\) is the dispersion parameter that is displayed in the output.
Log-Likelihood Functions

Binary Distribution

\[ \eta_i = x_i' \beta \]
\[ \mu_i = g^{-1}(\eta_i) \]
\[ l(\mu_i; y_i) = y_i \log\{\mu_i\} + (1 - y_i) \log\{1 - \mu_i\} \]

Here, \( \mu_i \) 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 \( \eta_i \) 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

\[ \eta_i = x_i' \beta \]
\[ \mu_i = g^{-1}(\eta_i) \]
\[ l(\mu_i; y_i, w_i) = w_i \left( y_i \log\{\mu_i\} + (n_i - y_i) \log\{1 - \mu_i\}\right) \\
+ w_i \left( \log\{\Gamma(n_i + 1)\} - \log\{\Gamma(y_i + 1)\} - \log\{\Gamma(n_i - y_i + 1)\}\right) \]

where \( y_i \) and \( n_i \) are the values of the events and trials of the \( i \)th observation, respectively. The value \( \mu_i \) 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

\[ \eta_i = x_i' \beta \]
\[ \mu_i = g^{-1}(\eta_i) \]
\[ l(\mu_i; y_i, w_i) = \frac{w_i}{\phi} \log\left( \frac{w_i y_i}{\phi \mu_i} \right) - \frac{w_i y_i}{\phi \mu_i} - \log\{y_i \} - \log\left( \Gamma\left( \frac{w_i}{\phi} \right) \right) \]

For the gamma distribution, \( \nu = \frac{1}{\phi} \) is the estimated dispersion parameter that is displayed in the output.
Generalized Poisson Distribution

\[\xi_i = (1 - \exp\{-\phi\}/w_i)\]
\[\mu_i^* = \mu_i - \xi_i(y_i - i)\]
\[l(\mu_i^*; y_i, w_i) = \log\{\mu_i^* - \xi_i y_i\} + (y_i - 1) \log\{\mu_i^*\} - \mu_i^* - \log\{\Gamma(y_i + 1)\}\]

where \(\phi\) is the dispersion parameter that is displayed in the output.

Geometric Distribution

\[l(\mu_i; y_i, w_i) = y_i \log\left\{\frac{\mu_i}{w_i}\right\} - (y_i + w_i) \log\left\{1 + \frac{\mu_i}{w_i}\right\} + \log\left\{\frac{\Gamma(y_i + w_i)}{\Gamma(w_i)\Gamma(y_i + 1)}\right\}\]

Inverse Gaussian Distribution

\[\eta_i = x_i^T \beta\]
\[\mu_i = g^{-1}(\eta_i)\]
\[l(\mu_i; y_i, w_i) = -\frac{1}{2} \left[ \frac{w_i(y_i - \mu_i)^2}{y_i \mu^2 \phi} + \log\left(\frac{\phi y_i^3}{w_i}\right) + \log(2\pi) \right]\]

where \(\phi\) is the dispersion parameter that is displayed in the output.

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 \(i\)th observation is

\[l(\mu_i; y_i, w_i) = w_i \sum_{j=1}^{J} y_{ij} \log\{\mu_{ij}\}\]

In this expression, \(J\) denotes the number of response categories (the number of possible outcomes) and \(\mu_{ij}\) is the probability that the \(i\)th observation takes on the response value associated with category \(j\). The category probabilities must satisfy

\[\sum_{j=1}^{J} \mu_j = 1\]

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.
Log-Likelihood Functions

Negative Binomial Distribution

\[ \eta_i = x_i' \beta \]
\[ \mu_i = g^{-1}(\eta_i) \]
\[ l(\mu_i; y_i, w_i) = y_i \log \left( \frac{k \mu}{w_i} \right) - (y_i + w_i/k) \log \left( 1 + \frac{k \mu}{w_i} \right) + \log \left( \frac{\Gamma(y_i + w_i/k)}{\Gamma(y_i + 1) \Gamma(w_i/k)} \right) \]

where \( k \) is the negative binomial dispersion parameter that is displayed in the output.

Normal Distribution

\[ \eta_i = x_i' \beta \]
\[ \mu_i = g^{-1}(\eta_i) \]
\[ l(\mu_i; y_i, w_i) = -\frac{1}{2} \left[ \frac{w_i(y_i - \mu_i)^2}{\phi} + \log \left( \frac{\phi}{w_i} \right) + \log(2\pi) \right] \]

where \( \phi \) is the dispersion parameter that is displayed in the output.

Poisson Distribution

\[ \eta_i = x_i' \beta \]
\[ \mu_i = g^{-1}(\eta_i) \]
\[ l(\mu_i; y_i, w_i) = w_i [y_i \log(\mu_i) - \mu_i - \log(y_i!)] \]

\( t \) Distribution

\[ z_i = -0.5 \log\{\phi/w_i\} + \log \{\Gamma(0.5(v + 1))\} - \log \{\Gamma(0.5v)\} - 0.5 \times \log \{\pi v\} \]
\[ l(\mu_i; y_i, w_i) = -\left( \frac{v + 1}{2} \right) \log \left\{ 1 + \frac{w_i}{v} \frac{(y_i - \mu_i)^2}{\phi} \right\} + z_i \]

where \( \phi \) 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

\[ L(\mu; y) = \sum_{i=1}^{n} f_i l(\mu_i; y_i, w_i) \]

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 111. 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

\[ Q_p(y, \mu, \phi, p) = \sum_i q(y_i, \mu_i, \phi, p) \]

where the contribution from an observation is

\[ q(y_i, \mu_i, \phi, p) = -0.5 \log\left(2\pi \frac{\phi}{w_i} y_i^p\right) - w_i \left( \frac{y_i^{2-p} - (2 - p)y_i \mu_i^{1-p} + (1 - p)\mu_i^{2-p}}{(1 - p)(1 - p)} \right) / \phi \]

where 1 < p < 2.

**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 \( \phi \) 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 \( j \)th subject, and let \( x_j \) 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 \( b \) that correctly allocates all observations to their response groups; that is,

\[
\begin{align*}
& b'x_j > 0 \quad Y_j = 0 \\
& b'x_j < 0 \quad Y_j = 1
\end{align*}
\]

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 \( b \) such that

\[
\begin{align*}
& b'x_j \geq 0 \quad Y_j = 0 \\
& b'x_j \leq 0 \quad Y_j = 1
\end{align*}
\]

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 63 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 \( X = (x_1, x_2, \ldots, x_m) \) denote the matrix of covariates, and let \( y \) denote the response. Then for a given parameter \( t \), the LASSO regression coefficients \( \beta = (\beta_1, \beta_2, \ldots, \beta_m) \) are the solution to the constrained least squares problem:

\[
\min ||y - X\beta||^2 \quad \text{subject to} \quad \sum_{j=1}^{m} |\beta_j| \leq t
\]

For generalized linear models, the LASSO regression coefficients \( \beta = (\beta_1, \beta_2, \ldots, \beta_m) \) are the solution to the constrained optimization problem,
\[
\min \{ -L(\mu; y) \} \quad \text{subject to} \quad \sum_{j=1}^{m} |\beta_j| \leq t
\]

where \( L \) is the log-likelihood function defined in the section “Log-Likelihood Functions” on page 112.

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 \( \phi \) 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.

**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 118.

**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:

\[
\begin{align*}
\text{AIC} & = -2l + 2p \\
\text{AICC} & = \begin{cases} 
-2l + 2pF/(F - p - 1) & \text{when } F > p + 2 \\
-2l + 2p(p + 2) & \text{otherwise}
\end{cases} \\
\text{SBC} & = -2l + p \log(F)
\end{align*}
\]

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.
Predicted Values and Regression Diagnostics

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 3.11 (Brier 1950; Murphy 1973); it is also called the Brier score or Brier reliability.

Table 3.11 Average Square Error Computations

<table>
<thead>
<tr>
<th>Response Type</th>
<th>ASE (Brier Score)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary</td>
<td>$\frac{1}{F} \sum_i f_i (y_i (1 - \hat{\pi}_i)^2 + (1 - y_i)\hat{\pi}_i^2)$</td>
</tr>
<tr>
<td>Binomial</td>
<td>$\frac{1}{F} \sum_i f_i (r_i/t_i - \hat{\pi}_i)^2$</td>
</tr>
<tr>
<td>Multinomial</td>
<td>$\frac{1}{F} \sum_i f_i \sum_j (y_{ij} - \hat{\pi}_{ij})^2$</td>
</tr>
</tbody>
</table>

In Table 3.11, $F = \sum_i f_i$, $r_i$ 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 $\hat{\pi}_i$ is the predicted probability of an event. For polytomous response models, $y_{ij} = 1$ if the $i$th observation has response level $j$, and $\pi_{ij}$ 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 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 $x_i$ for the $i$th observation in your data table, and the model-predicted parameter estimates $\hat{\beta}$, you can write the linear predictor $\hat{\eta}_i = x'_i \hat{\beta}$. The mean of the $i$th observation $\mu_i(\hat{\beta})$, or the model-predicted event probability $\hat{\pi}_i$, is $\mu_i(\hat{\beta}) = \hat{\pi}_i = g^{-1}(\eta_i)$, where the link function $g$ is chosen by specifying the LINK= option. The variance function of the distribution is $V(\mu)$, and $\Sigma$ is the estimated covariance of $\hat{\beta}$. Denote the frequency of the $i$th observation as $f_i$ and the weight as $w_i$.

For multinomial 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 $\hat{\beta}$ consist of the slope parameters: $\hat{\eta}_{ij} = g(\Pr(Y \leq j | x_i) = \hat{\alpha}_j + x'_i \hat{\beta}$ and $\hat{\pi}_{ij} = \Pr(Y \leq j | x_i) = g^{-1}(\eta_{ij})$ for $1 \leq j < J$.

Confidence Intervals

Approximate confidence intervals for predicted values can be computed as follows. The variance of the linear predictor is estimated by

$$\sigma^2 = x'_i \Sigma x_i$$

For multinomial models, the variance also depends on the response function. Let $\delta_j$ be a $(J-1)$ column vector whose $j$th entry is equal to 1 and all other entries are equal to 0. Redefine $x_i = (\delta'_j, x'_i)'$, $\eta_i = \eta_{ij}$, and $\pi_i = \pi_{ij}$. Then

$$\hat{\sigma}^2(\eta_i) = x'_i \Sigma x_i$$
The asymptotic $100(1 - \alpha)\%$ confidence interval for $\eta_i$ 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 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 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

$$v_i = \frac{\phi V(\mu_i)}{f_i w_i}$$

For the $i$th observation, let

$$w_{ei} = v_i^{-1}(g'(\mu_i))^{-2}$$

where $g'(\mu_i)$ is the derivative of the link function evaluated at $\mu_i$. Let $W_e$ be the diagonal matrix, with $w_{ei}$ denoting the $i$th diagonal element. The weight matrix $W_e$ is used to compute the expected information matrix. Define the leverage, or hat-matrix diagonal, $h_i$, as the $i$th diagonal element of the matrix

$$W_e^{1/2}X(X'W_eX)^{-1}X'W_e^{1/2}$$

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 $i$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 residual is

$$y_{ij} - \hat{\pi}_{ij}$$

where $y_{ij} = 1$ if the $i$th 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.

The Pearson residual is the square root of the $i$th 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 $i$th 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,

$$d_i = 2f_i w_i t_i \left[ y_i / t_i \log \left( \frac{y_i / t_i}{\mu_i} \right) + (1 - y_i / t_i) \log \left( \frac{1 - y_i / t_i}{1 - \mu_i} \right) \right]$$

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 $\hat{\beta}$. 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 $i$th observation are computed as follows:

$$\text{DIFDEV}_i = r_{Di}^2 + \overline{C}_i$$
$$\text{DIFCHISQ}_i = \overline{C}_i / h_i$$
Joint Tests and Type 3 Tests

Linear hypotheses for $\beta$ are expressed in matrix form as

$$H_0: \mathbf{L}\beta = c$$

where $\mathbf{L}$ is a matrix of coefficients for the linear hypotheses and $c$ is a vector of constants. The vector of regression coefficients $\beta$ includes slope parameters and intercept parameters. The Wald chi-square statistic for testing $H_0$ is computed as

$$\chi^2_W = (\mathbf{L}\hat{\beta} - c)'[\mathbf{L}\hat{\Sigma}(\hat{\beta})\mathbf{L}']^{-1}(\mathbf{L}\hat{\beta} - c)$$

where $\hat{\Sigma}(\hat{\beta})$ is the estimated covariance matrix. Under $H_0$, $\chi^2_W$ 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 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 68 in Chapter 2, “Shared Concepts.”
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 3.12 summarizes the optimization techniques available in PROC GENSELECT.

<table>
<thead>
<tr>
<th>TECHNIQUE=</th>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUREG</td>
<td>Trust region method</td>
</tr>
<tr>
<td>NEWRAP</td>
<td>Newton-Raphson method with line search</td>
</tr>
<tr>
<td>NRRIDG</td>
<td>Newton-Raphson method with ridging</td>
</tr>
<tr>
<td>QUANEW</td>
<td>Quasi-Newton methods</td>
</tr>
<tr>
<td>DBLDOG</td>
<td>Double-dogleg method</td>
</tr>
<tr>
<td>CONGRA</td>
<td>Conjugate gradient methods</td>
</tr>
<tr>
<td>NMSIMP</td>
<td>Nelder-Mead simplex method</td>
</tr>
</tbody>
</table>

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 68 in Chapter 2, “Shared Concepts,” is helpful in choosing a suitable optimization algorithm.

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.

**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 106.

**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 a multinomial model, the note that follows the “Response Profile” table indicates how the ordered response levels are accumulated. 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.
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.

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 118.

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.

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.

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.

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.

Output CAS Tables

When you specify the OUTPUT statement, the GENSELECT procedure produces the output data table along with a table that lists the CAS library, the data table name, and the number of rows and columns in that data table.
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 3.13.

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Statement</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClassInfo</td>
<td>Level information from the CLASS statement</td>
<td>CLASS</td>
<td>Default</td>
</tr>
<tr>
<td>ConvergenceStatus</td>
<td>Status of optimization at conclusion of optimization</td>
<td>PROC GENSELECT</td>
<td>Default</td>
</tr>
<tr>
<td>CorrB</td>
<td>Correlation matrix of parameter estimates</td>
<td>PROC GENSELECT</td>
<td>CORRB</td>
</tr>
<tr>
<td>CovB</td>
<td>Covariance matrix of parameter estimates</td>
<td>PROC GENSELECT</td>
<td>COVB</td>
</tr>
<tr>
<td>Dimensions</td>
<td>Model dimensions</td>
<td>PROC GENSELECT</td>
<td>Default</td>
</tr>
<tr>
<td>EntryCandidates</td>
<td>Details about candidates for entry into the model</td>
<td>SELECTION</td>
<td>METHOD=FORWARD DETAILS=STEP</td>
</tr>
<tr>
<td>FitStatistics</td>
<td>Fit statistics</td>
<td>PROC GENSELECT</td>
<td>Default</td>
</tr>
<tr>
<td>IterHistory</td>
<td>Iteration history</td>
<td>PROC GENSELECT</td>
<td>ITHIST</td>
</tr>
<tr>
<td>ModelInfo</td>
<td>Information about the modeling environment</td>
<td>PROC GENSELECT</td>
<td>Default</td>
</tr>
<tr>
<td>ModelAnova</td>
<td>Model analysis of variance (Type III)</td>
<td>MODEL</td>
<td>TYPE3</td>
</tr>
<tr>
<td>NObs</td>
<td>Number of observations read and used, and number of events and trials, if applicable</td>
<td>PROC GENSELECT</td>
<td>Default</td>
</tr>
<tr>
<td>OutCASTblFull</td>
<td>Library and name of the output data table, and number of rows and columns in the table</td>
<td>OUTPUT</td>
<td>OUT=</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DISPLAYOUT</td>
<td></td>
</tr>
<tr>
<td>ParameterEstimates</td>
<td>Solutions for the parameter estimates associated with effects in MODEL statements</td>
<td>PROC GENSELECT</td>
<td>Default</td>
</tr>
<tr>
<td>PredProbName</td>
<td>Displays the predicted probability variable in the scoring code associated with each response level</td>
<td>CODE</td>
<td>PCATALL</td>
</tr>
<tr>
<td>RemovalCandidates</td>
<td>Details about candidates for removal from the model</td>
<td>SELECTION</td>
<td>METHOD=BACKWARD DETAILS=STEP</td>
</tr>
<tr>
<td>ResponseProfile</td>
<td>Response categories for binary and multinomial data</td>
<td>Default</td>
<td></td>
</tr>
<tr>
<td>SelectedEffects</td>
<td>List of effects selected for the model</td>
<td>SELECTION</td>
<td>Default</td>
</tr>
</tbody>
</table>
Examples: GENSELECT Procedure

Example 3.1: Model Selection

The following statements examine the same data set that is used in the section “Getting Started: GENSELECT Procedure” on page 80, 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 80. It contains 100 observations on a count response variable (Y), a continuous variable (Total) to be used in Example 3.2, 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:

```
proc genselect data=mycas.getStarted;
    class C1-C5;
    model Y = C1-C5 / Distribution=Poisson;
    selection method=forward details=all;
run;
```

The model selection tables are shown in Output 3.1.1 through Output 3.1.4.

The “Selection Information” table in Output 3.1.1 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 3.1.1  Selection Information

The GENSELECT Procedure

<table>
<thead>
<tr>
<th>Selection Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selection Method</td>
</tr>
<tr>
<td>Select Criterion</td>
</tr>
<tr>
<td>Stop Criterion</td>
</tr>
<tr>
<td>Effect Hierarchy Enforced</td>
</tr>
<tr>
<td>Stop Horizon</td>
</tr>
</tbody>
</table>

The “Selection Summary” table in Output 3.1.2 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 3.1.2  Selection Summary Information

<table>
<thead>
<tr>
<th>Selection Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step</td>
</tr>
<tr>
<td>0 Intercept</td>
</tr>
<tr>
<td>2 C2</td>
</tr>
<tr>
<td>3 C5</td>
</tr>
<tr>
<td>4 C1</td>
</tr>
<tr>
<td>5 C4</td>
</tr>
</tbody>
</table>

Selection stopped at a local minimum of the STOP criterion.

The model at step 2 is selected.

Selected Effects: Intercept C2 C5

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 3.1.3 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 3.6. 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.
Chapter 3: The GENSELECT Procedure

Output 3.1.3  Fit Statistics

<table>
<thead>
<tr>
<th>Fit Statistics</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-2 Log Likelihood</td>
<td>303.58036</td>
</tr>
<tr>
<td>AIC (smaller is better)</td>
<td>317.58036</td>
</tr>
<tr>
<td>AICC (smaller is better)</td>
<td>318.79775</td>
</tr>
<tr>
<td>SBC (smaller is better)</td>
<td>335.81655</td>
</tr>
</tbody>
</table>

The parameter estimates of the selected model are shown in Output 3.1.4. 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 3.1.4  Parameter Estimates

<table>
<thead>
<tr>
<th>Parameter</th>
<th>DF</th>
<th>Estimate</th>
<th>Error</th>
<th>Chi-Square</th>
<th>Pr &gt; ChiSq</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intercept</td>
<td>1</td>
<td>0.528024</td>
<td>0.215231</td>
<td>6.0186</td>
<td>0.0142</td>
</tr>
<tr>
<td>C2</td>
<td>1</td>
<td>0.962636</td>
<td>0.236921</td>
<td>16.5088</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>C2 1</td>
<td>1</td>
<td>0.845132</td>
<td>0.233634</td>
<td>13.0851</td>
<td>0.0003</td>
</tr>
<tr>
<td>C2 2</td>
<td>1</td>
<td>0.213247</td>
<td>0.248896</td>
<td>0.7341</td>
<td>0.3916</td>
</tr>
<tr>
<td>C2 3</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C5</td>
<td>1</td>
<td>-0.763491</td>
<td>0.207082</td>
<td>13.5933</td>
<td>0.0002</td>
</tr>
<tr>
<td>C5 1</td>
<td>1</td>
<td>-0.785915</td>
<td>0.201271</td>
<td>15.2471</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>C5 2</td>
<td>1</td>
<td>-0.541212</td>
<td>0.208661</td>
<td>6.7275</td>
<td>0.0095</td>
</tr>
<tr>
<td>C5 3</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Example 3.2: Gamma Model

The following statements examine the data set getStarted, which is used in the section “Getting Started: GENSELECT Procedure” on page 80, 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=mycas.getStarted;
class C1-C5;
model Total = C1-C5 / Distribution=Gamma Link=Log CLB;
code File='ScoringParameters.txt';
run;
```

The “Parameter Estimates” table in Output 3.2.1 shows the resulting regression model parameter estimates and the estimated gamma dispersion parameter.
Now suppose you want to compute predicted values for some different data. If \( x \) is a vector of explanatory variables that might not be in the original data and \( \hat{\beta} \) is the vector of estimated regression parameters from the model, then \( \mu = g^{-1}(x'\hat{\beta}) \) 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
;

<table>
<thead>
<tr>
<th>Parameter</th>
<th>DF</th>
<th>Estimate</th>
<th>Standard Error</th>
<th>Chi-Square</th>
<th>Pr &gt; ChiSq</th>
<th>95% Confidence Limits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intercept</td>
<td>1</td>
<td>4.028096</td>
<td>0.454883</td>
<td>78.4153</td>
<td>&lt;.0001</td>
<td>3.13654 - 4.91965</td>
</tr>
<tr>
<td>C1</td>
<td>1</td>
<td>-0.664442</td>
<td>0.256719</td>
<td>0.0630</td>
<td>0.8018</td>
<td>-0.56760 - 0.43872</td>
</tr>
<tr>
<td>C1</td>
<td>1</td>
<td>-1.308470</td>
<td>0.318002</td>
<td>16.9305</td>
<td>&lt;.0001</td>
<td>-1.93174 - 0.68520</td>
</tr>
<tr>
<td>C1</td>
<td>1</td>
<td>0.295802</td>
<td>0.257834</td>
<td>1.3162</td>
<td>0.2513</td>
<td>-0.20954 - 0.80115</td>
</tr>
<tr>
<td>C2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2</td>
<td>1</td>
<td>1.154556</td>
<td>0.295110</td>
<td>15.3060</td>
<td>&lt;.0001</td>
<td>0.57615 - 1.73296</td>
</tr>
<tr>
<td>C2</td>
<td>1</td>
<td>0.824472</td>
<td>0.295273</td>
<td>7.7966</td>
<td>0.0052</td>
<td>0.24575 - 1.40320</td>
</tr>
<tr>
<td>C2</td>
<td>1</td>
<td>-0.287943</td>
<td>0.288421</td>
<td>0.9967</td>
<td>0.3181</td>
<td>-0.85324 - 0.27735</td>
</tr>
<tr>
<td>C2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C3</td>
<td>1</td>
<td>-0.008546</td>
<td>0.283509</td>
<td>0.0009</td>
<td>0.9760</td>
<td>-0.56421 - 0.54712</td>
</tr>
<tr>
<td>C3</td>
<td>1</td>
<td>-0.319783</td>
<td>0.276053</td>
<td>1.3419</td>
<td>0.2467</td>
<td>-0.86084 - 0.22177</td>
</tr>
<tr>
<td>C3</td>
<td>1</td>
<td>-0.071452</td>
<td>0.296418</td>
<td>0.0581</td>
<td>0.8095</td>
<td>-0.65242 - 0.50952</td>
</tr>
<tr>
<td>C3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4</td>
<td>1</td>
<td>-0.143018</td>
<td>0.288101</td>
<td>0.2464</td>
<td>0.6196</td>
<td>-0.70769 - 0.42165</td>
</tr>
<tr>
<td>C4</td>
<td>1</td>
<td>-0.219348</td>
<td>0.290441</td>
<td>0.5704</td>
<td>0.4501</td>
<td>-0.78860 - 0.34991</td>
</tr>
<tr>
<td>C4</td>
<td>1</td>
<td>0.091763</td>
<td>0.278635</td>
<td>0.1085</td>
<td>0.7419</td>
<td>-0.45435 - 0.63788</td>
</tr>
<tr>
<td>C4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C5</td>
<td>1</td>
<td>-1.227558</td>
<td>0.267605</td>
<td>21.0425</td>
<td>&lt;.0001</td>
<td>-1.75205 - 0.70306</td>
</tr>
<tr>
<td>C5</td>
<td>1</td>
<td>-0.560699</td>
<td>0.252238</td>
<td>4.9413</td>
<td>0.0262</td>
<td>-1.05508 - 0.06632</td>
</tr>
<tr>
<td>C5</td>
<td>1</td>
<td>-0.252865</td>
<td>0.259980</td>
<td>0.9473</td>
<td>0.3304</td>
<td>-0.76238 - 0.25645</td>
</tr>
<tr>
<td>C5</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dispersion</td>
<td>1</td>
<td>1.672305</td>
<td>0.238205</td>
<td></td>
<td></td>
<td>1.26494 - 2.21086</td>
</tr>
</tbody>
</table>

Now suppose you want to compute predicted values for some different data. If \( x \) is a vector of explanatory variables that might not be in the original data and \( \hat{\beta} \) is the vector of estimated regression parameters from the model, then \( \mu = g^{-1}(x'\hat{\beta}) \) 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.
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 3.2.2.

```sas
data Scores;
  set ScoringData;
  %inc 'ScoringParameters.txt';
run;
proc print data=Scores;
run;
```

Output 3.2.2  Predicted Values for Scoring Data

<table>
<thead>
<tr>
<th>Obs</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
<th>C5</th>
<th>P_Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>27.449</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>10.349</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>12.590</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>25.020</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>32.222</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>46.020</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>7.282</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>138.244</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>13.523</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>82.063</td>
</tr>
</tbody>
</table>

References


Chapter 4
The KCLUS Procedure

Contents

Overview: KCLUS Procedure ................................................................. 136
PROC KCLUS Features ................................................................. 136
Using CAS Sessions and CAS Engine Librefs ................................. 137
Getting Started: KCLUS Procedure .................................................. 137
Syntax: KCLUS Procedure ............................................................... 141
PROC KCLUS Statement ............................................................... 141
CODE Statement ............................................................... 146
DISPLAY Statement ............................................................... 146
DISPLAYOUT Statement ............................................................ 147
FREQ Statement ............................................................... 148
INPUT Statement ............................................................... 148
SCORE Statement ............................................................... 148
Details: KCLUS Procedure ............................................................. 149
Obtaining the Statistics for Clustering .................................................. 149
Missing Values ............................................................... 149
Initial Seed Selection ............................................................... 150
Standardization ............................................................... 150
Finding the Number of Clusters ........................................................ 150
Clustering Nominal Variables .......................................................... 151
  Dissimilarity Measures ............................................................... 152
  Computing Cluster Centers .......................................................... 153
Displayed Output ............................................................... 153
  Number of Observations ............................................................... 153
  Model Information ............................................................... 153
  Cluster Summary ............................................................... 153
  Iteration History ............................................................... 154
  Descriptive Statistics ............................................................... 154
  Within-Cluster Statistics .............................................................. 154
  Cluster Summary for Nominal Variables ........................................... 154
  Frequencies for Nominal Variables ................................................ 155
  Standardization ............................................................... 155
  Aligned Box Criterion Parameters .................................................. 155
  Aligned Box Criterion Statistics ..................................................... 156
  Estimated Number of Clusters ........................................................ 156
  Output CAS Tables ............................................................... 156
ODS Table Names ............................................................... 156
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 one or more quantitative variables. 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 and uses the \( k \)-modes algorithm for clustering nominal input variables. The KCLUS procedure uses only interval variables or nominal variables to perform clustering. It does not perform clustering for a mix of interval and nominal input variables.

The KCLUS procedure uses the least squares (\( L_2 \)) estimation in the \( k \)-means clustering method to compute the cluster centroids. In this method, each iteration reduces the criterion (for example, the least squares criterion for the Euclidean distance) until convergence is achieved or 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:

- can execute clustering in parallel
- is highly multithreaded
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:

```sql
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:

```sql
cas mysess;
libname mycas cas sessref=mysess;
```

The CAS statement creates the CAS session named `mysess`, and the LIBNAME statement creates the `mycas` 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:

```sql
cas mysess terminate;
```

For more information about the CAS and LIBNAME statements, see the section “Introduction to Shared Concepts” on page 8 in Chapter 2, “Shared Concepts.”

Getting Started: KCLUS 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 8 and “Loading a SAS Data Set onto a CAS Server” on page 9 in Chapter 2, “Shared Concepts.”

This example shows how to use the KCLUS procedure to compute clusters of observations in a CAS table.
Suppose you want to group the observations in the input CAS table mycas.inpData, in which the variables are raw measures on interval scales.

The following DATA step creates the input data table, mycas.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.

```plaintext
data mycas.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 CAS engine libref is named mycas, but you can substitute any appropriately defined CAS engine libref.

The following statements run PROC KCLUS and output the results to ODS tables:

```plaintext
proc kclus data=mycas.inpData maxclusters=3;
  input x y;
  freq freq;
run;
```

Figure 4.1 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 4.1** Number of Observations

**Using PROC KCLUS to Analyze Data**

**The KCLUS Procedure**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Observations Read</td>
<td>2900</td>
</tr>
<tr>
<td>Number of Observations Used</td>
<td>2700</td>
</tr>
<tr>
<td>Sum of Frequencies Read</td>
<td>3900</td>
</tr>
<tr>
<td>Sum of Frequencies Used</td>
<td>3700</td>
</tr>
</tbody>
</table>

**Figure 4.2** 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 4.2** shows the number of clusters and the default values for other options.

**Figure 4.2** Model Information

<table>
<thead>
<tr>
<th>Model Information</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum Iterations</td>
<td>10</td>
</tr>
<tr>
<td>Stop Criterion</td>
<td>Cluster Change</td>
</tr>
<tr>
<td>Stop Criterion Value</td>
<td>0</td>
</tr>
<tr>
<td>Clusters</td>
<td>3</td>
</tr>
<tr>
<td>Initialization</td>
<td>Forgy</td>
</tr>
<tr>
<td>Seed</td>
<td>987467829</td>
</tr>
<tr>
<td>Distance</td>
<td>Euclidean</td>
</tr>
<tr>
<td>Standardization</td>
<td>None</td>
</tr>
<tr>
<td>Interval Imputation</td>
<td>None</td>
</tr>
</tbody>
</table>

For each cluster, **Figure 4.3** 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 4.3** also displays information about the nearest cluster to that cluster and the distance between their centroids.

**Figure 4.3** Cluster Summary

<table>
<thead>
<tr>
<th>Cluster Summary for Interval Variables</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Distance from Cluster</td>
<td></td>
</tr>
<tr>
<td>Centroid to Observation</td>
<td></td>
</tr>
<tr>
<td>Distance to Nearest Cluster</td>
<td></td>
</tr>
<tr>
<td>Nearest Cluster Centroid</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Frequency</th>
<th>Minimum</th>
<th>Maximum</th>
<th>Average</th>
<th>SSE</th>
<th>Standard Deviation</th>
<th>Nearest Cluster</th>
<th>Distance to Nearest Cluster Centroid</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1175</td>
<td>0.0202</td>
<td>29.2304</td>
<td>4.4741</td>
<td>31968.1</td>
<td>5.2160</td>
<td>2</td>
<td>13.8882</td>
</tr>
<tr>
<td>2</td>
<td>2038</td>
<td>0.2289</td>
<td>31.7970</td>
<td>5.2019</td>
<td>78608.4</td>
<td>6.2106</td>
<td>1</td>
<td>13.8882</td>
</tr>
<tr>
<td>3</td>
<td>487</td>
<td>0.5652</td>
<td>30.4447</td>
<td>10.7298</td>
<td>73156.7</td>
<td>12.2564</td>
<td>2</td>
<td>18.3687</td>
</tr>
</tbody>
</table>
Figure 4.4 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. If the input variables are nominal, then the “Iteration History” table displays Within Distance 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.

<table>
<thead>
<tr>
<th>Iteration Number</th>
<th>SSE</th>
<th>SSE Change</th>
<th>Stop Criterion</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>260987</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>190242</td>
<td>-70745</td>
<td>4.666667</td>
</tr>
<tr>
<td>2</td>
<td>185578</td>
<td>-4664.513827</td>
<td>2.407407</td>
</tr>
<tr>
<td>3</td>
<td>184388</td>
<td>-1190.314300</td>
<td>1.333333</td>
</tr>
<tr>
<td>4</td>
<td>184034</td>
<td>-353.882617</td>
<td>0.703704</td>
</tr>
<tr>
<td>5</td>
<td>183866</td>
<td>-167.400789</td>
<td>0.666667</td>
</tr>
<tr>
<td>6</td>
<td>183768</td>
<td>-98.177548</td>
<td>0.333333</td>
</tr>
<tr>
<td>7</td>
<td>183739</td>
<td>-29.222378</td>
<td>0.074074</td>
</tr>
<tr>
<td>8</td>
<td>183733</td>
<td>-5.722173</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 4.5 and Figure 4.6 show statistics for each variable in the INPUT statement. Figure 4.5 shows the variable statistics for all the observations in the input data table, and Figure 4.6 shows the variable statistics for the observations that belong to a specific cluster.

**Figure 4.5** Descriptive Statistics

<table>
<thead>
<tr>
<th>Variable</th>
<th>Mean</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>11.020648</td>
<td>8.189686</td>
</tr>
<tr>
<td>y</td>
<td>10.756547</td>
<td>9.180654</td>
</tr>
</tbody>
</table>

**Figure 4.6** Within-Cluster Statistics

<table>
<thead>
<tr>
<th>Variable</th>
<th>Cluster</th>
<th>Mean</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>1</td>
<td>18.9930</td>
<td>6.0294</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>9.5626</td>
<td>10.5049</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>-2.1130</td>
<td>9.0921</td>
</tr>
<tr>
<td>y</td>
<td>1</td>
<td>19.5808</td>
<td>7.9047</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>9.3852</td>
<td>11.0225</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>-4.7954</td>
<td>8.3923</td>
</tr>
</tbody>
</table>
Syntax: KCLUS Procedure

The following statements are available in the KCLUS procedure:

```plaintext
PROC KCLUS < options > ;
  CODE < options > ;
  DISPLAY < table-list > </options> ;
  DISPLAYOUT table-spec-list </options> ;
  FREQ variable ;
  INPUT variables < LEVEL= NOMINAL | INTERVAL > ;
  SCORE OUT= CAS-libref.data-table < options > ;
```

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

```plaintext
PROC KCLUS < options > ;
```

The PROC KCLUS statement invokes the procedure. Table 4.1 summarizes the options available in the PROC KCLUS statement.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA=</td>
<td>Specifies the input data table</td>
</tr>
<tr>
<td>OUTSTAT(OUTITER)=</td>
<td>Specifies the output data table to contain cluster centroids</td>
</tr>
<tr>
<td>DISTANCE=</td>
<td>Specifies the distance measure for similarity measurement (used for interval input variables)</td>
</tr>
<tr>
<td>DISTANCE=</td>
<td>Specifies the distance measure for similarity measurement (used for nominal input variables)</td>
</tr>
<tr>
<td>INIT=</td>
<td>Specifies the method for obtaining the initial estimate of cluster centers</td>
</tr>
<tr>
<td>MAXCLUSTERS=</td>
<td>Specifies the number of clusters</td>
</tr>
<tr>
<td>MAXITER=</td>
<td>Specifies the maximum number of iterations</td>
</tr>
<tr>
<td>SEED=</td>
<td>Specifies the seed that is used for pseudorandom number generation</td>
</tr>
<tr>
<td>STOPCRITERION=</td>
<td>Specifies the stop criterion method to use for convergence</td>
</tr>
<tr>
<td>IMPUTE=</td>
<td>Specifies the imputation method for interval input variables</td>
</tr>
<tr>
<td>IMPUTENOM=</td>
<td>Specifies the imputation method for nominal input variables</td>
</tr>
<tr>
<td>STANDARDIZE=</td>
<td>Specifies the method for standardizing interval input variables</td>
</tr>
</tbody>
</table>

Table 4.1  continued

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Number of Clusters Estimation Options</strong></td>
<td></td>
</tr>
<tr>
<td>NOC=</td>
<td>Specifies the method for estimating the number of clusters</td>
</tr>
<tr>
<td><strong>Performance Options</strong></td>
<td></td>
</tr>
<tr>
<td>NTHREADS=</td>
<td>Specifies the number of threads for the computation</td>
</tr>
</tbody>
</table>

You can specify the following options:

**DATA=**CAS-libref.data-table
names the input data table for PROC KCLUS to use. The default is the most recently created data table. *CAS-libref.data-table* is a two-level name, where

*CAS-libref* refers to a collection of information that is defined in the LIBNAME statement and includes the *caslib*, 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 *CAS-libref*, see the section “Using CAS Sessions and CAS Engine Librefs” on page 137.

*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 151.
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 150.

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 MAX-CLUSTERS= option.

By default, NOC=NONE.

\textbf{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.

\textbf{OUTSTAT < (OUTITER) >=CAS-libref.data-table} 

creates the output data table that contains the cluster centroids for each cluster. \textit{CAS-libref.data-table} is a two-level name, where \textit{CAS-libref} refers to the caslib and session identifier, and \textit{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 137.

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:

\textbf{OUTITER} also outputs the cluster centroids to the OUTSTAT= data table for each iteration.

\textbf{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.

\textbf{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:

\textbf{NONE} does not standardize and uses the actual values for the input variables.

\textbf{RANGE} standardizes the input variables by using the range method. PROC KCLUS uses the minimum as the location and the range as the scale.

\textbf{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.

\textbf{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 \textit{convergence\_methods}:
**Chapter 4: The KCLUS Procedure**

**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 the sum of within-cluster distance for the $k$-modes algorithm.

---

**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 ODS 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, all ODS tables are sent to the client and then the client creates a subset. If both DISPLAY and ODS statements are used together, the DISPLAY statement takes precedence over the ODS statements. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

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

**CASESENSITIVE**

performs a case-sensitive comparison of table names in the table-list to ODS 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 ODS tables except those specified in the *table-list*.

**EXCLUDEALL**

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

**TRACE**

displays the ODS table names, labels, and paths.

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

A path is a table name that is prefixed with dot-separated grouping information. For example, a SelectionSummary table that is produced by a procedure during a selection routine might have the path *Bygroup1.Summary.SelectionSummary*. A partial pathname does not include all groups; for example, *Selection-Summary* and *Summary.SelectionSummary* are partial pathnames for *Bygroup1.Summary.SelectionSummary*.

When you specify a table name or partial pathname, all ODS 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 starts with a “/” or a “!”. 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.

---

**DISPLAYOUT Statement**

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

The DISPLAYOUT statement enables you to create CAS 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 CAS output tables to create. Each entry in the list has either a *key* or a *key=value* format:

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

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 (/):

**NOREPLACE**

does not replace an existing CAS output table of the same name.
REPEATED
replicates the CAS output tables on all nodes.

**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 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 the FREQ statement is not specified, 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.

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.

**SCORE Statement**

```
SCORE OUT=CAS-libref.data-table < option > ;
```

```
OUTPUT OUT=CAS-libref.data-table < option > ;
```

The SCORE statement causes the KCLUS procedure to write the cluster membership information of each observation to the output data table. This information includes the variables that are specified in the COPYVARS= option and two new variables, _CLUSTER_ID_ (the ID of the closest cluster) and _DISTANCE_ (the distance between the observation and the centroid of that cluster). If you specify STANDARDIZE=RANGE or STANDARDIZE=STD in the PROC KCLUS statement, then PROC KCLUS adds another column called _STANDARDIZED_DISTANCE_, which contains the distance between the standardized values of the observation and the standardized values of cluster centroid.

You must specify the following option:
OUT=\texttt{CAS-libref.data-table}

names the output data table for PROC KCLUS to use. You must specify this option before any other options. \texttt{CAS-libref.data-table} is a two-level name, where

\texttt{CAS-libref} refers to a collection of information that is defined in the LIBNAME statement and includes the \texttt{caslib}, 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 \texttt{CAS-libref}, see the section “Using CAS Sessions and CAS Engine Librefs” on page 137.

\texttt{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 \texttt{COPYVARS=} option, followed by the \texttt{_CLUSTER_ID} variable and the \texttt{_DISTANCE} variable, are added to the output data table that is specified in this option.

You can also specify the following option:

\texttt{COPYVAR=variable}

\texttt{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 \texttt{COPYVARS=} option accepts numeric and character variables.

---

**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 result of clustering because the distance measures that are computed for these observations do not reflect the actual values. To solve this problem, you can use the \texttt{IMPUTE=} option for interval input variables and the \texttt{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. 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. The NOC= option works only for interval variables. 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).

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.

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 two nominal variables \(j\).

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 two nominal variables \(j\) 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 global frequency-based dissimilarity measure takes into account the frequencies of levels of each variable in the input data table. The dissimilarity measure between two nominal variables \(j\) 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 two nominal variables \(j\) is

\[
\delta(x_j, y_j) = \begin{cases} 
1 - \frac{n_{x_j}}{n_c} & \text{if } x_j = y_j \\
1 & \text{if } x_j \neq y_j
\end{cases}
\]

\(n_c\) is the number of observations in cluster \(c\) and \(n_{x_j}\) 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.

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 used. The number of observations that are used is calculated by considering the value in the variable that is specified in the FREQ statement and the missing values in the observations if IMPUTE=NONE in the PROC KCLUS statement.

Model Information

The “Model Information” table displays the basic information about the parameters that are used in the cluster analysis. This information includes the maximum number of iterations, stop criterion method and the value for that criterion, number of clusters, initialization technique, seed value, distance method, standardization method, and imputation method.

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 change

Descriptive Statistics

The “Descriptive Statistics” table displays the mean and standard deviation for each interval variable in the INPUT statement. The mean and standard deviation are calculated by using all the observations in the input data table.

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 that cluster
• frequencies of the levels of each variable in the input data table.
• frequencies of the levels of each variable in that cluster

**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

Output CAS Tables

The “Output CAS Table” lists the number of observations and the number of variables that are included in the output data table.

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 4.2.

Table 4.2  ODS Tables Produced by PROC KCLUS

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Statement</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABCParameters</td>
<td>Parameters that are specified in the NOC= option</td>
<td>PROC</td>
<td>NOC=ABC</td>
</tr>
<tr>
<td>ABCResults</td>
<td>Estimated number of clusters and the criteria that are used to calculate them</td>
<td>PROC</td>
<td>NOC=ABC</td>
</tr>
<tr>
<td>ABCStats</td>
<td>Statistics that are computed for estimating the number of clusters</td>
<td>PROC</td>
<td>NOC=ABC</td>
</tr>
</tbody>
</table>
Examples: KCLUS 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 8 and “Loading a SAS Data Set onto a CAS Server” on page 9 in Chapter 2, “Shared Concepts.”

Example 4.1: Cluster Analysis

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 CAS engine libref in the first statement of the following DATA step:

```r
data mycas.iris;
  set sashelp.iris;
run;
```

The following statements perform clustering:

```r
proc kclus data=mycas.iris maxclusters=3 outstat(outiter)=kclusOutstat1;
  input SepalLength SepalWidth PetalLength PetalWidth;
  score out=mycas.kclusOut1 copyvars=(SepalLength SepalWidth
                                PetalLength PetalWidth Species);
run;
```

In this example, PROC KCLUS generates the data table mycas.kclusOut1, which contains the cluster membership information for each observation in the input data table. For each observation, the mycas.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 4.1.1 through Output 4.1.6.

### Output 4.1.1 Number of Observations

**Using PROC KCLUS to Analyze Data**

**The KCLUS Procedure**

<table>
<thead>
<tr>
<th>Model Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Observations Read</td>
</tr>
<tr>
<td>Number of Observations Used</td>
</tr>
</tbody>
</table>

### Output 4.1.2 Model Information

<table>
<thead>
<tr>
<th>Model Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum Iterations</td>
</tr>
<tr>
<td>Stop Criterion</td>
</tr>
<tr>
<td>Stop Criterion Value</td>
</tr>
<tr>
<td>Clusters</td>
</tr>
<tr>
<td>Initialization</td>
</tr>
<tr>
<td>Seed</td>
</tr>
<tr>
<td>Distance</td>
</tr>
<tr>
<td>Standardization</td>
</tr>
<tr>
<td>Interval Imputation</td>
</tr>
</tbody>
</table>
### Output 4.1.3 Cluster Summary

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Frequency</th>
<th>Minimum</th>
<th>Maximum</th>
<th>Average</th>
<th>SSE</th>
<th>Standard Deviation</th>
<th>Nearest Cluster</th>
<th>Distance to Nearest Cluster Centroid</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>50</td>
<td>0.6618</td>
<td>12.4803</td>
<td>4.8171</td>
<td>1515.1</td>
<td>5.5047</td>
<td>2</td>
<td>33.4949</td>
</tr>
<tr>
<td>2</td>
<td>61</td>
<td>2.3571</td>
<td>16.4680</td>
<td>7.3111</td>
<td>3829.1</td>
<td>7.9229</td>
<td>3</td>
<td>17.8842</td>
</tr>
<tr>
<td>3</td>
<td>39</td>
<td>2.3945</td>
<td>15.5156</td>
<td>7.3185</td>
<td>2541.4</td>
<td>8.0724</td>
<td>2</td>
<td>17.8842</td>
</tr>
</tbody>
</table>

### Output 4.1.4 Iteration History

<table>
<thead>
<tr>
<th>Iteration Number</th>
<th>SSE</th>
<th>SSE Change</th>
<th>Stop Criterion</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>17261</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>7917.116083</td>
<td>-9343.883917</td>
<td>1.333333</td>
</tr>
<tr>
<td>2</td>
<td>7892.130972</td>
<td>-24.985111</td>
<td>0.666667</td>
</tr>
<tr>
<td>3</td>
<td>7885.566583</td>
<td>-6.564390</td>
<td>0</td>
</tr>
</tbody>
</table>

### Output 4.1.5 Descriptive Statistics

<table>
<thead>
<tr>
<th>Variable</th>
<th>Mean</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>SepalLength</td>
<td>58.433333</td>
<td>8.280661</td>
</tr>
<tr>
<td>SepalWidth</td>
<td>30.573333</td>
<td>4.358663</td>
</tr>
<tr>
<td>PetalLength</td>
<td>37.580000</td>
<td>17.652982</td>
</tr>
<tr>
<td>PetalWidth</td>
<td>11.993333</td>
<td>7.622377</td>
</tr>
</tbody>
</table>
Output 4.1.6 Within-Cluster Statistics

<table>
<thead>
<tr>
<th>Variable</th>
<th>Cluster</th>
<th>Mean</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>SepalLength</td>
<td>1</td>
<td>50.0600</td>
<td>3.5249</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>58.8361</td>
<td>4.4803</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>68.5385</td>
<td>4.8820</td>
</tr>
<tr>
<td>SepalWidth</td>
<td>1</td>
<td>34.2800</td>
<td>3.7906</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>27.4098</td>
<td>2.9290</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>30.7692</td>
<td>2.8696</td>
</tr>
<tr>
<td>PetalLength</td>
<td>1</td>
<td>14.6200</td>
<td>1.7366</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>43.8852</td>
<td>5.1157</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>57.1538</td>
<td>5.1018</td>
</tr>
<tr>
<td>PetalWidth</td>
<td>1</td>
<td>2.4600</td>
<td>1.0539</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>14.3443</td>
<td>2.9994</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>20.5385</td>
<td>2.9633</td>
</tr>
</tbody>
</table>

The following statements extract the first 10 observations from the output data table; they are shown in Output 4.1.7.

```
proc print noobs data=mycas.kclusOut1(obs=10);
run;
```

Output 4.1.7 First 10 Observations in the Output Data Table

Using PROC KCLUS to Analyze Data

<table>
<thead>
<tr>
<th>SepalLength</th>
<th>SepalWidth</th>
<th>PetalLength</th>
<th>PetalWidth</th>
<th>Species</th>
<th><em>CLUSTER_ID</em></th>
<th><em>DISTANCE</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>33</td>
<td>14</td>
<td>2</td>
<td>Setosa</td>
<td>1</td>
<td>1.4959946524</td>
</tr>
<tr>
<td>46</td>
<td>34</td>
<td>14</td>
<td>3</td>
<td>Setosa</td>
<td>1</td>
<td>1.41518670499</td>
</tr>
<tr>
<td>46</td>
<td>36</td>
<td>10</td>
<td>2</td>
<td>Setosa</td>
<td>1</td>
<td>6.4029680618</td>
</tr>
<tr>
<td>51</td>
<td>33</td>
<td>17</td>
<td>5</td>
<td>Setosa</td>
<td>1</td>
<td>3.8259639308</td>
</tr>
<tr>
<td>55</td>
<td>35</td>
<td>13</td>
<td>2</td>
<td>Setosa</td>
<td>1</td>
<td>5.2685861481</td>
</tr>
<tr>
<td>48</td>
<td>31</td>
<td>16</td>
<td>2</td>
<td>Setosa</td>
<td>1</td>
<td>4.1373904819</td>
</tr>
<tr>
<td>52</td>
<td>34</td>
<td>14</td>
<td>2</td>
<td>Setosa</td>
<td>1</td>
<td>2.1066561181</td>
</tr>
<tr>
<td>49</td>
<td>36</td>
<td>14</td>
<td>1</td>
<td>Setosa</td>
<td>1</td>
<td>2.5686572368</td>
</tr>
<tr>
<td>44</td>
<td>32</td>
<td>13</td>
<td>2</td>
<td>Setosa</td>
<td>1</td>
<td>6.6901420015</td>
</tr>
<tr>
<td>50</td>
<td>35</td>
<td>16</td>
<td>6</td>
<td>Setosa</td>
<td>1</td>
<td>3.8675573687</td>
</tr>
</tbody>
</table>

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:

```r
proc print noobs data=kclusOutstat1(firstobs=1 obs=3);
run;
```

```r
proc print noobs data=kclusOutstat1(firstobs=22 obs=24);
run;
```

Output 4.1.8 and Output 4.1.9 show the results.

**Output 4.1.8** Cluster Centroids before the First Iteration

**Using PROC KCLUS to Analyze Data**

<table>
<thead>
<tr>
<th>ITERATION</th>
<th><em>CLUSTER_ID</em></th>
<th>SepalLength</th>
<th>SepalWidth</th>
<th>PetalLength</th>
<th>PetalWidth</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>43</td>
<td>30</td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>65</td>
<td>28</td>
<td>46</td>
<td>15</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>67</td>
<td>30</td>
<td>50</td>
<td>17</td>
</tr>
</tbody>
</table>

**Output 4.1.9** Cluster Centroids after the Last Iteration

**Using PROC KCLUS to Analyze Data**

<table>
<thead>
<tr>
<th><em>CLUSTER_ID</em></th>
<th>SepalLength</th>
<th>SepalWidth</th>
<th>PetalLength</th>
<th>PetalWidth</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>51.9375</td>
<td>36.3125</td>
<td>14.7500</td>
<td>2.7188</td>
</tr>
<tr>
<td>7</td>
<td>63.1458</td>
<td>28.9583</td>
<td>49.7396</td>
<td>17.0313</td>
</tr>
<tr>
<td>7</td>
<td>47.3182</td>
<td>29.2727</td>
<td>17.7273</td>
<td>3.5000</td>
</tr>
</tbody>
</table>

**Example 4.2: Finding the Number of Clusters**

You can find the number of clusters in the data table by specifying NOC=ABC in the PROC KCLUS statement as follows:

```r
data mycas.iris;
  set sashelp.iris;
run;
```

```r
proc kclus data=mycas.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 4.2.1 through Output 4.2.4.

Output 4.2.1 shows the parameters that are used in the aligned box criterion (ABC) method.
Output 4.2.1 Aligned Box Criterion Parameters

Using PROC KCLUS to Analyze Data

The KCLUS Procedure

<table>
<thead>
<tr>
<th>ABC Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum Cluster</td>
</tr>
<tr>
<td>2</td>
</tr>
</tbody>
</table>

Output 4.2.2 shows the statistics that are obtained for each candidate number of clusters.

Output 4.2.2 Aligned Box Criterion Statistics

<table>
<thead>
<tr>
<th>ABC Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logarithm of Within-Cluster SSE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Number of Clusters</th>
<th>Input</th>
<th>Reference</th>
<th>Gap</th>
<th>Simulation Adjusted Standard Deviation</th>
<th>One Standard Error Adjusted Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>9.6313</td>
<td>10.1237</td>
<td>0.4923</td>
<td>0.0531</td>
<td>0.4392</td>
</tr>
<tr>
<td>3</td>
<td>8.9727</td>
<td>9.4894</td>
<td>0.5167</td>
<td>0.3678</td>
<td>0.1489</td>
</tr>
<tr>
<td>4</td>
<td>8.6811</td>
<td>8.9211</td>
<td>0.2400</td>
<td>0.0408</td>
<td>0.1992</td>
</tr>
<tr>
<td>5</td>
<td>8.5528</td>
<td>8.8012</td>
<td>0.2485</td>
<td>0.0536</td>
<td>0.1948</td>
</tr>
<tr>
<td>6</td>
<td>8.2784</td>
<td>8.4730</td>
<td>0.1945</td>
<td>0.0587</td>
<td>0.1359</td>
</tr>
<tr>
<td>7</td>
<td>8.2133</td>
<td>8.3702</td>
<td>0.1569</td>
<td>0.0517</td>
<td>0.1053</td>
</tr>
<tr>
<td>8</td>
<td>8.0750</td>
<td>8.5884</td>
<td>0.5134</td>
<td>0.0419</td>
<td>0.4715</td>
</tr>
<tr>
<td>9</td>
<td>8.0444</td>
<td>8.1240</td>
<td>0.0796</td>
<td>0.0639</td>
<td>0.0158</td>
</tr>
</tbody>
</table>

Output 4.2.3 Estimated Number of Clusters

<table>
<thead>
<tr>
<th>Estimated Number of Clusters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Clusters</td>
</tr>
<tr>
<td>---------------------</td>
</tr>
<tr>
<td>FirstPeak</td>
</tr>
</tbody>
</table>
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 4.2.4**.

**Example 4.3: Clustering Nominal Variables**

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 CAS engine libref in the first statement of the following DATA step. You can execute the following SAS code to generate the input data table, **mycas.baseball**:

```sas
Data mycas.baseball;
  Set sashelp.baseball;
  Keep Team League Division Position Div;
Run;
```

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:

```sas
proc kclus data=mycas.baseball maxiter=10 maxc=5 DISTANCENOM=RELATIVEFREQ
  outstat(out ITER)=kclusOutstat2;
  input Team League Division Position Div / level=nominal;
  score out=mycas.kclusOut2 copyvars=(Team League Division Position Div);
  ods output FreqNom=FreqNom1;
run;
```

**Output 4.3.1** shows the cluster summary table that is produced for five clusters.
Chapter 4: The KCLUS Procedure

Output 4.3.1 Cluster Summary Table for Five Clusters

Using PROC KCLUS to Analyze Data

The KCLUS Procedure

Cluster Summary for Nominal Variables
Distance from Cluster
Centroid to Observation

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Frequency</th>
<th>Minimum</th>
<th>Maximum</th>
<th>Average</th>
<th>Within Cluster Distance</th>
<th>Nearest Cluster Distance to Nearest Cluster Centroid</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>90</td>
<td>1.7111</td>
<td>2.0000</td>
<td>1.9580</td>
<td>176.2</td>
<td>2.38824</td>
</tr>
<tr>
<td>2</td>
<td>85</td>
<td>1.7059</td>
<td>2.0000</td>
<td>1.9550</td>
<td>166.2</td>
<td>2.38667</td>
</tr>
<tr>
<td>3</td>
<td>59</td>
<td>1.6271</td>
<td>2.0000</td>
<td>1.9282</td>
<td>113.8</td>
<td>4.20000</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>0.9375</td>
<td>1.6250</td>
<td>1.3867</td>
<td>22.1875</td>
<td>3.20000</td>
</tr>
<tr>
<td>5</td>
<td>72</td>
<td>1.6806</td>
<td>2.0000</td>
<td>1.9466</td>
<td>140.2</td>
<td>2.38824</td>
</tr>
</tbody>
</table>

Output 4.3.2 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 4.3.2:

```
proc print noobs data=FreqNom1(obs=12);
run;
```

Output 4.3.2 Frequencies for Nominal Variables

Using PROC KCLUS to Analyze Data

<table>
<thead>
<tr>
<th>Variable</th>
<th>Level</th>
<th>FrequencyRead</th>
<th>_1</th>
<th>_2</th>
<th>_3</th>
<th>_4</th>
<th>_5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Team</td>
<td>Atlanta</td>
<td>11 0 0 10 1 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Team</td>
<td>Baltimore</td>
<td>15 0 15 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Team</td>
<td>Boston</td>
<td>10 0 10 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Team</td>
<td>California</td>
<td>13 13 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Team</td>
<td>Chicago</td>
<td>24 13 0 0 0 11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Team</td>
<td>Cincinnati</td>
<td>12 0 0 11 1 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Team</td>
<td>Cleveland</td>
<td>12 0 12 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Team</td>
<td>Detroit</td>
<td>12 0 12 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Team</td>
<td>Houston</td>
<td>11 0 0 0 11 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Team</td>
<td>Kansas City</td>
<td>14 14 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Team</td>
<td>Los Angeles</td>
<td>14 0 0 13 1 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Team</td>
<td>Milwaukee</td>
<td>14 0 14 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>


Chapter 5
The LOGSELECT Procedure

Contents

Overview: LOGSELECT Procedure ................................................. 168
   PROC LOGSELECT Features ............................................. 169
   PROC LOGSELECT Compared with Other SAS Procedures ............. 169
   Using CAS Sessions and CAS Engine Librefs ......................... 170
Getting Started: LOGSELECT Procedure .................................. 171
   Binary Logistic Regression ............................................. 171
Syntax: LOGSELECT Procedure .................................................. 177
   PROC LOGSELECT Statement ........................................... 177
   BY Statement ............................................................. 180
   CLASS Statement ....................................................... 180
   CODE Statement ......................................................... 181
   DISPLAY Statement ..................................................... 181
   DISPLAYOUT Statement ................................................ 182
   EFFECT Statement ....................................................... 183
   FREQ Statement ........................................................ 184
   MODEL Statement ......................................................... 184
   OUTPUT Statement ....................................................... 188
   PARTITION Statement ................................................... 193
   SELECTION Statement ................................................... 193
   WEIGHT Statement ........................................................ 195
Details: LOGSELECT Procedure ................................................ 195
   Missing Values .......................................................... 195
   Response Distributions .................................................. 195
   Log-Likelihood Functions .............................................. 196
   Existence of Maximum Likelihood Estimates ......................... 198
   The LASSO Method of Model Selection ................................ 199
   Partition Fit Statistics .................................................. 200
   Model Fit and Assessment Statistics ................................ 200
   Predicted Probabilities and Regression Diagnostics ................ 202
   Joint Tests and Type 3 Tests .......................................... 204
   Multithreading ........................................................... 205
   Optimization Algorithms ............................................... 205
   Displayed Output ........................................................ 206
   ODS Table Names ........................................................ 210
Examples: LOGSELECT Procedure .............................................. 211
   Example 5.1: Model Selection ......................................... 211
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 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 \( \mu \) 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

\[
g(\pi \mid x) = \alpha + \beta'x
\]

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, \( \pi \), is the mean \( \mu \) 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, \( Y \in \{1, \ldots, k, k+1\} \)), then the LOGSELECT procedure uses the preceding link functions and fits ordinal response models of the form

\[
g(\pi_i \mid x) = \alpha_i + \beta'x, \quad i = 1, \ldots, k
\]

where \( \pi_i = \Pr(Y \leq i) \) are cumulative probabilities of the ordered response categories.

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
- 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

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 68 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.

Both the LOGSELECT and HPLOGISTIC procedures fit and perform model selection for logistic regression models. The models can contain main effects that consist of both continuous and classification variables and interaction effects of these variables. The HPLOGISTIC procedure provides GLM and reference parameterizations of classification variables; the LOGSELECT procedure provides the full set of parameterizations that are available in PROC LOGISTIC and other modeling procedures. The LOGSELECT procedure is
additionally capable of creating complex constructed effects, including univariate spline and polynomial expansions.

With the LOGSELECT and HPLOGISTIC procedures, you request model selection by using the SELECTION statement. Both procedures offer the same methods of effect selection, but the LOGSELECT procedure also provides LASSO selection.

The LOGSELECT procedure provides more regression diagnostics than the HPLOGISTIC procedure. This release of the LOGSELECT procedure is more limited in postfitting functionality than the HPLOGISTIC procedure.

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 cross-products 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, information-criterion-based selection and stopping criteria, and validation-based criteria.

The LOGISTIC procedure offers a wide variety of postfitting analyses, such as contrasts, estimates, tests of model effects, least squares means, and odds ratios. This release of the LOGSELECT procedure is limited in postfitting functionality, because for large data sets the focus is primarily on model fitting and scoring.

The LOGSELECT procedure is specifically 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:

```plaintext
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 mycas cas sessref=mysess;

The CAS statement creates the CAS session named mysess, and the LIBNAME statement creates the mycas 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 8 in Chapter 2, “Shared Concepts.”

---

**Binary Logistic Regression**

**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 8 and “Loading a SAS Data Set onto a CAS Server” on page 9 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`), in your CAS session:

```sas
data mycas.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
```

---

*Getting Started: LOGSELECT Procedure*
| 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 | 2.2 |
| 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 1 | 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 |
These statements assume that your CAS engine libref is named mycas, but you can substitute any appropriately defined 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.

```plaintext
proc logselect data=mycas.getStarted ithist;
  class C;
  model y = C x1-x10;
run;
```

The output from this analysis is presented in Figure 5.1 through Figure 5.10.

Figure 5.1 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.
The LOGSELECT Procedure

**Figure 5.1** Model Information

<table>
<thead>
<tr>
<th>Model Information</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Source</td>
<td>GETSTARTED</td>
</tr>
<tr>
<td>Response Variable</td>
<td>y</td>
</tr>
<tr>
<td>Distribution</td>
<td>Binary</td>
</tr>
<tr>
<td>Link Function</td>
<td>Logit</td>
</tr>
<tr>
<td>Optimization Technique</td>
<td>Newton-Raphson with Ridging</td>
</tr>
</tbody>
</table>

**Figure 5.2** Number of Observations

<table>
<thead>
<tr>
<th>Number of Observations Read</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Observations Used</td>
<td>100</td>
</tr>
</tbody>
</table>

The “Response Profile” table in **Figure 5.3** 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-ordered 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 5.3** Response Profile

<table>
<thead>
<tr>
<th>Response Profile</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Ordered Value</td>
<td>Total Frequency</td>
</tr>
<tr>
<td>y</td>
<td>69</td>
</tr>
<tr>
<td>1</td>
<td>31</td>
</tr>
</tbody>
</table>

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 5.4**.

**Figure 5.4** Class Level Information

<table>
<thead>
<tr>
<th>Class Level Information</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Class Levels Values</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>10 A B C D E F G H I J</td>
</tr>
</tbody>
</table>

The “Iteration History” table is shown in **Figure 5.5**. The Newton-Raphson algorithm with ridging converged after four iterations, not counting the initial setup iteration.
Figure 5.6 displays the final convergence status of the Newton-Raphson algorithm. The GCONV= relative convergence criterion is satisfied.

**Figure 5.6** Convergence Status

| Convergence criterion (GCONV=1E-8) satisfied. |

Figure 5.7 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 $x_1$–$x_{10}$. 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 5.7** Dimensions in Binomial Logistic Regression

<table>
<thead>
<tr>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Columns in Design</td>
</tr>
<tr>
<td>Number of Effects</td>
</tr>
<tr>
<td>Max Effect Columns</td>
</tr>
<tr>
<td>Rank of Design</td>
</tr>
<tr>
<td>Parameters in Optimization</td>
</tr>
</tbody>
</table>

Figure 5.8 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 5.8** Null Test

<table>
<thead>
<tr>
<th>Testing Global Null Hypothesis: BETA=0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test</td>
</tr>
<tr>
<td>Likelihood Ratio</td>
</tr>
<tr>
<td>DF</td>
</tr>
<tr>
<td>Chi-Square</td>
</tr>
<tr>
<td>Pr &gt; ChiSq</td>
</tr>
</tbody>
</table>

The “Fit Statistics” table is shown in Figure 5.9. 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).
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 5.9** Fit Statistics

<table>
<thead>
<tr>
<th>Fit Statistics</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(-2) Log Likelihood</td>
<td>88.70072</td>
</tr>
<tr>
<td>AIC (smaller is better)</td>
<td>128.70072</td>
</tr>
<tr>
<td>AICC (smaller is better)</td>
<td>139.33363</td>
</tr>
<tr>
<td>SBC (smaller is better)</td>
<td>180.80412</td>
</tr>
</tbody>
</table>

However, the “Parameter Estimates” table in **Figure 5.10** shows that many parameters have fairly large \(p\)-values, indicating that one or more of the model effects might not be necessary.

**Figure 5.10** Parameter Estimates

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Estimate</th>
<th>Standard Error</th>
<th>Chi-Square</th>
<th>Pr &gt; ChiSq</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intercept</td>
<td>1.210139</td>
<td>1.750750</td>
<td>0.4778</td>
<td>0.4894</td>
</tr>
<tr>
<td>C A</td>
<td>3.434071</td>
<td>1.613142</td>
<td>4.5318</td>
<td>0.0333</td>
</tr>
<tr>
<td>C B</td>
<td>2.163830</td>
<td>1.427083</td>
<td>2.2990</td>
<td>0.1295</td>
</tr>
<tr>
<td>C C</td>
<td>0.655205</td>
<td>1.081024</td>
<td>0.3674</td>
<td>0.5445</td>
</tr>
<tr>
<td>C D</td>
<td>2.494481</td>
<td>1.109366</td>
<td>5.0560</td>
<td>0.0245</td>
</tr>
<tr>
<td>C E</td>
<td>3.244860</td>
<td>1.432140</td>
<td>5.1336</td>
<td>0.0235</td>
</tr>
<tr>
<td>C F</td>
<td>3.605371</td>
<td>1.307025</td>
<td>7.6091</td>
<td>0.0058</td>
</tr>
<tr>
<td>C G</td>
<td>2.084085</td>
<td>1.189764</td>
<td>3.0684</td>
<td>0.0798</td>
</tr>
<tr>
<td>C H</td>
<td>2.936779</td>
<td>1.293876</td>
<td>5.1518</td>
<td>0.0232</td>
</tr>
<tr>
<td>C I</td>
<td>1.378522</td>
<td>1.031899</td>
<td>1.7847</td>
<td>0.1816</td>
</tr>
<tr>
<td>C J</td>
<td>0.0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(x1)</td>
<td>0.032181</td>
<td>0.057100</td>
<td>0.3176</td>
<td>0.5730</td>
</tr>
<tr>
<td>(x2)</td>
<td>-0.367741</td>
<td>0.153824</td>
<td>5.7152</td>
<td>0.0168</td>
</tr>
<tr>
<td>(x3)</td>
<td>0.314640</td>
<td>0.357414</td>
<td>0.7750</td>
<td>0.3787</td>
</tr>
<tr>
<td>(x4)</td>
<td>-0.051957</td>
<td>0.024428</td>
<td>4.5239</td>
<td>0.0334</td>
</tr>
<tr>
<td>(x5)</td>
<td>-0.006828</td>
<td>0.010555</td>
<td>0.4184</td>
<td>0.5177</td>
</tr>
<tr>
<td>(x6)</td>
<td>0.253852</td>
<td>0.378462</td>
<td>0.4499</td>
<td>0.5024</td>
</tr>
<tr>
<td>(x7)</td>
<td>-0.007233</td>
<td>0.010734</td>
<td>0.4541</td>
<td>0.5004</td>
</tr>
<tr>
<td>(x8)</td>
<td>2.537042</td>
<td>0.994180</td>
<td>6.5122</td>
<td>0.0107</td>
</tr>
<tr>
<td>(x9)</td>
<td>-0.167502</td>
<td>0.106789</td>
<td>2.4603</td>
<td>0.1168</td>
</tr>
<tr>
<td>(x10)</td>
<td>-0.222155</td>
<td>0.157743</td>
<td>1.9834</td>
<td>0.1590</td>
</tr>
</tbody>
</table>

Finally, the procedure displays the table in **Figure 5.11**, which shows the amount of time (in seconds) that PROC LOGSELECT required to perform different tasks in the analysis.
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<(response-options)> = <effects> </model-options> ;
    OUTPUT OUT=CAS-libref.data-table <options> <keyword =name> . . . <keyword =name> > ;
    PARTITION partition-options ;
    SELECTION <METHOD=method<(method-options)> > </options> ;
    WEIGHT variable ;
```

The PROC LOGSELECT statement and at least one MODEL statement are required. The CLASS statement can appear multiple times. If a CLASS statement is specified, it must precede the MODEL statements.

---

**Figure 5.11** Procedure Timing

<table>
<thead>
<tr>
<th>Task</th>
<th>Seconds</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setup and Parsing</td>
<td>0.02</td>
<td>13.96%</td>
</tr>
<tr>
<td>Levelization</td>
<td>0.04</td>
<td>33.19%</td>
</tr>
<tr>
<td>Model Initialization</td>
<td>0.00</td>
<td>2.10%</td>
</tr>
<tr>
<td>SSCP Computation</td>
<td>0.00</td>
<td>3.24%</td>
</tr>
<tr>
<td>Model Fitting</td>
<td>0.04</td>
<td>38.60%</td>
</tr>
<tr>
<td>Cleanup</td>
<td>0.00</td>
<td>3.23%</td>
</tr>
<tr>
<td>Total</td>
<td>0.11</td>
<td>100.00%</td>
</tr>
</tbody>
</table>
Table 5.1 PROC LOGSELECT Statement Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALPHA=</td>
<td>Specifies a global significance level</td>
</tr>
<tr>
<td>DATA=</td>
<td>Specifies the CAS input data table</td>
</tr>
<tr>
<td>MAXOPTBATCH=</td>
<td>Specifies the maximum number of observations to be computed in each batch</td>
</tr>
</tbody>
</table>

Output Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CORRB</td>
<td>Displays the “Parameter Estimates Correlation Matrix” table</td>
</tr>
<tr>
<td>COVB</td>
<td>Displays the “Parameter Estimates Covariance Matrix” table</td>
</tr>
<tr>
<td>ITHIST</td>
<td>Displays the “Iteration History” table</td>
</tr>
<tr>
<td>NOCHECK</td>
<td>Disables checking for infinite parameters</td>
</tr>
<tr>
<td>NOCLPRINT</td>
<td>Limits or suppresses the display of class levels</td>
</tr>
<tr>
<td>NOSTDERR</td>
<td>Suppresses computation of the covariance matrix and standard errors</td>
</tr>
<tr>
<td>PARTFIT</td>
<td>Displays the fit statistics that are produced when your data are partitioned</td>
</tr>
</tbody>
</table>

Optimization Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSCONV=</td>
<td>Tunes the absolute function convergence criterion</td>
</tr>
<tr>
<td>ABSFCONV=</td>
<td>Tunes the absolute function difference convergence criterion</td>
</tr>
<tr>
<td>ABSGCONV=</td>
<td>Tunes the absolute gradient convergence criterion</td>
</tr>
<tr>
<td>ABSXCONV=</td>
<td>Tunes the absolute parameter convergence criterion</td>
</tr>
<tr>
<td>FCONV=</td>
<td>Tunes the relative function difference convergence criterion</td>
</tr>
<tr>
<td>FCONV2=</td>
<td>Tunes the second relative function difference convergence criterion</td>
</tr>
<tr>
<td>GCONV=</td>
<td>Tunes the relative gradient convergence criterion</td>
</tr>
<tr>
<td>GCONV2=</td>
<td>Tunes the second relative gradient convergence criterion</td>
</tr>
<tr>
<td>MAXFUNC=</td>
<td>Specifies the maximum number of function evaluations in any optimization</td>
</tr>
<tr>
<td>MAXITER=</td>
<td>Specifies the maximum number of iterations in any optimization</td>
</tr>
<tr>
<td>MAXTIME=</td>
<td>Specifies the upper limit of CPU time (in seconds) for any optimization</td>
</tr>
<tr>
<td>MINITER=</td>
<td>Specifies the minimum number of iterations in any optimization</td>
</tr>
<tr>
<td>NORMALIZE=</td>
<td>Specifies whether the objective function is normalized during optimization</td>
</tr>
<tr>
<td>TECHNIQUE=</td>
<td>Selects the optimization technique</td>
</tr>
</tbody>
</table>

LASSO Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LASSORHO=</td>
<td>Specifies the base regularization parameter for the LASSO method</td>
</tr>
<tr>
<td>LASSOSTEPS=</td>
<td>Specifies the maximum number of steps for the LASSO method</td>
</tr>
<tr>
<td>LASSOTOL=</td>
<td>Specifies the convergence criterion for the LASSO method</td>
</tr>
</tbody>
</table>

The optimization options are fully described in the section “Optimization Options” on page 38 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.
**MAXOPTBATCH=number**

specifies the maximum number of observations to be included in a batch. 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 MAXOPTBATCH value decreases memory usage but might lead to longer computation times, whereas a larger MAXOPTBATCH value might lead to shorter computation times but increases memory usage. By default, MAXOPTBATCH=256.

**CORRB**

creates the “Parameter Estimates Correlation Matrix” table. The correlation matrix is computed by normalizing the covariance matrix $\Sigma$. That is, if $\sigma_{ij}$ is an element of $\Sigma$, then the corresponding element of the correlation matrix is $\rho_{ij} = \frac{\sigma_{ij}}{\sigma_i \sigma_j}$, 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 of the matrix of second derivatives of the log-likelihood function with respect to the model parameters (the Hessian matrix).

**DATA=CAS-libref.data-table**

names the input data table for PROC LOGSELECT to use. The default is the most recently created data table. **CAS-libref.data-table** is a two-level name, where

- **CAS-libref** refers to a collection of information that is defined in the LIBNAME statement and includes the caslib, 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 **CAS-libref**, see the section “Using CAS Sessions and CAS Engine Librefs” on page 170.
- **data-table** specifies the name of the input data table.

**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.

**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 198.
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 198.

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.

**BY Statement**

```sas
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 *SAS Language Reference: Concepts*.

**CLASS Statement**

```sas
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 5.2 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 10 in Chapter 2, “Shared Concepts.”*

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DESCENDING</td>
<td>Reverses the sort order</td>
</tr>
<tr>
<td>MISSING</td>
<td>Treats missing values as valid levels</td>
</tr>
<tr>
<td>ORDER=</td>
<td>Specifies the sort order for the levels</td>
</tr>
<tr>
<td>PARAM=</td>
<td>Specifies the parameterization of the variable</td>
</tr>
</tbody>
</table>
**CODE Statement**

**CODE** <options> ;

The CODE statement writes SAS DATA step code for computing predicted values of the fitted model either to a file or to a catalog entry. This code can then be included in a DATA step to score new data.

Table 5.3 summarizes the options available in the CODE statement.

Table 5.3 CODE Statement Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REF=</td>
<td>Specifies the reference level of the variable</td>
</tr>
<tr>
<td>SPLIT</td>
<td>Allows design columns for a variable to enter or leave the model independently</td>
</tr>
<tr>
<td>COMMENT</td>
<td>Adds comments to the generated code</td>
</tr>
<tr>
<td>FILE=</td>
<td>Names the file where the generated code is saved</td>
</tr>
<tr>
<td>FORMATWIDTH=</td>
<td>Specifies the numeric format width for the regression coefficients</td>
</tr>
<tr>
<td>INDENTSIZE=</td>
<td>Specifies the number of spaces to indent the generated code</td>
</tr>
<tr>
<td>LABELID=</td>
<td>Specifies a number used to construct names and labels</td>
</tr>
<tr>
<td>LINESIZE=</td>
<td>Specifies the line size for the generated code</td>
</tr>
<tr>
<td>NOTRIM</td>
<td>Compares formatted values, including blank padding</td>
</tr>
<tr>
<td>PCATALL</td>
<td>Generates probabilities for all levels of categorical response variables</td>
</tr>
</tbody>
</table>

For more information about the syntax of the CODE statement, see the section “CODE Statement” on page 14 in Chapter 2, “Shared Concepts.”

**DISPLAY Statement**

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

The DISPLAY statement enables you to specify a list of ODS 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, all ODS tables are sent to the client and then the client creates a subset. If both DISPLAY and ODS statements are used together, the DISPLAY statement takes precedence over the ODS statements. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

You can specify the following options after a slash (/):
CASESENSITIVE
performs a case-sensitive comparison of table names in the table-list to ODS 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 ODS tables except those specified in the table-list.

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

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

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

A path is a table name that is prefixed with dot-separated grouping information. For example, a SelectionSummary table that is produced by a procedure 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 ODS 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 starts with a “/” or a “!”. 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.

DISPLAYOUT Statement

DISPLAYOUT table-spec-list </options> ;

The DISPLAYOUT statement enables you to create CAS 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 CAS output tables to create. Each entry in the list has either a key or a key=value format:

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

key specifies key as the ODS table name and also as the CAS output table name.

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 (/):
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 48 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 5.4 summarizes the options available in the EFFECT statement.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Collection Effects Options</strong></td>
<td></td>
</tr>
<tr>
<td>DETAILS</td>
<td>Displays the constituents of the collection effect</td>
</tr>
<tr>
<td><strong>Multimember Effects Options</strong></td>
<td></td>
</tr>
<tr>
<td>DETAILS</td>
<td>Displays the levels of the multimember effect</td>
</tr>
<tr>
<td>NOEFFECT</td>
<td>Specifies that observations whose levels are all missing for the multimember variables should have 0 values in the corresponding design matrix columns</td>
</tr>
<tr>
<td>STDIZE</td>
<td>Standardizes the design matrix entries so that each observation has a sum of 1</td>
</tr>
<tr>
<td>WEIGHT=</td>
<td>Specifies the weight variable for the contributions of each classification effect</td>
</tr>
<tr>
<td><strong>Polynomial Effects Options</strong></td>
<td></td>
</tr>
<tr>
<td>DEGREE=</td>
<td>Specifies the degree of the polynomial</td>
</tr>
</tbody>
</table>
### Table 5.4  continued

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

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 17 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 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 the FREQ statement is not specified, each observation is assigned a frequency of 1.

---

### 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 binary 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 45 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. The model-options control other aspects of model formation and inference. Table 5.5 summarizes these options.

### Table 5.5  MODEL Statement Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Response Variable Options</strong></td>
<td></td>
</tr>
<tr>
<td>DESCENDING</td>
<td>Reverses the response categories</td>
</tr>
<tr>
<td>EVENT=</td>
<td>Specifies the event category</td>
</tr>
<tr>
<td>ORDER=</td>
<td>Specifies the sort order</td>
</tr>
<tr>
<td>REF=</td>
<td>Specifies the reference category</td>
</tr>
<tr>
<td><strong>Model Options</strong></td>
<td></td>
</tr>
<tr>
<td>CLB</td>
<td>Requests confidence limits</td>
</tr>
<tr>
<td>INCLUDE=</td>
<td>Includes effects in all models for model selection</td>
</tr>
<tr>
<td>INFORMATIVE</td>
<td>Models missing values by using extra indicator variables</td>
</tr>
<tr>
<td>LINK=</td>
<td>Specifies the link function</td>
</tr>
<tr>
<td>NOINT</td>
<td>Suppresses the intercept</td>
</tr>
<tr>
<td>OFFSET=</td>
<td>Specifies the offset variable</td>
</tr>
<tr>
<td>START=</td>
<td>Includes effects in the initial model for model selection</td>
</tr>
<tr>
<td>TYPE3</td>
<td>Displays the Type 3 or joint tests of effects</td>
</tr>
</tbody>
</table>

### Response Variable Options

Response variable options determine how the LOGSELECT procedure models probabilities for binary and multinomial response 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 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’.

```sas
proc logselect data=mycas.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. When ORDER=FORMATTED (the default) 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 the ORDER= option is interpreted:

<table>
<thead>
<tr>
<th>ORDER=</th>
<th>Levels Sorted By</th>
</tr>
</thead>
<tbody>
<tr>
<td>FORMATTED</td>
<td>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.</td>
</tr>
<tr>
<td>FREQ</td>
<td>Descending frequency count (levels with the most observations come first in the order)</td>
</tr>
<tr>
<td>INTERNAL</td>
<td>Unformatted value; the sort order is machine-dependent.</td>
</tr>
</tbody>
</table>

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 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. 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**

**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.

**INCLUDE=**
forces effects to be included in all models. If you specify INCLUDE=, then the first effects that are listed in the MODEL statement are included in all models. If you specify INCLUDE= or if you specify INCLUDE=, 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 (/).

**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 65 in Chapter 2, “Shared Concepts.”

**LINK=**
specifies the link function for the model. The default link is the logit. The keywords and the associated link functions are shown in Table 5.6.

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Link Function</th>
<th>( g(\pi) = \eta = )</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLOGLOG</td>
<td>Complementary log-log</td>
<td>( \log(-\log(1 - \pi)) )</td>
</tr>
<tr>
<td>LOGIT</td>
<td>Logit</td>
<td>( \log(\pi/(1 - \pi)) )</td>
</tr>
<tr>
<td>LOGLOG</td>
<td>Log-log</td>
<td>( -\log(-\log(\pi)) )</td>
</tr>
<tr>
<td>PROBIT</td>
<td>Probit</td>
<td>( \Phi^{-1}(\pi) )</td>
</tr>
</tbody>
</table>

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.

**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.

**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.

**TYPE3**
requests that Wald statistics for Type 3 contrasts be computed for each effect that is specified in the MODEL statement. For more information, see the section “Joint Tests and Type 3 Tests” on page 204.

### OUTPUT Statement

```
OUTPUT OUT=CAS-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 the response variable has more than two categories, you can request the “Statistic Options” listed in Table 5.7; 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. If you specify the PREDPROBS option, then only the PRED and IPRED variables 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.

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 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 `caslib`, 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 170.

- `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.

**OBSCAT**

requests (for multinomial models) that observationwise statistics be produced only for the observed response level. If you do not specify the OBSCAT option and the ordinal response variable has $J$ levels, then $J-1$ records are output for every observation in the input data that corresponds to the $J-1$ lower-ordered response categories.

**PREDPROBS**

creates variables (for multinomial models) for each response category that corresponds to the requested `IPRED` and `PREDICTED` keywords.

**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.

Table 5.7 summarizes the `keywords` available in the OUTPUT statement.
Table 5.7  OUTPUT Statement Keywords

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
<th>Default Names</th>
</tr>
</thead>
<tbody>
<tr>
<td>Statistic Options</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INDIVIDUAL</td>
<td>Specifies the individual predicted probabilities</td>
<td><em>IPRED</em></td>
</tr>
<tr>
<td>LCL</td>
<td>Specifies the lower confidence limit for the linear predictor</td>
<td><em>LCL</em></td>
</tr>
<tr>
<td>LCLM</td>
<td>Specifies the lower confidence limit for the event probability</td>
<td><em>LCLM</em></td>
</tr>
<tr>
<td>PREDICTED</td>
<td>Specifies the predicted probabilities</td>
<td><em>PRED</em></td>
</tr>
<tr>
<td>RESRAW</td>
<td>Specifies the raw residual</td>
<td><em>RESRAW</em></td>
</tr>
<tr>
<td>STDERRBETA</td>
<td>Specifies the standard error estimate of the linear predictor</td>
<td><em>STDXBETA</em></td>
</tr>
<tr>
<td>UCL</td>
<td>Specifies the upper confidence limit for the linear predictor</td>
<td><em>UCL</em></td>
</tr>
<tr>
<td>UCLM</td>
<td>Specifies the upper confidence limit for the event probability</td>
<td><em>UCLM</em></td>
</tr>
<tr>
<td>XBETA</td>
<td>Specifies the linear predictor</td>
<td><em>XBETA</em></td>
</tr>
<tr>
<td>Diagnostic Options for Binary and Binomial Response Data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CBAR</td>
<td>Specifies the confidence interval displacement</td>
<td><em>CBAR</em></td>
</tr>
<tr>
<td>DIFCHISQ</td>
<td>Specifies the deletion chi-square goodness-of-fit change</td>
<td><em>DIFCHISQUARE</em></td>
</tr>
<tr>
<td>DIFDEV</td>
<td>Specifies the deletion deviance change</td>
<td><em>DIFDEVIANCE</em></td>
</tr>
<tr>
<td>H</td>
<td>Specifies the leverage</td>
<td><em>HATDIAG</em></td>
</tr>
<tr>
<td>RESCHI</td>
<td>Specifies the Pearson chi-square residual</td>
<td><em>RESCHI</em></td>
</tr>
<tr>
<td>RESDEV</td>
<td>Specifies the deviance residual</td>
<td><em>RESDEV</em></td>
</tr>
<tr>
<td>RESLIK</td>
<td>Specifies the likelihood residual</td>
<td><em>RESLIK</em></td>
</tr>
<tr>
<td>RESWORK</td>
<td>Specifies the working residual</td>
<td><em>RESWORK</em></td>
</tr>
<tr>
<td>STDRESCHI</td>
<td>Specifies the standardized Pearson chi-square residual</td>
<td><em>STDRESCHI</em></td>
</tr>
<tr>
<td>STDRESDEV</td>
<td>Specifies the standardized deviance residual</td>
<td><em>STDRESDEV</em></td>
</tr>
</tbody>
</table>

The following list describes these *keywords*. For more information, see the section “Predicted Probabilities and Regression Diagnostics” on page 202.

**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_.

**H**

specifies the diagonal element of the hat matrix (leverage) for detecting extreme points in the design space. The default name is _HATDIAG_.

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_.

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_.

PREDICTED
PRED
PROB
P
specifies the predicted values (predicted probabilities of events) for a binary response variable 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, \( \Pr(Y = 3) = 1 \), 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_.

RESLIK

specifies the likelihood residual for identifying poorly fitted observations. The default name is _RESLIK_.

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 5.8 shows how this variable is interpreted for each observation.

<table>
<thead>
<tr>
<th>Value</th>
<th>Observation Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not used</td>
</tr>
<tr>
<td>1</td>
<td>Training</td>
</tr>
<tr>
<td>2</td>
<td>Validation</td>
</tr>
<tr>
<td>3</td>
<td>Testing</td>
</tr>
</tbody>
</table>

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_.

STDXBETA specifies the standard error estimates of XBETA. The default name is _STDXBETA_.

UCL 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 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 specifies the linear predictor. The default name is _XBETA_.

LINP
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 66 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 of the observations 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 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 200. For an illustration, see Example 5.3.

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 32 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.
Chapter 5: The LOGSELECT Procedure

**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 63 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 HPLOGISTIC 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) computed on the validation data as the criterion (not available for the SELECT= option).

For more information, see the section “Information Criteria” on page 200. 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.

If you specify METHOD=LASSO and you 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.

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. 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.

The LOGSELECT procedure does not support the `ORDERSELECT` option in this release.

---

**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, all observations that are used in the analysis are assigned a weight of 1.

---

**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 65 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
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

$$
\begin{align*}
\Pr(Y = 0) &= p \\
\Pr(Y = 1) &= 1 - p
\end{align*}
$$

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 $i$th 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 $x_1$ and $x_2$.

```plaintext
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.

Whenever the LOGSELECT procedure determines that the response variable 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.

---

### Log-Likelihood Functions

The LOGSELECT procedure forms the log-likelihood functions of the various models as

$$
L(\mu; y) = \sum_{i=1}^{n} f_i l(\mu_i; y_i, w_i)
$$

where $\mu_i$ is the probability that the $i$th observation is an event, $l(\mu_i; y_i, w_i)$ is the log-likelihood contribution of the $i$th 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 \( i \)th binary observation as

\[
\eta_i = \mathbf{x}_i' \boldsymbol{\beta} \\
\mu_i(\boldsymbol{\beta}) = g^{-1}(\eta_i) \\
l(\mu_i(\boldsymbol{\beta}); y_i) = y_i \log\{\mu_i\} + (1 - y_i) \log\{1 - \mu_i\}
\]

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 \( \eta_i \) 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 \( i \)th binomial observation as

\[
\eta_i = \mathbf{x}_i' \boldsymbol{\beta} \\
\mu_i(\boldsymbol{\beta}) = g^{-1}(\eta_i) \\
l(\mu_i(\boldsymbol{\beta}); y_i, w_i) = w_i (y_i \log\{\mu_i\} + (n_i - y_i) \log\{1 - \mu_i\}) \\
+ w_i \log\left(\frac{n_i}{y_i}\right)
\]

where \( y_i \) and \( n_i \) are the values of the events and trials of the \( i \)th observation, respectively. The value \( \mu_i \) 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 \( i \)th observation is

\[
l(\mu_i; y_i, w_i) = w_i \sum_{j=1}^{J} y_{ij} \log\{\mu_{ij}\}
\]
In this expression, $J$ denotes the number of response categories (the number of possible outcomes) and $\mu_{ij}$ is the probability that the $i$th observation takes on the response value associated with category $j$. 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.

**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 $j$th subject, and let $x_j$ 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 $b$ that correctly allocates all observations to their response groups; that is,

\[
\begin{align*}
\{ & b'x_j > 0 \quad Y_j = 0 \\
\{ & b'x_j < 0 \quad Y_j = 1
\end{align*}
\]

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 $b$ such that

\[
\begin{align*}
\{ & b'x_j \geq 0 \quad Y_j = 0 \\
\{ & b'x_j \leq 0 \quad Y_j = 1
\end{align*}
\]

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

LASSO Selection

The LOGSELECT procedure implements the group LASSO method, which is described in the section “Group LASSO Selection” on page 63 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 $X = (x_1, x_2, \ldots, x_m)$ denote the matrix of covariates, and let $y$ denote the response. Then for a given parameter $t$, the LASSO regression coefficients $\beta = (\beta_1, \beta_2, \ldots, \beta_m)$ are the solution to the following constrained least squares problem:

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

For generalized linear models, the LASSO regression coefficients $\beta = (\beta_1, \beta_2, \ldots, \beta_m)$ are the solution to the following constrained optimization problem,

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

where $L$ is the log-likelihood function defined in the section “Log-Likelihood Functions” on page 196.

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).
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 200.

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:

\[
\begin{align*}
\text{AIC} &= -2l + 2p \\
\text{AICC} &= \begin{cases} 
-2l + 2pF/(F - p - 1) & \text{when } F > p + 2 \\
-2l + 2p(p + 2) & \text{otherwise}
\end{cases} \\
\text{SBC} &= -2l + p \log(F)
\end{align*}
\]

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:

\[R^2 = 1 - \left\{ \frac{L(0)}{L(\hat{\beta})} \right\}^{\frac{2}{n}}\]

Here, $L(0)$ is the likelihood of the intercept-only model, $L(\hat{\beta})$ 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, \( L(\hat{\beta}) \) approaches \( L(0) \) 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:

\[
R^2_{\text{max}} = 1 - \{L(0)\}_1^n
\]

\[
R^2_N = \frac{R^2}{R^2_{\text{max}}}
\]

Another measure, from McFadden (1974), is also bounded by 0 and 1:

\[
R^2_M = 1 - \left( \frac{\log L(\hat{\beta})}{\log L(0)} \right)
\]

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 5.9 (Brier 1950; Murphy 1973); it is also called the Brier score or Brier reliability.

<table>
<thead>
<tr>
<th>Response Type</th>
<th>ASE (Brier Score)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary</td>
<td>( \frac{1}{F} \sum f_i (y_i (1 - \hat{\pi}_i)^2 + (1 - y_i)(\hat{\pi}_i)^2) )</td>
</tr>
<tr>
<td>Binomial</td>
<td>( \frac{1}{F} \sum f_i (r_i / t_i - \hat{\pi}_i)^2 )</td>
</tr>
<tr>
<td>Multinomial</td>
<td>( \frac{1}{F} \sum f_i \sum_j (y_{ij} - \hat{\pi}_{ij})^2 )</td>
</tr>
</tbody>
</table>

In Table 5.9, \( F = \sum f_i \), \( r_i \) 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 \( \hat{\pi}_i \) is the predicted probability of an event. For polytomous response models, \( y_{ij} = 1 \) if the \( i \)th observation has response level \( j \), and \( \pi_{ij} \) is the model-predicted probability of response level \( j \) for observation \( i \).

### Difference of Means

For a binary response model, write the mean of the model-predicted probabilities of event (\( Y=0 \)) observations as \( \overline{X}_1 = \frac{\sum_{i=1}^n (\hat{\pi}_i | y_i = 0)}{n_0} \) and of nonevent (\( Y=1 \)) observations as \( \overline{X}_2 = \frac{\sum_{i=1}^n (\hat{\pi}_i | y_i = 1)}{n_0} \), where \( \hat{\pi}_i \) 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 \( \Delta m \) statistic (with unit standard error) that is discussed in the signal detection literature (McNicol 2005).
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.

For a binary response model, given a vector of covariates \( x_i \) for the \( i \)th observation in your data table and the model-predicted parameter estimates \( \hat{\beta} \), you can write the linear predictor \( \hat{\eta}_i = x_i'\hat{\beta} \). The mean of the \( i \)th observation \( \mu_i(\hat{\beta}) \), or the model-predicted event probability \( \hat{\pi}_i = \pi = g^{-1}(\eta_i) \), where the link function \( g \) is chosen 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 \( \Sigma \) is the estimated covariance of \( \hat{\beta} \). Denote the frequency of the \( i \)th observation as \( f_i \) and the weight as \( w_i \).

For multinomial 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 \( \beta \) consist of the slope parameters: \( \hat{\eta}_{ij} = g(\Pr(Y \leq j \mid x_i) = \hat{\alpha}_j + x_i'\hat{\beta} \) and \( \hat{\pi}_{ij} = \Pr(Y \leq j \mid x_i) = g^{-1}(\eta_{ij}) \) for \( 1 \leq j < J \).

Confidence Intervals

Approximate confidence intervals for predicted probabilities can be computed as follows. The variance of the linear predictor is estimated by

\[
\hat{\sigma}^2(\eta_i) = x_i'\Sigma x_i
\]

For multinomial models, the variance also depends on the response function. Let \( \delta_j \) be a \((J-1)\) column vector whose \( j \)th entry is equal to 1 and all other entries are equal to 0. Redefine \( x_i = (\delta_j', x_i')' \), \( \eta_i = \eta_{ij} \), and \( \pi_i = \pi_{ij} \). Then

\[
\hat{\sigma}^2(\eta_i) = x_i'\Sigma x_i
\]

The asymptotic 100(1 - \( \alpha \))% confidence interval for \( \eta_i \) 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 \( \pi_i \) 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

\[
V(\pi_i) = \frac{V(\pi_i)}{f_i w_i}
\]
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 $\pi_i$. The weight matrix $W_e$ 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

$$W_e^{1/2}X(X'W_eX)^{-1}X'W_e^{1/2}$$

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).

**Residuals**

Residuals are useful in identifying observations that are not explained well by the model. For binary and binomial response data, the raw residual is

$$r_i = y_i/t_i - \hat{\pi}_i$$

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 residual is

$$y_{ij} - \hat{\pi}_{ij}$$

where $y_{ij} = 1$ if the $i$th 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 $i$.

The Pearson residual is the square root of the $i$th observation’s contribution to Pearson’s chi-square:

$$r_{pi} = r_i \sqrt{\frac{f_i w_i}{V(\pi_i)}}$$

The deviance residual is the square root of the contribution of the $i$th observation to the deviance, with the sign of the raw residual,

$$r_{di} = (\text{sign}(r_i)) \sqrt{d_i}$$

where

$$d_i = 2f_i w_i t_i \left[ \frac{y_i/t_i}{\pi_i} \log \left( \frac{y_i/t_i}{\pi_i} \right) + (1 - y_i/t_i) \log \left( \frac{1 - y_i/t_i}{1 - \pi_i} \right) \right]$$

The working residual is

$$r_{wi} = r_i \left( \frac{\partial \pi_i}{\partial \eta_i} \right)^{-1}$$
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 \( \hat{\beta} \). 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:

\[ C_i = r_{pi}^2 h_i / (1 - h_i) \]

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 \( i \)th observation are computed as follows:

\[ \text{DIFDEV}_i = r_{Di}^2 + C_i \]
\[ \text{DIFCHISQ}_i = C_i / h_i \]

**Joint Tests and Type 3 Tests**

Linear hypotheses for \( \beta \) are expressed in matrix form as

\[ H_0: L\beta = c \]

where \( L \) is a matrix of coefficients for the linear hypotheses and \( c \) is a vector of constants. The vector of regression coefficients \( \beta \) includes slope parameters and intercept parameters. The Wald chi-square statistic for testing \( H_0 \) is computed as

\[ \chi^2_W = (L\hat{\beta} - c)'[L\hat{V}(\hat{\beta})L']^{-1}(L\hat{\beta} - c) \]

where \( \hat{V}(\hat{\beta}) \) is the estimated covariance matrix. Under \( H_0 \), \( \chi^2_W \) has an asymptotic chi-square distribution with \( r \) degrees of freedom, where \( r \) is the rank of \( L \).
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 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 68 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 5.10 summarizes the optimization techniques available in PROC LOGSELECT.
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 68 in Chapter 2, “Shared Concepts,” is helpful in choosing a suitable optimization algorithm.

### 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.

#### 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 195.

#### 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 a WEIGHT statement is specified, 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 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

---

**Table 5.10** Optimization Techniques

<table>
<thead>
<tr>
<th>TECHNIQUE=</th>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUREG</td>
<td>Trust region method</td>
</tr>
<tr>
<td>NEWRAP</td>
<td>Newton-Raphson method with line search</td>
</tr>
<tr>
<td>NRRIDG</td>
<td>Newton-Raphson method with ridging</td>
</tr>
<tr>
<td>QUANEW</td>
<td>Quasi-Newton methods</td>
</tr>
<tr>
<td>DBLDOG</td>
<td>Double-dogleg method</td>
</tr>
<tr>
<td>CONGRA</td>
<td>Conjugate gradient methods</td>
</tr>
<tr>
<td>NMSIMP</td>
<td>Nelder-Mead simplex method</td>
</tr>
</tbody>
</table>
binary model, the note that follows the “Response Profile” table indicates which outcome is modeled as the event. For a multinomial model, the note that follows the “Response Profile” table indicates how the ordered response levels are accumulated. 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 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 DETAILS=NONE 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 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 LOGSELECT statement.
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 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 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 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).
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 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 200.

**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.

**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.

**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.

**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 equal to zero. For more information, see the section “Joint Tests and Type 3 Tests” on page 204.

**Score Code Variables for Predicted Probability**

When you specify the PCATALL option in the CODE statement, the LOGSELECT 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 LOGSELECT required to perform different tasks in the analysis.
Output CAS Tables

When you specify an OUTPUT statement or a DISPLAYOUT statement to create output tables on your CAS server, the LOGSELECT procedure produces the “Output CAS Tables” table. This table displays the names, the number of rows and columns, and the CAS libraries in which the output table is created for every output table that you requested.

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 5.11.

**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 31 in Chapter 2, “Shared Concepts.”

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Statement</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClassInfo</td>
<td>Level information from the CLASS statement</td>
<td>CLASS</td>
<td>Default</td>
</tr>
<tr>
<td>ConvergenceStatus</td>
<td>Status of optimization at conclusion of optimization</td>
<td>Default</td>
<td></td>
</tr>
<tr>
<td>CorrB</td>
<td>Correlation matrix of parameter estimates</td>
<td>PROC LOGSELECT</td>
<td>CORRB</td>
</tr>
<tr>
<td>CovB</td>
<td>Covariance matrix of parameter estimates</td>
<td>PROC LOGSELECT</td>
<td>COVB</td>
</tr>
<tr>
<td>Dimensions</td>
<td>Model dimensions</td>
<td>Default</td>
<td></td>
</tr>
<tr>
<td>EntryCandidates</td>
<td>Details about candidates for entry into the model</td>
<td>SELECTION</td>
<td>METHOD=FORWARD or STEPWISE and DETAILS=STEP</td>
</tr>
<tr>
<td>FitStatistics</td>
<td>Fit statistics</td>
<td>Default</td>
<td></td>
</tr>
<tr>
<td>GlobalTest</td>
<td>Test of the model versus the null model</td>
<td>Default</td>
<td></td>
</tr>
<tr>
<td>IterHistory</td>
<td>Iteration history</td>
<td>PROC LOGSELECT</td>
<td>ITHIST</td>
</tr>
<tr>
<td>ModelInfo</td>
<td>Information about the modeling environment</td>
<td>Default</td>
<td></td>
</tr>
<tr>
<td>ModelAnova</td>
<td>Model analysis of variance (Type III)</td>
<td>MODEL</td>
<td>TYPE3</td>
</tr>
<tr>
<td>NObs</td>
<td>Number of observations read and used, and number of events and trials, if applicable</td>
<td>Default</td>
<td></td>
</tr>
<tr>
<td>OutCASTblFull</td>
<td>Library and name of the output data table, and number of rows and columns in the table</td>
<td>OUTPUT DISPLAYOUT</td>
<td>OUT=</td>
</tr>
</tbody>
</table>
Table 5.11  continued

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Statement</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ParameterEstimates</td>
<td>Solutions for the parameter estimates associated with effects in <code>MODEL</code> statements</td>
<td>Default</td>
<td></td>
</tr>
<tr>
<td>PredProbName</td>
<td>Displays the predicted probability variable in the scoring code associated with each response level</td>
<td>CODE</td>
<td>PCATALL</td>
</tr>
<tr>
<td>RemovalCandidates</td>
<td>Details about candidates for removal from the model</td>
<td>SELECTION</td>
<td>METHOD=BACKWARD or STEPWISE and DETAILS=STEP</td>
</tr>
<tr>
<td>ResponseProfile</td>
<td>Response categories for binary and multinomial data</td>
<td>Default</td>
<td></td>
</tr>
<tr>
<td>SelectedEffects</td>
<td>List of effects selected for the model</td>
<td>SELECTION</td>
<td>Default</td>
</tr>
<tr>
<td>SelectionInfo</td>
<td>Information about the settings for model selection</td>
<td>SELECTION</td>
<td>Default</td>
</tr>
<tr>
<td>SelectionReason</td>
<td>Reason why the particular model was selected</td>
<td>SELECTION</td>
<td>Default</td>
</tr>
<tr>
<td>SelectionSummary</td>
<td>Summary information about model selection steps</td>
<td>SELECTION</td>
<td>Default</td>
</tr>
<tr>
<td>StopReason</td>
<td>Reason for termination of model selection</td>
<td>SELECTION</td>
<td>Default</td>
</tr>
<tr>
<td>Timing</td>
<td>Absolute and relative times for tasks performed by the procedure</td>
<td>Default</td>
<td></td>
</tr>
</tbody>
</table>

Examples: LOGSELECT 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 8 and “Loading a SAS Data Set onto a CAS Server” on page 9 in Chapter 2, “Shared Concepts.”

Example 5.1: Model Selection

The following statements examine the same data as in the section “Getting Started: LOGSELECT Procedure” on page 171, 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 requests that all tables related to model selection be produced.
The model selection tables are shown in Output 5.1.1 through Output 5.1.3. Results from the selected model are shown in Output 5.1.4 and Output 5.1.5.

The “Selection Information” table in Output 5.1.1 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 5.1.1 Selection Information

<table>
<thead>
<tr>
<th>Selection Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selection Method</td>
</tr>
<tr>
<td>Select Criterion</td>
</tr>
<tr>
<td>Stop Criterion</td>
</tr>
<tr>
<td>Effect Hierarchy Enforced</td>
</tr>
<tr>
<td>Stop Horizon</td>
</tr>
</tbody>
</table>

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 5.1.2 displays this table for the first step; the other steps are not shown here.

Output 5.1.2 Step1 Entry Candidates

<table>
<thead>
<tr>
<th>Rank</th>
<th>Effect</th>
<th>SBC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>x8</td>
<td>128.7318</td>
</tr>
<tr>
<td>2</td>
<td>x2</td>
<td>129.0423</td>
</tr>
<tr>
<td>3</td>
<td>x4</td>
<td>129.3708</td>
</tr>
<tr>
<td>4</td>
<td>x9</td>
<td>131.0854</td>
</tr>
<tr>
<td>5</td>
<td>x1</td>
<td>131.8215</td>
</tr>
<tr>
<td>6</td>
<td>x10</td>
<td>132.4430</td>
</tr>
<tr>
<td>7</td>
<td>x5</td>
<td>132.9424</td>
</tr>
<tr>
<td>8</td>
<td>x3</td>
<td>132.9667</td>
</tr>
<tr>
<td>9</td>
<td>x7</td>
<td>133.0021</td>
</tr>
<tr>
<td>10</td>
<td>x6</td>
<td>133.0283</td>
</tr>
</tbody>
</table>

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 5.1.3 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, $x_2$ made the most significant contribution when an effect was added to a model that contains the intercept and $x_8$. 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 5.1.1) indicates that at most three steps beyond the minimum SBC value are taken.

In Output 5.1.3, the “Selection Summary” table is followed by three small tables that summarize why the process stopped and which model is selected.

**Output 5.1.3** Selection Summary Information

<table>
<thead>
<tr>
<th>Step</th>
<th>Effect Entered</th>
<th>Number Effects In</th>
<th>SBC</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Intercept</td>
<td>1</td>
<td>128.4253</td>
</tr>
<tr>
<td>1</td>
<td>$x_8$</td>
<td>2</td>
<td>128.7318</td>
</tr>
<tr>
<td>2</td>
<td>$x_2$</td>
<td>3</td>
<td>128.2892*</td>
</tr>
<tr>
<td>3</td>
<td>$x_4$</td>
<td>4</td>
<td>130.0901</td>
</tr>
<tr>
<td>4</td>
<td>$x_9$</td>
<td>5</td>
<td>131.9534</td>
</tr>
<tr>
<td>5</td>
<td>$x_1$</td>
<td>6</td>
<td>134.7945</td>
</tr>
</tbody>
</table>

* Optimal Value Of Criterion

Selection stopped at a local minimum of the STOP criterion.

The model at step 2 is selected.

**Selected Effects**: Intercept $x_2$ $x_8$

Output 5.1.4 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 5.9. 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 5.1.4** Fit Statistics and Null Test

<table>
<thead>
<tr>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Columns in Design</td>
</tr>
<tr>
<td>Number of Effects</td>
</tr>
<tr>
<td>Max Effect Columns</td>
</tr>
<tr>
<td>Rank of Design</td>
</tr>
<tr>
<td>Parameters in Optimization</td>
</tr>
</tbody>
</table>

**Testing Global Null Hypothesis: BETA=0**

<table>
<thead>
<tr>
<th>Test</th>
<th>DF</th>
<th>Chi-Square</th>
<th>Pr &gt; ChiSq</th>
</tr>
</thead>
<tbody>
<tr>
<td>Likelihood Ratio</td>
<td>2</td>
<td>9.4237</td>
<td>0.0090</td>
</tr>
</tbody>
</table>
The parameter estimates of the selected model are shown in Output 5.1.5. 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 5.1.5 Parameter Estimates

<table>
<thead>
<tr>
<th>Parameter</th>
<th>DF</th>
<th>Estimate</th>
<th>Error</th>
<th>Chi-Square</th>
<th>Pr &gt; ChiSq</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intercept</td>
<td>1</td>
<td>0.858377</td>
<td>0.550302</td>
<td>2.4331</td>
<td>0.1188</td>
</tr>
<tr>
<td>x2</td>
<td>1</td>
<td>-0.250240</td>
<td>0.114578</td>
<td>4.7700</td>
<td>0.0290</td>
</tr>
<tr>
<td>x8</td>
<td>1</td>
<td>1.783956</td>
<td>0.790845</td>
<td>5.0885</td>
<td>0.0241</td>
</tr>
</tbody>
</table>

You can construct the prediction equation for this model from the parameter estimates as follows. The estimated linear predictor for an observation is

$$\hat{\eta} = 0.8584 - 0.2503 \times x_2 + 1.7840 \times x_8$$

The predicted probability that variable \(y\) takes the value 0 is

$$\hat{P}_r(Y = 0) = \frac{1}{1 + \exp\{-\hat{\eta}\}}$$

Example 5.2: Modeling Binomial Data

If \(Y_1, \ldots, Y_n\) are independent binary (Bernoulli) random variables with common success probability \(\pi\), then their sum is a binomial random variable. In other words, a binomial random variable with parameters \(n\) and \(\pi\) can be generated as the sum of \(n\) Bernoulli(\(\pi\)) 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(\(n, \pi\)) distribution, where the success probability \(\pi\) is a function of heating and soaking time.

```plaintext
data Ingots;
  input Heat Soak r n @@;
  Obsnum= _n_;
datelines;
  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
```
Example 5.2: Modeling Binomial Data

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 OUTPUT statement stores the linear predictors and the predicted probabilities in the mycas.Out data table along with the other variables from the input data table.

```plaintext
data mycas.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 OUTPUT statement stores the linear predictors and the predicted probabilities in the mycas.Out data table along with the other variables from the input data table.

```plaintext
proc logselect data=mycas.Ingots;
   model r/n = Heat Soak Heat*Soak;
   output out=mycas.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 5.2.1). 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 5.2.1** Model Information and Number of Observations

<table>
<thead>
<tr>
<th>The LOGSELECT Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Model Information</strong></td>
</tr>
<tr>
<td>Data Source</td>
</tr>
<tr>
<td>Response Variable (Events)</td>
</tr>
<tr>
<td>Response Variable (Trials)</td>
</tr>
<tr>
<td>Distribution</td>
</tr>
<tr>
<td>Link Function</td>
</tr>
<tr>
<td>Optimization Technique</td>
</tr>
</tbody>
</table>

| Number of Observations Read | 19 |
| Number of Observations Used | 19 |

<table>
<thead>
<tr>
<th>Response Profile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ordered Value</td>
</tr>
<tr>
<td>Binary Outcome</td>
</tr>
<tr>
<td>Event</td>
</tr>
<tr>
<td>Nonevent</td>
</tr>
</tbody>
</table>

The second table in Output 5.2.1 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 5.2.2** displays the convergence status table for this run. The LOGSELECT procedure satisfies the GCONV= convergence criterion.
Output 5.2.2 Convergence Status

Convergence criterion (GCONV=1E-8) satisfied.

Output 5.2.3 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 5.2.3 Dimensions in Binomial Logistic Regression

<table>
<thead>
<tr>
<th>Dimensions</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Columns in Design</td>
<td>4</td>
</tr>
<tr>
<td>Number of Effects</td>
<td>4</td>
</tr>
<tr>
<td>Max Effect Columns</td>
<td>1</td>
</tr>
<tr>
<td>Rank of Design</td>
<td>4</td>
</tr>
<tr>
<td>Parameters in Optimization</td>
<td>4</td>
</tr>
</tbody>
</table>

Output 5.2.4 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 5.2.4 Fit Statistics

<table>
<thead>
<tr>
<th>Fit Statistics</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-2 Log Likelihood</td>
<td>27.9569</td>
</tr>
<tr>
<td>AIC (smaller is better)</td>
<td>35.9569</td>
</tr>
<tr>
<td>AICC (smaller is better)</td>
<td>38.81403</td>
</tr>
<tr>
<td>SBC (smaller is better)</td>
<td>39.73464</td>
</tr>
</tbody>
</table>

Output 5.2.5 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 5.2.5 Null Test

<table>
<thead>
<tr>
<th>Testing Global Null Hypothesis: BETA=0</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Test</td>
<td>DF</td>
</tr>
<tr>
<td>Likelihood Ratio</td>
<td>3</td>
</tr>
</tbody>
</table>

The “Parameter Estimates” table in Output 5.2.6 displays the estimates and standard errors of the model effects.
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

\[ \hat{\eta} = -5.9902 + 0.09634 \times 14 + 0.2996 \times 1.7 - 0.00884 \times 14 \times 7 = -4.34256 \]

The probability that an ingot with these characteristics is not ready for rolling is

\[ \hat{\pi} = \frac{1}{1 + \exp\left(-(-4.34256)\right)} = 0.01284 \]

PROC LOGSELECT computes these linear predictors and probabilities and stores them in the mycas.Out data set. The following PROC PRINT statement produces Output 5.2.7, 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=mycas.Out;
   where Heat=14 & Soak=1.7;
run;
```

Output 5.2.7  Predicted Probability for Heat=14 and Soak=1.7

<table>
<thead>
<tr>
<th>Obs</th>
<th>Pred <em>XBETA</em></th>
<th>Heat</th>
<th>Soak</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.012836</td>
<td>14</td>
<td>1.7</td>
</tr>
</tbody>
</table>

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 set, which contains 12 events in 387 trials, into a data table that has 387 observations:

```
data mycas.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.
Chapter 5: The LOGSELECT Procedure

```plaintext
proc logselect data=mycas.Ingots_binary;
   model y(event='1') = Heat Soak Heat*Soak;
run;
```

Output 5.2.8 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 5.2.8** Model Information in Binary Model

The LOGSELECT Procedure

<table>
<thead>
<tr>
<th>Model Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Source</td>
</tr>
<tr>
<td>Response Variable</td>
</tr>
<tr>
<td>Distribution</td>
</tr>
<tr>
<td>Link Function</td>
</tr>
<tr>
<td>Optimization Technique</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Number of Observations Read</th>
<th>387</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Observations Used</td>
<td>387</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Response Profile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ordered Value</td>
</tr>
<tr>
<td>------------------</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
</tbody>
</table>

Probability modeled is \( y = 1 \).

Output 5.2.9 displays the result of the test of the global null hypothesis and the parameter estimates. These results match those in Output 5.2.5 and Output 5.2.6.

**Output 5.2.9** Null Test and Parameter Estimates

Testing Global Null Hypothesis: BETA=0

<table>
<thead>
<tr>
<th>Test</th>
<th>DF</th>
<th>Chi-Square</th>
<th>Pr &gt; ChiSq</th>
</tr>
</thead>
<tbody>
<tr>
<td>Likelihood Ratio</td>
<td>3</td>
<td>11.7663</td>
<td>0.0082</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter Estimates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter</td>
</tr>
<tr>
<td>-----------</td>
</tr>
<tr>
<td>Intercept</td>
</tr>
<tr>
<td>Heat</td>
</tr>
<tr>
<td>Soak</td>
</tr>
<tr>
<td>Heat * Soak</td>
</tr>
</tbody>
</table>
Example 5.3: Partitioning Data

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.

```plaintext
data mycas.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.

```plaintext
class proc logselect data=mycas.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 5.3.1 and Output 5.3.2.

The “Number of Observations” and “Response Profile” tables in Output 5.3.1 are divided into training and testing columns.

**Output 5.3.1** Partitioned Counts

**The LOGSELECT Procedure**

<table>
<thead>
<tr>
<th>Description</th>
<th>Number of Observations</th>
<th>Total</th>
<th>Training</th>
<th>Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Observations Read</td>
<td>4601</td>
<td>3065</td>
<td>1536</td>
<td></td>
</tr>
<tr>
<td>Number of Observations Used</td>
<td>4601</td>
<td>3065</td>
<td>1536</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ordered Value</th>
<th>Total Frequency Training</th>
<th>Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0</td>
<td>2788</td>
<td>1847</td>
</tr>
<tr>
<td>2 1</td>
<td>1813</td>
<td>1218</td>
</tr>
</tbody>
</table>

Probability modeled is Class = 1.
Chapter 5: The LOGSELECT Procedure

The likelihood-based fit statistics for the selected model are displayed in the “Fit Statistics” table in Output 5.3.2, which has columns for the training and testing subsets.

Output 5.3.2  Partitioned Fit Statistics

<table>
<thead>
<tr>
<th>Description</th>
<th>Training</th>
<th>Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2 Log Likelihood</td>
<td>1242.59491</td>
<td>823.68742</td>
</tr>
<tr>
<td>AIC (smaller is better)</td>
<td>1292.59491</td>
<td>873.68742</td>
</tr>
<tr>
<td>AICC (smaller is better)</td>
<td>1293.02268</td>
<td>874.54835</td>
</tr>
<tr>
<td>SBC (smaller is better)</td>
<td>1443.28998</td>
<td>1007.11085</td>
</tr>
<tr>
<td>Average Square Error</td>
<td>0.05659</td>
<td>0.06353</td>
</tr>
<tr>
<td>-2 Log L (Intercept-only)</td>
<td>4118.98701</td>
<td>2050.73514</td>
</tr>
<tr>
<td>R-Square</td>
<td>0.60877</td>
<td>0.55016</td>
</tr>
<tr>
<td>Max-rescaled R-Square</td>
<td>0.82359</td>
<td>0.74661</td>
</tr>
<tr>
<td>McFadden's R-Square</td>
<td>0.69833</td>
<td>0.59835</td>
</tr>
<tr>
<td>Misclassification Rate</td>
<td>0.07471</td>
<td>0.07813</td>
</tr>
<tr>
<td>Difference of Means</td>
<td>0.75122</td>
<td>0.73431</td>
</tr>
</tbody>
</table>

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 200. 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:

```plaintext
proc logselect data=mycas.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 5.3.2.

Example 5.4: Ordinal Logistic Regression

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 mycas.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 mycas.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.

```plaintext
proc logselect data=mycas.Cheese;
    freq freq;
    class Additive(ref='4') / param=ref ;
    model y=Additive;
run;
```

Results from the logistic analysis are shown in Output 5.4.1 through Output 5.4.3.

The “Response Profile” table in Output 5.4.1 shows that the strong dislike ($y=1$) end of the rating scale is associated with lower Ordered Values in the “Response Profile” table; hence the probability of disliking the additives is modeled.

**Output 5.4.1** Proportional Odds Model Regression Analysis

**Multiple Response Cheese Tasting Experiment**

**The LOGSELECT Procedure**

<table>
<thead>
<tr>
<th>Model Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Source</td>
</tr>
<tr>
<td>Response Variable</td>
</tr>
<tr>
<td>Number of Response Levels</td>
</tr>
<tr>
<td>Frequency Variable</td>
</tr>
<tr>
<td>Distribution</td>
</tr>
<tr>
<td>Link Type</td>
</tr>
<tr>
<td>Link Function</td>
</tr>
<tr>
<td>Optimization Technique</td>
</tr>
</tbody>
</table>

| Number of Observations Read | 36 |
| Number of Observations Used | 28 |
| Sum of Frequencies Read | 208 |
| Sum of Frequencies Used | 208 |
Output 5.4.1  continued

<table>
<thead>
<tr>
<th>Response Profile</th>
<th>Ordered Value</th>
<th>Total Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>2 2</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>3 3</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>4 4</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>5 5</td>
<td>41</td>
<td></td>
</tr>
<tr>
<td>6 6</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td>7 7</td>
<td>39</td>
<td></td>
</tr>
<tr>
<td>8 8</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>9 9</td>
<td>12</td>
<td></td>
</tr>
</tbody>
</table>

Probabilities modeled are cumulated over the lower Ordered Values.

<table>
<thead>
<tr>
<th>Class Level Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
</tr>
<tr>
<td>Additive</td>
</tr>
</tbody>
</table>

Output 5.4.2  Proportional Odds Model Regression Analysis

Convergence criterion (GCONV=1E-8) satisfied.

<table>
<thead>
<tr>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Columns in Design</td>
</tr>
<tr>
<td>Number of Effects</td>
</tr>
<tr>
<td>Max Effect Columns</td>
</tr>
<tr>
<td>Rank of Design</td>
</tr>
<tr>
<td>Parameters in Optimization</td>
</tr>
</tbody>
</table>

Testing Global Null Hypothesis: BETA=0

<table>
<thead>
<tr>
<th>Test</th>
<th>DF</th>
<th>Chi-Square</th>
<th>Pr &gt; ChiSq</th>
</tr>
</thead>
<tbody>
<tr>
<td>Likelihood Ratio</td>
<td>3</td>
<td>148.4539</td>
<td>&lt;.0001</td>
</tr>
</tbody>
</table>

Fit Statistics

<table>
<thead>
<tr>
<th>-2 Log Likelihood</th>
<th>AIC (smaller is better)</th>
<th>AICC (smaller is better)</th>
<th>SBC (smaller is better)</th>
</tr>
</thead>
<tbody>
<tr>
<td>711.34790</td>
<td>733.34790</td>
<td>734.69484</td>
<td>770.06082</td>
</tr>
</tbody>
</table>

The positive value (1.6128) for the parameter estimate for Additive=1 in Output 5.4.3 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 5.4.3  Proportional Odds Model Regression Analysis

<table>
<thead>
<tr>
<th>Parameter</th>
<th>DF</th>
<th>Estimate</th>
<th>Standard Error</th>
<th>Chi-Square</th>
<th>Pr &gt; ChiSq</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intercept 1</td>
<td>1</td>
<td>7.080166</td>
<td>0.564010</td>
<td>157.5844</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Intercept 2</td>
<td>1</td>
<td>-6.024980</td>
<td>0.476431</td>
<td>159.9230</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Intercept 3</td>
<td>1</td>
<td>-4.925416</td>
<td>0.425651</td>
<td>133.8992</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Intercept 4</td>
<td>1</td>
<td>-3.856801</td>
<td>0.388022</td>
<td>98.7968</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Intercept 5</td>
<td>1</td>
<td>-2.520552</td>
<td>0.345268</td>
<td>53.2940</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Intercept 6</td>
<td>1</td>
<td>-1.568538</td>
<td>0.312208</td>
<td>25.2408</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Intercept 7</td>
<td>1</td>
<td>-0.066875</td>
<td>0.273819</td>
<td>0.0596</td>
<td>0.8071</td>
</tr>
<tr>
<td>Intercept 8</td>
<td>1</td>
<td>1.492974</td>
<td>0.335696</td>
<td>19.7794</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Additive 1</td>
<td>1</td>
<td>1.612791</td>
<td>0.380544</td>
<td>17.9617</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Additive 2</td>
<td>1</td>
<td>4.964640</td>
<td>0.476721</td>
<td>108.4546</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Additive 3</td>
<td>1</td>
<td>3.322683</td>
<td>0.421830</td>
<td>62.0444</td>
<td>&lt;.0001</td>
</tr>
</tbody>
</table>

References


# Chapter 6
## The NLMOD Procedure

## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview: NLMOD Procedure</td>
<td>226</td>
</tr>
<tr>
<td>PROC NLMOD Features</td>
<td>226</td>
</tr>
<tr>
<td>PROC NLMOD Compared with the Other SAS Procedures</td>
<td>226</td>
</tr>
<tr>
<td>PROC NLMOD Compared with the HPNLMOD Procedure</td>
<td>227</td>
</tr>
<tr>
<td>PROC NLMOD Compared with the NLIN Procedure</td>
<td>227</td>
</tr>
<tr>
<td>PROC NLMOD Compared with the NLMIXED Procedure</td>
<td>227</td>
</tr>
<tr>
<td>Using CAS Sessions and CAS Engine Librefs</td>
<td>227</td>
</tr>
<tr>
<td>Getting Started: NLMOD Procedure</td>
<td>228</td>
</tr>
<tr>
<td>Least Squares Model</td>
<td>228</td>
</tr>
<tr>
<td>Binomial Model</td>
<td>229</td>
</tr>
<tr>
<td>Syntax: NLMOD Procedure</td>
<td>231</td>
</tr>
<tr>
<td>PROC NLMOD Statement</td>
<td>232</td>
</tr>
<tr>
<td>BOUNDS Statement</td>
<td>234</td>
</tr>
<tr>
<td>BY Statement</td>
<td>234</td>
</tr>
<tr>
<td>DISPLAY Statement</td>
<td>234</td>
</tr>
<tr>
<td>DISPLAYOUT Statement</td>
<td>235</td>
</tr>
<tr>
<td>ESTIMATE Statement</td>
<td>236</td>
</tr>
<tr>
<td>ID Statement</td>
<td>236</td>
</tr>
<tr>
<td>MODEL Statement</td>
<td>236</td>
</tr>
<tr>
<td>PARAMETERS Statement</td>
<td>237</td>
</tr>
<tr>
<td>PREDICT Statement</td>
<td>240</td>
</tr>
<tr>
<td>RESTRICT Statement</td>
<td>241</td>
</tr>
<tr>
<td>Programming Statements</td>
<td>241</td>
</tr>
<tr>
<td>Details: NLMOD Procedure</td>
<td>243</td>
</tr>
<tr>
<td>Least Squares Estimation</td>
<td>243</td>
</tr>
<tr>
<td>Built-In Log-Likelihood Functions</td>
<td>244</td>
</tr>
<tr>
<td>Multithreading</td>
<td>246</td>
</tr>
<tr>
<td>Optimization Algorithms</td>
<td>246</td>
</tr>
<tr>
<td>Displayed Output</td>
<td>246</td>
</tr>
<tr>
<td>ODS Table Names</td>
<td>249</td>
</tr>
<tr>
<td>Examples: NLMOD Procedure</td>
<td>250</td>
</tr>
<tr>
<td>Example 6.1: Segmented Model</td>
<td>250</td>
</tr>
<tr>
<td>References</td>
<td>254</td>
</tr>
</tbody>
</table>
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

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
- 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 68 in Chapter 2, “Shared Concepts.”

PROC NLMOD Compared with the 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:

```plaintext
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:

```plaintext
cas mysess;
libname mycas cas sessref=mysess;
```
The CAS statement creates the CAS session named `mysess`, and the LIBNAME statement creates the `mycas` 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:

```sas
   cas mysess terminate;
```

For more information about the CAS and LIBNAME statements, see the section “Introduction to Shared Concepts” on page 8 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**

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

\[ f(x, \theta) = \frac{\theta_1 x_i}{\theta_2 + x_i}, \quad \text{for } i = 1, 2, \ldots, n \]

where \( x_i \) represents the concentration for \( n \) trials and \( f(x, \theta) \) is the reaction rate. The vector \( \theta \) 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 8 and “Loading a SAS Data Set onto a CAS Server” on page 9 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:

```sas
   data mycas.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 CAS engine libref is named mycas, but you can substitute any appropriately defined CAS engine libref.

The following SAS statements estimate the parameters $\theta_1$ and $\theta_2$:

```sas
proc nlmod data=mycas.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 6.1. The table displays the degrees of freedom, sums of squares, and mean squares along with the model $F$ test.

![Figure 6.1 Nonlinear Least Squares Analysis of Variance](#)

<table>
<thead>
<tr>
<th>Source</th>
<th>DF</th>
<th>Sum of Squares</th>
<th>Mean Square</th>
<th>F Value</th>
<th>Approx Pr &gt; F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>2</td>
<td>290116</td>
<td>145058</td>
<td>88537.2</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Error</td>
<td>12</td>
<td>19.6606</td>
<td>1.6384</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Uncorrected Total</td>
<td>14</td>
<td>290135</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

An intercept was not specified for this model.

Finally, Figure 6.2 displays the parameter estimates, standard errors, $t$ statistics, and 95% confidence intervals for $\theta_1$ and $\theta_2$.

![Figure 6.2 Parameter Estimates and Approximate 95% Confidence Intervals](#)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Estimate</th>
<th>Standard Error</th>
<th>DF</th>
<th>t Value</th>
<th>Approx Pr &gt;</th>
<th>Approx 95% Confidence Limits</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\theta_1$</td>
<td>158.1</td>
<td>0.6737</td>
<td>12</td>
<td>234.67</td>
<td>&lt;.0001</td>
<td>156.6 159.6</td>
</tr>
<tr>
<td>$\theta_2$</td>
<td>0.0741</td>
<td>0.00313</td>
<td>12</td>
<td>23.69</td>
<td>&lt;.0001</td>
<td>0.0673 0.0809</td>
</tr>
</tbody>
</table>

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**

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 mycas.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 1.10 .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 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=mycas.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 228. No analysis of variance table is produced for this model; fit statistics based on the value of the likelihood function are displayed in Figure 6.3.
Parameter estimates for the binary distribution model that uses the same quantities as in the section “Least Squares Model” on page 228 are displayed in Figure 6.4.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Estimate</th>
<th>Standard Error</th>
<th>DF</th>
<th>t Value</th>
<th>Approx Pr &gt;</th>
<th>Approximate 95% Confidence Limits</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>-36.7548</td>
<td>32.3607</td>
<td>27</td>
<td>-1.14</td>
<td>0.2660</td>
<td>-103.2 29.6439</td>
</tr>
<tr>
<td>a</td>
<td>-5.6298</td>
<td>4.6376</td>
<td>27</td>
<td>-1.21</td>
<td>0.2353</td>
<td>-15.1454 3.8858</td>
</tr>
<tr>
<td>b</td>
<td>-2.2513</td>
<td>0.9790</td>
<td>27</td>
<td>-2.30</td>
<td>0.0294</td>
<td>-4.2599 -0.2426</td>
</tr>
<tr>
<td>c</td>
<td>45.1815</td>
<td>34.9095</td>
<td>27</td>
<td>1.29</td>
<td>0.2065</td>
<td>-26.4469 116.8</td>
</tr>
</tbody>
</table>

**Syntax: NLMOD Procedure**

The following statements are available in the NLMOD procedure:

```plaintext
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 6.1 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>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA=</td>
<td>Specifies the input data table</td>
</tr>
<tr>
<td>OUT=</td>
<td>Specifies the output data table</td>
</tr>
</tbody>
</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
- **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 38 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=***\(\alpha\)**

specifies the level of significance *\(\alpha\)* 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=** *CAS-libref.data-table*

names the input data table to be used by PROC NLMOD. The default is the most recently created data table. *CAS-libref* specifies the CAS library where the input data table resides, and *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=** *CAS-libref.data-table*

names the output data table to be created when one or more PREDICT statements are specified. *CAS-libref* specifies the CAS library where the output data table is to be created, and *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.
BOUNDDS Statement

```r
BOUND(S 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:

```r
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 $\theta_j$, 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 $\theta_j$.

BY Statement

```r
BY variables;
```

You can specify a BY statement in PROC NLMOOD 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 SAS Language Reference: Concepts.

DISPLAY Statement

```r
DISPLAY <\ table-list > < / options >;
```

The DISPLAY statement enables you to specify a list of ODS 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, all ODS tables are sent to the client and then the client creates a subset. If both DISPLAY and ODS statements are used together, the DISPLAY statement takes precedence over the ODS statements. For more information about ODS, see SAS Output Delivery System: Procedures Guide.

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

**CASESENSITIVE**

performs a case-sensitive comparison of table names in the **table-list** to ODS 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 ODS tables except those specified in the table-list.

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

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

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

A path is a table name that is prefixed with dot-separated grouping information. For example, a SelectionSummary table that is produced by a procedure during a selection routine might have the path Bygroup1.Summary.SelectionSummary. A partial pathname does not include all groups; for example, Selection- Summary and Summary.SelectionSummary are partial pathnames for Bygroup1.Summary.SelectionSummary.

When you specify a table name or partial pathname, all ODS 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 starts with a “/” or a “!”. 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.

DISPLAYOUT Statement

DISPLAYOUT table-spec-list </options> ;

The DISPLAYOUT statement enables you to create CAS 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 CAS output tables to create. Each entry in the list has either a key or a key=value format:

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

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 (/):

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

REPEATED
   replicates the CAS output tables on all nodes.
Chapter 6: The NLMOD Procedure

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=\alpha
```

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 PROC NLMOD statement to produce prediction statistics, ID variables are added to the output data table.

By default, PROC NLMOD do 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 ($\sim$), 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 244.

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 238.

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=CAS-libref.data-table

DATA=CAS-libref.data-table

specifies the data table that provides parameter starting values. CAS-libref specifies the CAS library where the input data table resides, and data-table specifies the name of the input data table.

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:

\[ \text{name} = \text{value-list} \]

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
- \( m1, m2, \ldots, mn \) 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;
```

<table>
<thead>
<tr>
<th>Possible Starting Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>B0</td>
</tr>
<tr>
<td>----</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
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 CAS 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 CAS 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:

```plaintext
proc nlmmod data=mycas.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 nlmmod data=mycas.D;
parameters / pdata=mycas.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:

```plaintext
data mycas.Test;
   input Parameter $ Estimate;
datalines;
   alpha 100
   BETA 4
   beta 4.1
   beta 4.2
   beta 4.1
   gamma 30
;
```
PREDICT Statement

\begin{verbatim}
PREDICT 'label' expression <options> ;
PREDICT 'label' MEAN <options> ;
\end{verbatim}

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:

\begin{verbatim}
ALPHA=\alpha
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.
\end{verbatim}
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 nlmom;
  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.

---

**Details: NLMOD Procedure**

**Least Squares Estimation**

Models that are estimated by PROC NLMOD can be represented by using the equations

\[
Y = f(\beta; z_1, \ldots, z_k) + \epsilon \\
E[\epsilon] = 0 \\
\text{Var}[\epsilon] = \sigma^2 I
\]

where

$Y$ is the $(n \times 1)$ vector of observed responses.

$f$ is the nonlinear prediction function of parameters and regressor variables.

$\beta$ is the vector of model parameters to be estimated.

$z_1, \ldots, z_k$ are the $(n \times 1)$ vectors of each of the $k$ regressor variables.

$\epsilon$ is the $(n \times 1)$ vector of residuals.
\(\sigma^2\) 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 “Parameter Estimates” 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:

\(Y \sim \text{normal}(m, v)\)

\[
l(m, v; y) = -\frac{1}{2} \left( \log(2\pi) + \frac{(y - m)^2}{v} + \log(v) \right) \]

\[E[Y] = m\]

\[\text{Var}[Y] = v\]

\[v > 0\]

\(Y \sim \text{binary}(p)\)

\[l_1(p; y) = \begin{cases} y \log(p) & y > 0 \\ 0 & \text{otherwise} \end{cases}\]

\[l_2(p; y) = \begin{cases} (1 - y) \log(1 - p) & y < 1 \\ 0 & \text{otherwise} \end{cases}\]

\[l(p; y) = l_1(p; y) + l_2(p; y)\]

\[E[Y] = p\]

\[\text{Var}[Y] = p(1 - p)\]

\[0 < p < 1\]

\(Y \sim \text{binomial}(n, p)\)

\[l_c = \log\left(\Gamma(n + 1)\right) - \log\left(\Gamma(y + 1)\right) - \log\left(\Gamma(n - y + 1)\right)\]

\[l_1(n, p; y) = \begin{cases} y \log(p) & y > 0 \\ 0 & \text{otherwise} \end{cases}\]

\[l_2(n, p; y) = \begin{cases} (n - y) \log(1 - p) & n - y > 0 \\ 0 & \text{otherwise} \end{cases}\]

\[l(n, p; y) = l_c + l_1(n, p; y) + l_2(n, p; y)\]

\[E[Y] = np\]

\[\text{Var}[Y] = np(1 - p)\]

\[0 < p < 1\]
$Y \sim \text{gamma}(a, b)$

$$l(a, b; y) = -a \log\{b\} - \log\{\Gamma(a)\} + (a - 1) \log\{y\} - y/b$$

$$E[Y] = ab$$

$$\text{Var}[Y] = ab^2$$

$a > 0$

$b > 0$

This parameterization of the gamma distribution differs from the parameterization that the GENSELECT procedure uses. 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;
```

$Y \sim \text{negbin}(n, p)$

$$l(n, p; y) = \log\{\Gamma(n + y)\} - \log\{\Gamma(n)\} - \log\{\Gamma(y + 1)\} + n \log\{p\} + y \log\{1 - p\}$$

$$E[Y] = n \left( \frac{1 - p}{p} \right)$$

$$\text{Var}[Y] = n \left( \frac{1 - p}{p^2} \right)$$

$n \geq 0$

$0 < p < 1$

The value of parameter $n$ can be a real number; it does not have to be an integer.

$Y \sim \text{Poisson}(m)$

$$l(m; y) = y \log\{m\} - m - \log\{\Gamma(y + 1)\}$$

$$E[Y] = m$$

$$\text{Var}[Y] = m$$

$m > 0$
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 68 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 6.3 summarizes the optimization techniques available in PROC NLMOD.

<table>
<thead>
<tr>
<th>TECHNIQUE=</th>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUREG</td>
<td>Trust region method</td>
</tr>
<tr>
<td>NEWRAP</td>
<td>Newton-Raphson method with line search</td>
</tr>
<tr>
<td>NRRIDG</td>
<td>Newton-Raphson method with ridging</td>
</tr>
<tr>
<td>QUANEW</td>
<td>Quasi-Newton methods</td>
</tr>
<tr>
<td>DBLDOG</td>
<td>Double-dogleg method</td>
</tr>
<tr>
<td>CONGRA</td>
<td>Conjugate gradient methods</td>
</tr>
<tr>
<td>LEVMAR</td>
<td>Levenberg-Marquardt method</td>
</tr>
<tr>
<td>NMSIMP</td>
<td>Nelder-Mead simplex method</td>
</tr>
</tbody>
</table>

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 68 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:

\[
\text{AIC} = -2l + 2p \\
\text{AICC} = \begin{cases} 
-2l + 2pn/(n - p - 1) & n > p + 2 \\
-2l + 2p(p + 2) & \text{otherwise}
\end{cases} \\
\text{BIC} = -2l + p \log(n)
\]

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 6.4.

Table 6.4 ODS Tables Produced by PROC NLMOD

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Statement</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>AdditionalEstimates</td>
<td>Functions of estimated parameters and their associated statistics</td>
<td>ESTIMATE</td>
<td>Default</td>
</tr>
<tr>
<td>ANOVA</td>
<td>Least squares analysis of variance information</td>
<td>MODEL</td>
<td>RESIDUAL</td>
</tr>
<tr>
<td>Constraints</td>
<td>Information about the model’s linear constraints</td>
<td>RESTRICT</td>
<td>Default</td>
</tr>
<tr>
<td>ConvergenceStatus</td>
<td>Optimization success and convergence information</td>
<td>PROC NLMOD</td>
<td>Default</td>
</tr>
<tr>
<td>CorrB</td>
<td>Parameter correlation matrix</td>
<td>PROC NLMOD</td>
<td>CORR</td>
</tr>
<tr>
<td>CovB</td>
<td>Parameter covariance matrix</td>
<td>PROC NLMOD</td>
<td>COV</td>
</tr>
<tr>
<td>Dimensions</td>
<td>Number of parameters and their bounds</td>
<td>PROC NLMOD</td>
<td>Default</td>
</tr>
<tr>
<td>ECorrB</td>
<td>Additional estimates’ correlation matrix</td>
<td>PROC NLMOD</td>
<td>ECORR</td>
</tr>
<tr>
<td>ECovB</td>
<td>Additional estimates’ covariance matrix</td>
<td>PROC NLMOD</td>
<td>ECOV</td>
</tr>
<tr>
<td>FitStatistics</td>
<td>Statistics about the quality of the fit</td>
<td>PROC NLMOD</td>
<td>Default</td>
</tr>
</tbody>
</table>
Examples: NLMOD Procedure

Example 6.1: Segmented Model

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 $x_0$:

$$E[Y|x] = \begin{cases} 
\alpha + \beta x + \gamma x^2 & \text{if } x < x_0 \\
\gamma & \text{if } x \geq x_0
\end{cases}$$

In this model equation, $\alpha$, $\beta$, and $\gamma$ 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

$$c = E[Y|x_0] = \alpha + \beta x_0 + \gamma x_0^2$$

The smoothness condition requires that

$$\frac{\partial E[Y|x_0]}{\partial x} = \beta + 2\gamma x_0 \equiv 0$$
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:

```plaintext
data mycas.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:

```plaintext
proc nlmod data=mycas.A out=mycas.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 $\alpha$, $\beta$, and $\gamma$, 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 6.1.1 through Output 6.1.3. The iterative optimization converges after six iterations (Output 6.1.1). Output 6.1.2 shows the estimated parameters. Output 6.1.3 indicates that the join point is 12.7477 and the plateau value is 0.7775.
Output 6.1.1  Nonlinear Least Squares Iterative Phase

Quadratic Model with Plateau

The NLMOD Procedure

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Evaluations</th>
<th>Objective Function</th>
<th>Change</th>
<th>Max Gradient</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5</td>
<td>0.0035144531</td>
<td>7.184063</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0.0007352716</td>
<td>0.00277918</td>
<td>2.145337</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0.0006292751</td>
<td>0.00010600</td>
<td>0.032551</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>0.0006291261</td>
<td>0.00000015</td>
<td>0.002952</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>0.0006291244</td>
<td>0.00000000</td>
<td>0.000238</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>0.0006291244</td>
<td>0.00000000</td>
<td>0.000023</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>0.0006291244</td>
<td>0.00000000</td>
<td>2.313E-6</td>
</tr>
</tbody>
</table>

Convergence criterion (GCONV=1E-8) satisfied.

Output 6.1.2  Least Squares Analysis of the Quadratic Model

<table>
<thead>
<tr>
<th>Source</th>
<th>DF</th>
<th>Sum of Squares</th>
<th>Mean Square</th>
<th>F Value</th>
<th>Approx Pr &gt; F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>2</td>
<td>0.1769</td>
<td>0.0884</td>
<td>114.22</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Error</td>
<td>13</td>
<td>0.0101</td>
<td>0.000774</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Corrected Total</td>
<td>15</td>
<td>0.1869</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Output 6.1.3  Additional Estimates for the Quadratic Model

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Estimate</th>
<th>Standard Error</th>
<th>DF</th>
<th>t Value</th>
<th>Approx Pr &gt;</th>
<th>Approximate 95% Confidence Limits</th>
</tr>
</thead>
<tbody>
<tr>
<td>alpha</td>
<td>0.3921</td>
<td>0.0267</td>
<td>13</td>
<td>14.70</td>
<td>&lt;.0001</td>
<td>0.3345 0.4497</td>
</tr>
<tr>
<td>beta</td>
<td>0.0605</td>
<td>0.00842</td>
<td>13</td>
<td>7.18</td>
<td>&lt;.0001</td>
<td>0.0423 0.0787</td>
</tr>
<tr>
<td>gamma</td>
<td>-0.00237</td>
<td>0.000551</td>
<td>13</td>
<td>-4.30</td>
<td>0.0009</td>
<td>-0.00356 -0.00118</td>
</tr>
</tbody>
</table>

The following statements produce a graph, shown in Output 6.1.4, of the observed and predicted values along with reference lines for the join point and plateau estimates:
Example 6.1: Segmented Model

data B;
  set mycas.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 6.1.4 Observed and Predicted Values for the Quadratic Model
References


# Chapter 7
## The PCA Procedure

## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview: PCA Procedure</td>
<td>256</td>
</tr>
<tr>
<td>PROC PCA Features</td>
<td>256</td>
</tr>
<tr>
<td>PROC PCA Compared with Other SAS Procedures</td>
<td>257</td>
</tr>
<tr>
<td>PROC PCA Compared with the HPPRINCOMP Procedure</td>
<td>257</td>
</tr>
<tr>
<td>PROC PCA Compared with the PRINCOMP Procedure</td>
<td>258</td>
</tr>
<tr>
<td>Using CAS Sessions and CAS Engine Librefs</td>
<td>259</td>
</tr>
<tr>
<td>Getting Started: PCA Procedure</td>
<td>259</td>
</tr>
<tr>
<td>Syntax: PCA Procedure</td>
<td>264</td>
</tr>
<tr>
<td>PROC PCA Statement</td>
<td>264</td>
</tr>
<tr>
<td>BY Statement</td>
<td>269</td>
</tr>
<tr>
<td>CODE Statement</td>
<td>269</td>
</tr>
<tr>
<td>DISPLAY Statement</td>
<td>269</td>
</tr>
<tr>
<td>DISPLAYOUT Statement</td>
<td>270</td>
</tr>
<tr>
<td>FREQ Statement</td>
<td>271</td>
</tr>
<tr>
<td>OUTPUT Statement</td>
<td>271</td>
</tr>
<tr>
<td>PARTIAL Statement</td>
<td>273</td>
</tr>
<tr>
<td>VAR Statement</td>
<td>273</td>
</tr>
<tr>
<td>WEIGHT Statement</td>
<td>273</td>
</tr>
<tr>
<td>Details: PCA Procedure</td>
<td>274</td>
</tr>
<tr>
<td>Computing Principal Components</td>
<td>274</td>
</tr>
<tr>
<td>Eigenvalue Decomposition</td>
<td>274</td>
</tr>
<tr>
<td>NIPALS</td>
<td>274</td>
</tr>
<tr>
<td>ITERGS</td>
<td>274</td>
</tr>
<tr>
<td>Missing Values</td>
<td>275</td>
</tr>
<tr>
<td>OUTSTAT= Data Table</td>
<td>275</td>
</tr>
<tr>
<td>Displayed Output</td>
<td>277</td>
</tr>
<tr>
<td>Model Information</td>
<td>277</td>
</tr>
<tr>
<td>Number of Variables</td>
<td>277</td>
</tr>
<tr>
<td>Number of Observations</td>
<td>277</td>
</tr>
<tr>
<td>Simple Statistics</td>
<td>277</td>
</tr>
<tr>
<td>Centering and Scaling Information</td>
<td>277</td>
</tr>
<tr>
<td>Explained Variation of Variables</td>
<td>277</td>
</tr>
<tr>
<td>Correlation Matrix</td>
<td>278</td>
</tr>
<tr>
<td>Regression Statistics</td>
<td>278</td>
</tr>
<tr>
<td>Regression Coefficients</td>
<td>278</td>
</tr>
<tr>
<td>Partial Correlation Matrix</td>
<td>278</td>
</tr>
</tbody>
</table>
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.

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 68 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 7.1.

<table>
<thead>
<tr>
<th>Feature</th>
<th>PROC PCA</th>
<th>PROC HPPRINCOMP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supports various methods to extract principal components by using the METHOD= option in the PROC statement</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Supports BY statement</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Supports DISPLAY and DISPLAYOUT statements</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Supports CODE and OUTPUT statements</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Table 7.1  continued

<table>
<thead>
<tr>
<th>Feature</th>
<th>PROC PCA</th>
<th>PROC HPGRINCOMP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supports ID statement</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Supports PLOTS= option</td>
<td>Yes, with limited number of options</td>
<td>No</td>
</tr>
<tr>
<td>Includes input variables in output data table</td>
<td>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.</td>
<td>Not by default, in order to avoid data duplication for large data sets. To include input variables, specify them in the ID statement.</td>
</tr>
<tr>
<td>Threading</td>
<td>Specifically designed for CAS; executes on multiple threads</td>
<td>Primarily designed for a distributed environment; executes on multiple threads</td>
</tr>
</tbody>
</table>

PROC PCA Compared with the PRINCOMP Procedure

The PCA procedure and the PRINCOMP procedure have the similarities and differences shown in Table 7.2.

Table 7.2  Comparison of PROC PCA and PROC PRINCOMP

<table>
<thead>
<tr>
<th>Feature</th>
<th>PROC PCA</th>
<th>PROC PRINCOMP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supports various methods to extract principal components by using the METHOD= option in the PROC statement</td>
<td>Yes</td>
<td>No, supports only eigenvalue decomposition</td>
</tr>
<tr>
<td>Supports BY statement</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Supports DISPLAY and DISPLAYOUT statements</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Supports CODE and OUTPUT statements</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Supports ID statement</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Supports PLOTS= option</td>
<td>Limited number of options</td>
<td>Full set of options</td>
</tr>
<tr>
<td>Accepts input data</td>
<td>Only CAS input data tables</td>
<td>Ordinary SAS data sets and certain types of special SAS data sets</td>
</tr>
<tr>
<td>Includes input variables in output data table</td>
<td>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.</td>
<td>Yes</td>
</tr>
<tr>
<td>Threading</td>
<td>Specifically designed for CAS; executes on multiple threads</td>
<td>Executes on a single thread</td>
</tr>
</tbody>
</table>


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 mycas cas sessref=mysess;
```

The CAS statement creates the CAS session named `mysess`, and the LIBNAME statement creates the `mycas` 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 8 in Chapter 2, “Shared Concepts.”

Getting Started: PCA 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 8 and “Loading a SAS Data Set onto a CAS Server” on page 9 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 mycas.Crime;
  title 'Crime Rates per 100,000 Population by State';
  input State $1-15 Murder Rape Robbery Assault
           Burglary Larceny Auto_Theft;
  datalines;
```
Chapter 7: The PCA Procedure

<table>
<thead>
<tr>
<th>State</th>
<th>Score</th>
<th>Comp 1</th>
<th>Comp 2</th>
<th>Comp 3</th>
<th>Comp 4</th>
<th>Comp 5</th>
<th>Comp 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alabama</td>
<td>14.2</td>
<td>25.2</td>
<td>96.8</td>
<td>278.3</td>
<td>1135.5</td>
<td>1881.9</td>
<td>280.7</td>
</tr>
<tr>
<td>Alaska</td>
<td>10.8</td>
<td>51.6</td>
<td>96.8</td>
<td>284.0</td>
<td>1331.7</td>
<td>3369.8</td>
<td>753.3</td>
</tr>
<tr>
<td>Arizona</td>
<td>9.5</td>
<td>34.2</td>
<td>138.2</td>
<td>312.3</td>
<td>2346.1</td>
<td>4467.4</td>
<td>439.5</td>
</tr>
<tr>
<td>Arkansas</td>
<td>8.8</td>
<td>27.6</td>
<td>83.2</td>
<td>203.4</td>
<td>972.6</td>
<td>1862.1</td>
<td>183.4</td>
</tr>
<tr>
<td>California</td>
<td>11.5</td>
<td>49.4</td>
<td>287.0</td>
<td>358.0</td>
<td>2139.4</td>
<td>3499.8</td>
<td>663.5</td>
</tr>
<tr>
<td>Colorado</td>
<td>6.3</td>
<td>42.0</td>
<td>170.7</td>
<td>292.9</td>
<td>1935.2</td>
<td>3903.2</td>
<td>477.1</td>
</tr>
<tr>
<td>Connecticut</td>
<td>4.2</td>
<td>16.8</td>
<td>129.5</td>
<td>1346.0</td>
<td>1859.9</td>
<td>3840.5</td>
<td>351.4</td>
</tr>
<tr>
<td>Delaware</td>
<td>6.0</td>
<td>24.9</td>
<td>157.0</td>
<td>1682.6</td>
<td>3678.4</td>
<td>467.0</td>
<td></td>
</tr>
<tr>
<td>Florida</td>
<td>10.2</td>
<td>39.6</td>
<td>187.9</td>
<td>449.1</td>
<td>1859.9</td>
<td>3840.5</td>
<td>351.4</td>
</tr>
<tr>
<td>Georgia</td>
<td>11.7</td>
<td>31.1</td>
<td>140.5</td>
<td>256.5</td>
<td>1351.1</td>
<td>2170.2</td>
<td>297.9</td>
</tr>
<tr>
<td>Hawaii</td>
<td>7.2</td>
<td>25.5</td>
<td>128.0</td>
<td>1911.5</td>
<td>3920.4</td>
<td>489.4</td>
<td></td>
</tr>
<tr>
<td>Idaho</td>
<td>5.5</td>
<td>19.4</td>
<td>39.6</td>
<td>172.5</td>
<td>1050.8</td>
<td>2599.6</td>
<td>237.6</td>
</tr>
<tr>
<td>Illinois</td>
<td>11.5</td>
<td>21.8</td>
<td>211.3</td>
<td>1085.0</td>
<td>2828.5</td>
<td>528.6</td>
<td></td>
</tr>
<tr>
<td>Indiana</td>
<td>7.4</td>
<td>26.5</td>
<td>123.2</td>
<td>153.5</td>
<td>1086.2</td>
<td>2498.7</td>
<td>377.4</td>
</tr>
<tr>
<td>Iowa</td>
<td>2.3</td>
<td>10.6</td>
<td>41.2</td>
<td>89.8</td>
<td>812.5</td>
<td>2685.1</td>
<td>219.9</td>
</tr>
<tr>
<td>Kansas</td>
<td>6.6</td>
<td>22.0</td>
<td>100.7</td>
<td>180.5</td>
<td>1270.4</td>
<td>2739.3</td>
<td>244.3</td>
</tr>
<tr>
<td>Kentucky</td>
<td>10.1</td>
<td>19.1</td>
<td>81.1</td>
<td>123.3</td>
<td>872.2</td>
<td>1662.1</td>
<td>245.4</td>
</tr>
<tr>
<td>Louisiana</td>
<td>15.5</td>
<td>30.9</td>
<td>142.9</td>
<td>335.5</td>
<td>1165.5</td>
<td>2469.9</td>
<td>337.7</td>
</tr>
<tr>
<td>Maine</td>
<td>2.4</td>
<td>13.5</td>
<td>38.7</td>
<td>170.0</td>
<td>1253.1</td>
<td>2350.7</td>
<td>246.9</td>
</tr>
<tr>
<td>Maryland</td>
<td>8.0</td>
<td>34.8</td>
<td>292.1</td>
<td>358.9</td>
<td>1400.0</td>
<td>3177.7</td>
<td>428.5</td>
</tr>
<tr>
<td>Massachusetts</td>
<td>3.1</td>
<td>20.8</td>
<td>169.1</td>
<td>231.6</td>
<td>1532.2</td>
<td>2311.3</td>
<td>1140.1</td>
</tr>
<tr>
<td>Michigan</td>
<td>9.3</td>
<td>38.9</td>
<td>261.9</td>
<td>274.6</td>
<td>1522.7</td>
<td>3159.0</td>
<td>545.5</td>
</tr>
<tr>
<td>Minnesota</td>
<td>2.7</td>
<td>19.5</td>
<td>85.9</td>
<td>85.8</td>
<td>134.7</td>
<td>2559.3</td>
<td>343.1</td>
</tr>
<tr>
<td>Mississippi</td>
<td>14.3</td>
<td>19.6</td>
<td>65.7</td>
<td>189.1</td>
<td>915.6</td>
<td>1239.9</td>
<td>144.4</td>
</tr>
<tr>
<td>Missouri</td>
<td>9.6</td>
<td>28.3</td>
<td>189.0</td>
<td>233.5</td>
<td>1318.3</td>
<td>2424.2</td>
<td>378.4</td>
</tr>
<tr>
<td>Montana</td>
<td>5.4</td>
<td>16.7</td>
<td>39.2</td>
<td>156.8</td>
<td>804.9</td>
<td>2773.2</td>
<td>309.2</td>
</tr>
<tr>
<td>Nebraska</td>
<td>3.9</td>
<td>18.1</td>
<td>64.7</td>
<td>112.7</td>
<td>760.0</td>
<td>2316.1</td>
<td>249.1</td>
</tr>
<tr>
<td>Nevada</td>
<td>15.8</td>
<td>49.1</td>
<td>323.1</td>
<td>355.0</td>
<td>2453.1</td>
<td>4212.6</td>
<td>559.2</td>
</tr>
<tr>
<td>New Hampshire</td>
<td>3.2</td>
<td>10.7</td>
<td>23.2</td>
<td>76.0</td>
<td>1041.7</td>
<td>2343.9</td>
<td>293.4</td>
</tr>
<tr>
<td>New Jersey</td>
<td>5.6</td>
<td>21.0</td>
<td>180.4</td>
<td>185.1</td>
<td>1435.8</td>
<td>2774.5</td>
<td>511.5</td>
</tr>
<tr>
<td>New Mexico</td>
<td>8.8</td>
<td>39.1</td>
<td>109.6</td>
<td>343.4</td>
<td>1418.7</td>
<td>3008.6</td>
<td>259.5</td>
</tr>
<tr>
<td>New York</td>
<td>10.7</td>
<td>29.4</td>
<td>472.6</td>
<td>319.1</td>
<td>1728.0</td>
<td>2782.0</td>
<td>745.8</td>
</tr>
<tr>
<td>North Carolina</td>
<td>10.6</td>
<td>17.0</td>
<td>61.3</td>
<td>318.3</td>
<td>1154.1</td>
<td>2037.8</td>
<td>192.1</td>
</tr>
<tr>
<td>North Dakota</td>
<td>0.9</td>
<td>9.0</td>
<td>13.3</td>
<td>43.8</td>
<td>446.1</td>
<td>1843.0</td>
<td>144.7</td>
</tr>
<tr>
<td>Ohio</td>
<td>7.8</td>
<td>27.3</td>
<td>190.5</td>
<td>181.1</td>
<td>1216.0</td>
<td>2696.8</td>
<td>400.4</td>
</tr>
<tr>
<td>Oklahoma</td>
<td>8.6</td>
<td>29.2</td>
<td>73.8</td>
<td>205.0</td>
<td>1288.2</td>
<td>2228.1</td>
<td>326.8</td>
</tr>
<tr>
<td>Oregon</td>
<td>4.9</td>
<td>39.9</td>
<td>124.1</td>
<td>286.9</td>
<td>1636.4</td>
<td>3506.1</td>
<td>388.9</td>
</tr>
<tr>
<td>Pennsylvania</td>
<td>5.6</td>
<td>19.0</td>
<td>130.3</td>
<td>128.0</td>
<td>877.5</td>
<td>1624.1</td>
<td>333.2</td>
</tr>
<tr>
<td>Rhode Island</td>
<td>3.6</td>
<td>10.5</td>
<td>86.5</td>
<td>201.0</td>
<td>1489.5</td>
<td>2844.1</td>
<td>791.4</td>
</tr>
<tr>
<td>South Carolina</td>
<td>11.9</td>
<td>33.0</td>
<td>105.9</td>
<td>485.3</td>
<td>1613.6</td>
<td>2342.4</td>
<td>245.1</td>
</tr>
<tr>
<td>South Dakota</td>
<td>2.0</td>
<td>13.5</td>
<td>17.9</td>
<td>155.7</td>
<td>570.5</td>
<td>1704.4</td>
<td>147.5</td>
</tr>
<tr>
<td>Tennessee</td>
<td>10.1</td>
<td>29.7</td>
<td>145.8</td>
<td>203.9</td>
<td>1259.7</td>
<td>1776.5</td>
<td>314.0</td>
</tr>
<tr>
<td>Texas</td>
<td>13.3</td>
<td>33.8</td>
<td>152.4</td>
<td>208.2</td>
<td>1603.1</td>
<td>2988.7</td>
<td>397.6</td>
</tr>
<tr>
<td>Utah</td>
<td>3.5</td>
<td>20.3</td>
<td>68.8</td>
<td>147.3</td>
<td>1171.6</td>
<td>3004.6</td>
<td>334.5</td>
</tr>
<tr>
<td>Vermont</td>
<td>1.4</td>
<td>15.9</td>
<td>30.8</td>
<td>101.2</td>
<td>1348.2</td>
<td>2201.0</td>
<td>265.2</td>
</tr>
<tr>
<td>Virginia</td>
<td>9.0</td>
<td>23.3</td>
<td>92.1</td>
<td>165.7</td>
<td>986.2</td>
<td>2521.2</td>
<td>226.7</td>
</tr>
<tr>
<td>Washington</td>
<td>4.3</td>
<td>39.6</td>
<td>106.2</td>
<td>224.8</td>
<td>1605.6</td>
<td>3386.9</td>
<td>360.3</td>
</tr>
<tr>
<td>West Virginia</td>
<td>6.0</td>
<td>13.2</td>
<td>42.2</td>
<td>597.4</td>
<td>1341.7</td>
<td>163.3</td>
<td></td>
</tr>
<tr>
<td>Wyoming</td>
<td>2.8</td>
<td>12.9</td>
<td>52.2</td>
<td>63.7</td>
<td>846.9</td>
<td>2614.2</td>
<td>220.7</td>
</tr>
<tr>
<td></td>
<td>21.9</td>
<td>39.7</td>
<td>173.9</td>
<td>811.6</td>
<td>2772.2</td>
<td>282.0</td>
<td></td>
</tr>
</tbody>
</table>

; These statements assume that your CAS engine libref is named mycas, 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 7.1 through Figure 7.4:

```plaintext
proc pca data=mycas.Crime;
run;
```

Figure 7.1 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 7.1 Model Information and Simple Statistics](image)

**Crime Rates per 100,000 Population by State**

<table>
<thead>
<tr>
<th>The PCA Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model Information</td>
</tr>
<tr>
<td>Data Source</td>
</tr>
<tr>
<td>Component Extraction Method</td>
</tr>
<tr>
<td>Number of Variables</td>
</tr>
<tr>
<td>Number of Principal Components</td>
</tr>
<tr>
<td>Number of Observations Read</td>
</tr>
<tr>
<td>Number of Observations Used</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Simple Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
</tr>
<tr>
<td>Murder</td>
</tr>
<tr>
<td>Rape</td>
</tr>
<tr>
<td>Robbery</td>
</tr>
<tr>
<td>Assault</td>
</tr>
<tr>
<td>Burglary</td>
</tr>
<tr>
<td>Larceny</td>
</tr>
<tr>
<td>Auto_Theft</td>
</tr>
</tbody>
</table>

Figure 7.2 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.
Chapter 7: The PCA Procedure

Figure 7.2 Correlation Matrix Table

<table>
<thead>
<tr>
<th>Variable</th>
<th>Murder</th>
<th>Rape</th>
<th>Robbery</th>
<th>Assault</th>
<th>Burglary</th>
<th>Larceny</th>
<th>Auto_Theft</th>
</tr>
</thead>
<tbody>
<tr>
<td>Murder</td>
<td>1.0000</td>
<td>0.6000</td>
<td>0.4768</td>
<td>0.6485</td>
<td>0.3778</td>
<td>0.0925</td>
<td>0.0555</td>
</tr>
<tr>
<td>Rape</td>
<td>0.6000</td>
<td>1.0000</td>
<td>0.5817</td>
<td>0.7316</td>
<td>0.7038</td>
<td>0.6009</td>
<td>0.3282</td>
</tr>
<tr>
<td>Robbery</td>
<td>0.4768</td>
<td>0.5817</td>
<td>1.0000</td>
<td>0.5452</td>
<td>0.6200</td>
<td>0.4371</td>
<td>0.5787</td>
</tr>
<tr>
<td>Assault</td>
<td>0.6485</td>
<td>0.7316</td>
<td>0.5452</td>
<td>1.0000</td>
<td>0.6082</td>
<td>0.3791</td>
<td>0.2520</td>
</tr>
<tr>
<td>Burglary</td>
<td>0.3778</td>
<td>0.7038</td>
<td>0.6200</td>
<td>0.6082</td>
<td>1.0000</td>
<td>0.7932</td>
<td>0.5390</td>
</tr>
<tr>
<td>Larceny</td>
<td>0.0925</td>
<td>0.6009</td>
<td>0.4371</td>
<td>0.3791</td>
<td>0.7932</td>
<td>1.0000</td>
<td>0.4246</td>
</tr>
<tr>
<td>Auto_Theft</td>
<td>0.0555</td>
<td>0.3282</td>
<td>0.5787</td>
<td>0.2520</td>
<td>0.5390</td>
<td>0.4246</td>
<td>1.0000</td>
</tr>
</tbody>
</table>

Figure 7.3 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 7.3 Eigenvalues Table

<table>
<thead>
<tr>
<th>Eigenvalues of the Correlation Matrix</th>
<th>Eigenvalue</th>
<th>Difference</th>
<th>Proportion</th>
<th>Cumulative</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4.045824</td>
<td>2.781795</td>
<td>0.5780</td>
<td>0.5780</td>
</tr>
<tr>
<td>2</td>
<td>1.264030</td>
<td>0.516529</td>
<td>0.1806</td>
<td>0.7586</td>
</tr>
<tr>
<td>3</td>
<td>0.747500</td>
<td>0.421175</td>
<td>0.1068</td>
<td>0.8653</td>
</tr>
<tr>
<td>4</td>
<td>0.326325</td>
<td>0.061119</td>
<td>0.0466</td>
<td>0.9120</td>
</tr>
<tr>
<td>5</td>
<td>0.265207</td>
<td>0.036843</td>
<td>0.0379</td>
<td>0.9498</td>
</tr>
<tr>
<td>6</td>
<td>0.228364</td>
<td>0.105613</td>
<td>0.0326</td>
<td>0.9825</td>
</tr>
<tr>
<td>7</td>
<td>0.122750</td>
<td>0.0175</td>
<td>1.0000</td>
<td></td>
</tr>
</tbody>
</table>

Figure 7.4 displays the “Eigenvectors” table. From the eigenvectors matrix, you can represent the first principal component, Prin1, as a linear combination of the original variables:

\[ Prin1 = -0.30289 \times (\text{Murder}) - 0.43410 \times (\text{Rape}) - 0.39705 \times (\text{Robbery}) \]

.  
.  
.  

- 0.28834 \times (\text{Auto_Theft})
Similarly, the second principal component, Prin2, is

\[ \text{Prin2} = 0.61893 \times (\text{Murder}) + 0.17053 \times (\text{Rape}) - 0.04713 \times (\text{Robbery}) \]

\[ - 0.50400 \times (\text{Auto_Theft}) \]

where the variables are standardized.

**Figure 7.4 Eigenvectors Table**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Prin1</th>
<th>Prin2</th>
<th>Prin3</th>
<th>Prin4</th>
<th>Prin5</th>
<th>Prin6</th>
<th>Prin7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Murder</td>
<td>-0.30289</td>
<td>0.61893</td>
<td>0.17353</td>
<td>-0.23308</td>
<td>-0.54896</td>
<td>0.26371</td>
<td>-0.26428</td>
</tr>
<tr>
<td>Rape</td>
<td>-0.43410</td>
<td>0.17053</td>
<td>-0.23539</td>
<td>0.06540</td>
<td>-0.18075</td>
<td>-0.78232</td>
<td>0.27946</td>
</tr>
<tr>
<td>Robbery</td>
<td>-0.39705</td>
<td>-0.04713</td>
<td>0.49208</td>
<td>-0.57470</td>
<td>0.50808</td>
<td>-0.09452</td>
<td>0.02497</td>
</tr>
<tr>
<td>Assault</td>
<td>-0.39622</td>
<td>0.35142</td>
<td>-0.05343</td>
<td>0.61743</td>
<td>0.51525</td>
<td>0.17395</td>
<td>-0.19921</td>
</tr>
<tr>
<td>Burglary</td>
<td>-0.44164</td>
<td>-0.20861</td>
<td>-0.22454</td>
<td>-0.02750</td>
<td>-0.11273</td>
<td>0.52340</td>
<td>0.65085</td>
</tr>
<tr>
<td>Larceny</td>
<td>-0.35634</td>
<td>-0.40570</td>
<td>-0.53681</td>
<td>-0.23231</td>
<td>-0.02172</td>
<td>0.04085</td>
<td>-0.60346</td>
</tr>
<tr>
<td>Auto_Theft</td>
<td>-0.28834</td>
<td>-0.50400</td>
<td>0.57524</td>
<td>0.41853</td>
<td>-0.35939</td>
<td>-0.06024</td>
<td>-0.15487</td>
</tr>
</tbody>
</table>

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= CAS-libref.data-table
    <COPYVARS=(variables)>
    <keyword=<prefix>>...<keyword=<prefix>>;
  PARTIAL variables ;
  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 7.3 summarizes the options available in the PROC PCA statement.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify Data Sets</td>
<td></td>
</tr>
<tr>
<td>DATA=</td>
<td>Specifies the CAS input data table</td>
</tr>
<tr>
<td>OUTSTAT=</td>
<td>Specifies the CAS output data table that contains various statistics</td>
</tr>
<tr>
<td>Specify Details of Analysis</td>
<td></td>
</tr>
<tr>
<td>COV</td>
<td>Computes the principal components from the covariance matrix</td>
</tr>
<tr>
<td>METHOD=</td>
<td>Specifies the principal component extraction method to be used</td>
</tr>
<tr>
<td>N=</td>
<td>Specifies the number of principal components to be computed</td>
</tr>
<tr>
<td>NOINT</td>
<td>Omits the intercept from the model</td>
</tr>
<tr>
<td>PREFIX=</td>
<td>Specifies a prefix for naming the principal components</td>
</tr>
<tr>
<td>PARPREFIX=</td>
<td>Specifies a prefix for naming the residual variables</td>
</tr>
<tr>
<td>SINGULAR=</td>
<td>Specifies the singularity criterion</td>
</tr>
<tr>
<td>STD</td>
<td>Standardizes the principal component scores</td>
</tr>
<tr>
<td>VARDEF=</td>
<td>Specifies the divisor used to calculate variances and standard deviations</td>
</tr>
</tbody>
</table>
Table 7.3 continued

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify ODS Graphics Details</td>
<td>PLOTS= Specifies options that control the details of the plots</td>
</tr>
</tbody>
</table>

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=** `CAS-libref.data-table`
names the input data table for PROC PCA to use. The default is the most recently created data table. `CAS-libref.data-table` is a two-level name, where

- **CAS-libref** refers to a collection of information that is defined in the LIBNAME statement and includes the `caslib`, 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 `CAS-libref`, see the section “Using CAS Sessions and CAS Engine Librefs” on page 259.

- **data-table** specifies the name of the input data table.

**METHOD=** `EIG | ITERGS< (iter-options) > | NIPALS< (iter-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 also specify the following optional `iter-options` in parentheses after METHOD=ITERGS:

**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, MAX-ITER=500.
Chapter 7: The PCA Procedure

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<(iter-options)>
requests the nonlinear iterative partial least squares (NIPALS) method. You can also specify the optional iter-options in parentheses after METHOD=NIPALS.

By default, METHOD=EIG. If you specify METHOD=NIPALS or METHOD=ITERGS, 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. The default is the number of variables. 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.

If you specify this option and also create an OUTSTAT= data table, the data table’s type is UCORR or UCOV rather than CORR or COV. For more information, see the section “OUTSTAT= Data Table” on page 275.

OUTSTAT=CAS-libref.data-table
creates an output data table to contain means, standard deviations, number of observations, correlations or covariances, eigenvalues, and eigenvectors. CAS-libref specifies the CAS library where the output data table is to be created, and data-table specifies the name of the output data table. If you specify the COV option, the data table’s type is COV or UCOV, depending on the NOINT option, and it contains covariances; otherwise, the data table’s type is CORR or UCORR, depending on the NOINT option, and it contains correlations. 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 275.

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

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.

**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.
268 ♦ Chapter 7: The PCA Procedure

**PREFIX=name**

specifies a prefix for naming the principal components. By default, the names are Prin1, Prin2, …, Prin_._. 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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Divisor</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>DF</td>
<td>Error degrees of freedom</td>
<td>( n - i ) (before partialing)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( n - p - i ) (after partialing)</td>
</tr>
<tr>
<td>N</td>
<td>Number of observations</td>
<td>( n )</td>
</tr>
<tr>
<td>WEIGHT</td>
<td>WGT</td>
<td>Sum of weights</td>
</tr>
<tr>
<td>WDF</td>
<td>Sum of weights minus one</td>
<td>( \left( \sum_{j=1}^{n} w_j \right) - i ) (before partialing)</td>
</tr>
</tbody>
</table>

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 SAS Language Reference: Concepts.

CODE Statement

CODE < options > ;

The CODE statement writes SAS DATA step code for computing the principal component scores either to a file or to a catalog entry. This code can then be included in a DATA step to score new data.

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 7.4 summarizes the options available in the CODE statement.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMMENT</td>
<td>Adds comments to the generated code</td>
</tr>
<tr>
<td>FILE=</td>
<td>Names the file where the generated code is saved</td>
</tr>
<tr>
<td>FORMATWIDTH=</td>
<td>Specifies the numeric format width for the regression coefficients</td>
</tr>
<tr>
<td>INDENTSIZE=</td>
<td>Specifies the number of spaces to indent the generated code</td>
</tr>
<tr>
<td>LABELID=</td>
<td>Specifies a number used to construct names and labels</td>
</tr>
<tr>
<td>LINESIZE=</td>
<td>Specifies the line size for the generated code</td>
</tr>
<tr>
<td>NOTRIM</td>
<td>Compares formatted values, including blank padding</td>
</tr>
</tbody>
</table>

For more information about the syntax of the CODE statement, see the section “CODE Statement” on page 14 in Chapter 2, “Shared Concepts.”

DISPLAY Statement

DISPLAY < table-list > < / options > ;

The DISPLAY statement enables you to specify a list of ODS 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, all ODS tables are sent to the client and then the client creates a subset. If both DISPLAY and ODS statements are used
together, the DISPLAY statement takes precedence over the ODS statements. For more information about ODS, see SAS Output Delivery System: Procedures Guide.

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

**CASESENSITIVE**
performs a case-sensitive comparison of table names in the table-list to ODS 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 ODS tables except those specified in the table-list.

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

**TRACE**
displays the ODS table names, labels, and paths.

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

A path is a table name that is prefixed with dot-separated grouping information. For example, a SelectionSummary table that is produced by a procedure 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 ODS 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 starts with a “/” or a “!”. 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.

### DISPLAYOUT Statement

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

The DISPLAYOUT statement enables you to create CAS 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 CAS output tables to create. Each entry in the list has either a key or a key=value format:

- **key=value** specifies key as the ODS table name, path, or partial pathname, and specifies value as the CAS output table name.
- **key** specifies key as the ODS table name and also as the CAS output table name.
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 (/):

NOREPLACE
do not replace an existing CAS output table of the same name.

REPEATED
replicates the CAS output tables on all nodes.

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 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 the FREQ statement is not specified, each observation is assigned a frequency of 1.

The FREQ statement is not supported if you specify METHOD=NIPALS or METHOD=ITERGS in the PROC PCA statement.

OUTPUT Statement

OUTPUT OUT=\textit{CAS-libref.data-table}
\begin{itemize}
  \item \texttt{<COPYVARS=(variables)>}
  \item \texttt{<keyword =prefix>...<keyword =prefix> ;}
\end{itemize}

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 \texttt{COPYVARS=} option are included.

You must specify the following option:

\texttt{OUT=\textit{CAS-libref.data-table}}

names the output data table for PROC PCA to use. You must specify this option before any other options. \textit{CAS-libref.data-table} is a two-level name, where

\begin{itemize}
  \item \textit{CAS-libref} refers to a collection of information that is defined in the LIBNAME statement and includes the caslib, 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 \textit{CAS-libref}, see the section “Using CAS Sessions and CAS Engine Librefs” on page 259.
  \item \textit{data-table} specifies the name of the output data table.
\end{itemize}
You can also specify the following syntax elements:

\[ \text{COPYVAR} = \text{variable} \]
\[ \text{COPYVARS} = (\text{variables}) \]

transfers one or more variables from the input data table to the output data table.

\[ \text{keyword} < = \text{prefix} > \]

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 \( x_1 \) and \( x_2 \), 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 \( x_1 \) and \( x_2 \), then \( \text{RESIDUAL=R} \) produces the variables \( R_{x_1} \) and \( R_{x_2} \).

- The keyword \( \text{SCORE} \) 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 \( \text{SCORE=T} \) produces the variables \( T_1 \), \( T_2 \), and \( T_3 \).

- The keywords \( \text{H} \), \( \text{STDSSE} \), and \( \text{TSQUARE} \) 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 \( \text{OUTSTAT=} \) data table, the residual variables are named by prefixing either the characters \( R_\) (by default) or the string specified in the \( \text{PARPREFIX=} \) option to the VAR statement variable names.

The PARTIAL statement is not supported if you specify \( \text{METHOD=NIPALS} \) or \( \text{METHOD=ITERGS} \) in the PROC PCA 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 \( \text{variable} \) 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, all observations that are used in the analysis are assigned a weight of 1.

The WEIGHT statement is not supported if you specify \( \text{METHOD=NIPALS} \) or \( \text{METHOD=ITERGS} \) in the PROC PCA statement.
Chapter 7: The PCA Procedure

Details: PCA Procedure

Computing Principal Components

The PCA procedure implements several methods of calculating principal components: eigenvalue decomposition, the nonlinear iterative partial least squares (NIPALS) method, and the iterative method based on the Gram-Schmidt orthogonalization (ITERGS) of Andrecut (2009). Eigenvalue decomposition is more efficient when you want to calculate all principal components, whereas the NIPALS method is faster if you want to extract only the first few principal components. For high-dimensional data tables, the NIPALS method is more efficient, whereas it gets expensive for eigenvalue decomposition to calculate all the components simultaneously.

Eigenvalue Decomposition

Let $X$ 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 $X^T X$, 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 $X^T X$. 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 $X$. The NIPALS method starts by calculating the loadings, $p$, as $p_0 = (t_0' t_0)^{-1} t_0' X$, where $t$ is the score vector. It then calculates an improved score vector, $t = X p$. The method iteratively computes the improved $p$ and $t$ 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 $X$ with the residual from the first component: $E = X - t_0 p_0$.

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:

$$p_c = p - P_k P_k' p$$
$$t_c = t - T_k T_k' t$$

Here, $p_c$ and $t_c$ are the corrected loading vector and score vector, respectively. $P_k$ is the matrix that is formed by using the first $k$ loadings. $T_k$ 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.

**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 following table relates the type of the OUTSTAT= data table to the options that are specified in the PROC PCA statement:

<table>
<thead>
<tr>
<th>Options</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>CORR</td>
</tr>
<tr>
<td>COV</td>
<td>COV</td>
</tr>
<tr>
<td>NOINT</td>
<td>UCORR</td>
</tr>
<tr>
<td>COV NOINT</td>
<td>UCOV</td>
</tr>
</tbody>
</table>

Note that the default (neither the COV nor NOINT option) produces a data table of type CORR.

The new 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:

<table>
<thead>
<tr>
<th><em>TYPE</em></th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEAN</td>
<td>mean of each variable. If you specify the PARTIAL statement, this observation is omitted.</td>
</tr>
<tr>
<td>STD</td>
<td>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.</td>
</tr>
<tr>
<td>USTD</td>
<td>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.</td>
</tr>
</tbody>
</table>
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'.

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=NIPALS or METHOD=ITERGS, 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=NIPALS or METHOD=ITERGS, 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=NIPALS or METHOD=ITERGS), 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=NIPALS or METHOD=ITERGS, 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.
**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 7.5.

### Table 7.5  ODS Tables Produced by PROC PCA

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Required Statement / Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>CenScaleInfo</td>
<td>Centering and scaling information</td>
<td>METHOD=NIPALS</td>
</tr>
<tr>
<td>Corr</td>
<td>Correlation matrix</td>
<td>METHOD=EIG</td>
</tr>
<tr>
<td>Cov</td>
<td>Covariance matrix</td>
<td>METHOD=EIG and COV</td>
</tr>
<tr>
<td>Eigenvalues</td>
<td>Eigenvalues</td>
<td>Default output</td>
</tr>
<tr>
<td>Eigenvectors</td>
<td>Eigenvectors</td>
<td>METHOD=EIG</td>
</tr>
<tr>
<td>Loadings</td>
<td>Loadings</td>
<td>Default output</td>
</tr>
<tr>
<td>ModelInfo</td>
<td>Model information</td>
<td>Default output</td>
</tr>
<tr>
<td>NObs</td>
<td>Number of observations read and used</td>
<td>Default output</td>
</tr>
<tr>
<td>NVars</td>
<td>Number of variables, partial variables, and</td>
<td>Default output</td>
</tr>
<tr>
<td></td>
<td>principal components</td>
<td></td>
</tr>
<tr>
<td>OutputCasTables</td>
<td>Library and name of output data tables, and</td>
<td>DISPLAYOUT statement</td>
</tr>
<tr>
<td></td>
<td>number of rows and columns in the table</td>
<td></td>
</tr>
<tr>
<td>ParCorr</td>
<td>Partial correlation matrix</td>
<td>PARTIAL statement</td>
</tr>
<tr>
<td>ParCov</td>
<td>Uncorrected partial covariance matrix</td>
<td>PARTIAL statement and COV</td>
</tr>
<tr>
<td>RegCoef</td>
<td>Regression coefficients</td>
<td>PARTIAL statement and COV</td>
</tr>
<tr>
<td>RSquareRMSE</td>
<td>Regression statistics: R-squares and RMSEs</td>
<td>PARTIAL statement</td>
</tr>
<tr>
<td>SimpleStatistics</td>
<td>Simple statistics</td>
<td>METHOD=EIG</td>
</tr>
<tr>
<td>StdRegCoef</td>
<td>Standardized regression coefficients</td>
<td>PARTIAL statement</td>
</tr>
<tr>
<td>Timing</td>
<td>Absolute and relative times of tasks that are</td>
<td>Default output</td>
</tr>
<tr>
<td></td>
<td>performed by the procedure</td>
<td></td>
</tr>
<tr>
<td>TotalVariance</td>
<td>Total variance</td>
<td>METHOD=EIG and COV</td>
</tr>
<tr>
<td>Variation</td>
<td>Explained variation of variables</td>
<td>METHOD=NIPALS</td>
</tr>
</tbody>
</table>

### 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 7.6, along with a description of each graph and the required statements and options.

### Table 7.6  Graphics Produced by PROC PCA

<table>
<thead>
<tr>
<th>ODS Graph Name</th>
<th>Plot Description</th>
<th>Required Statement / Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>PatternProfilePlot</td>
<td>Component pattern profile plot</td>
<td>PLOTS=PARTNERNPROFILE</td>
</tr>
<tr>
<td>ScreePlot</td>
<td>Scree and variance plots</td>
<td>Default output</td>
</tr>
<tr>
<td>VariancePlot</td>
<td>Variance proportion-explained plot</td>
<td>PLOTS=SCREE(UNPACKPANEL)</td>
</tr>
</tbody>
</table>
Examples: PCA 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 8 and “Loading a SAS Data Set onto a CAS Server” on page 9 in Chapter 2, “Shared Concepts.”

### Example 7.1: Analyzing Mean Temperatures of US Cities

This example analyzes mean daily temperatures of selected US cities in January and July. The following DATA step creates the data:

```sas
data mycas.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).

```sas
title 'Mean Temperature of Selected Cities in January and July';
proc pca data=mycas.Temperature cov;
var July January;
output out=mycas.Scores copyVars=Cityid;
run;
```

Output 7.1.1 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.
Example 7.2: Extracting Principal Components with NIPALS

Output 7.1.1 Results of Principal Component Analysis

Mean Temperature of Selected Cities in January and July

The PCA Procedure

<table>
<thead>
<tr>
<th>Model Information</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Source</td>
<td>TEMPERATURE</td>
</tr>
<tr>
<td>Component Extraction Method</td>
<td>Eigenvalue Decomposition</td>
</tr>
</tbody>
</table>

| Number of Variables | 2 |
| Number of Principal Components | 2 |
| Number of Observations Read | 64 |
| Number of Observations Used | 64 |

Simple Statistics

<table>
<thead>
<tr>
<th>Variable</th>
<th>Mean</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>July</td>
<td>75.60781</td>
<td>5.12762</td>
</tr>
<tr>
<td>January</td>
<td>32.09531</td>
<td>11.71243</td>
</tr>
</tbody>
</table>

Covariance Matrix

<table>
<thead>
<tr>
<th>Variable</th>
<th>July</th>
<th>January</th>
</tr>
</thead>
<tbody>
<tr>
<td>July</td>
<td>26.29248</td>
<td>46.82829</td>
</tr>
<tr>
<td>January</td>
<td>46.82829</td>
<td>137.18109</td>
</tr>
</tbody>
</table>

Total Variance: 163.47356647

Eigenvalues of the Covariance Matrix

<table>
<thead>
<tr>
<th>Eigenvalue</th>
<th>Difference</th>
<th>Proportion</th>
<th>Cumulative</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>154.310607</td>
<td>145.147647</td>
<td>0.9439</td>
</tr>
<tr>
<td>2</td>
<td>9.162960</td>
<td>0.0561</td>
<td>1.0000</td>
</tr>
</tbody>
</table>

Eigenvectors

<table>
<thead>
<tr>
<th>Variable</th>
<th>Prin1</th>
<th>Prin2</th>
</tr>
</thead>
<tbody>
<tr>
<td>July</td>
<td>0.34353</td>
<td>-0.93914</td>
</tr>
<tr>
<td>January</td>
<td>0.93914</td>
<td>0.34353</td>
</tr>
</tbody>
</table>

Example 7.2: Extracting Principal Components with NIPALS

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:

```plaintext
data mycas.Crime;
title 'Crime Rates per 100,000 Population by State';
input State $1-15 Murder Rape Robbery Assault Burglary Larceny Auto_Theft;
datalines;
```
The following statements use PROC PCA to extract principal components by using the NIPALS method, and produce default plots:

```plaintext
ods graphics on;
proc pca data=mycas.Crime method=nipals plots;
run;
```

Output 7.2.1 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 7.2.1**  Results of Principal Component Analysis Using NIPALS

**Crime Rates per 100,000 Population by State**

<table>
<thead>
<tr>
<th>State</th>
<th>Variable 1</th>
<th>Variable 2</th>
<th>Variable 3</th>
<th>Variable 4</th>
<th>Variable 5</th>
<th>Variable 6</th>
<th>Variable 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alabama</td>
<td>14.2</td>
<td>25.2</td>
<td>96.8</td>
<td>278.3</td>
<td>1135.5</td>
<td>1881.9</td>
<td>280.7</td>
</tr>
<tr>
<td>Alaska</td>
<td>10.8</td>
<td>51.6</td>
<td>96.8</td>
<td>284.0</td>
<td>1331.7</td>
<td>3369.8</td>
<td>753.3</td>
</tr>
<tr>
<td>Arizona</td>
<td>9.5</td>
<td>34.2</td>
<td>138.2</td>
<td>312.3</td>
<td>2346.1</td>
<td>4467.4</td>
<td>439.5</td>
</tr>
<tr>
<td>Arkansas</td>
<td>8.8</td>
<td>27.6</td>
<td>83.2</td>
<td>203.4</td>
<td>972.6</td>
<td>1862.1</td>
<td>183.4</td>
</tr>
<tr>
<td>California</td>
<td>11.5</td>
<td>49.4</td>
<td>287.0</td>
<td>358.0</td>
<td>2139.4</td>
<td>3499.8</td>
<td>663.5</td>
</tr>
<tr>
<td>Wisconsin</td>
<td>2.8</td>
<td>12.9</td>
<td>52.2</td>
<td>63.7</td>
<td>846.9</td>
<td>2614.2</td>
<td>220.7</td>
</tr>
<tr>
<td>Wyoming</td>
<td>.</td>
<td>21.9</td>
<td>39.7</td>
<td>173.9</td>
<td>811.6</td>
<td>2772.2</td>
<td>282.0</td>
</tr>
</tbody>
</table>

... more lines ...
Example 7.2: Extracting Principal Components with NIPALS

Output 7.2.1 continued

Centering and Scaling Information

<table>
<thead>
<tr>
<th>Variable</th>
<th>Subtracted off</th>
<th>Divided by</th>
</tr>
</thead>
<tbody>
<tr>
<td>Murder</td>
<td>7.51667</td>
<td>3.93059</td>
</tr>
<tr>
<td>Rape</td>
<td>26.07500</td>
<td>10.81304</td>
</tr>
<tr>
<td>Robbery</td>
<td>127.55625</td>
<td>88.49374</td>
</tr>
<tr>
<td>Assault</td>
<td>214.58750</td>
<td>100.64360</td>
</tr>
<tr>
<td>Burglary</td>
<td>1316.37917</td>
<td>423.31261</td>
</tr>
<tr>
<td>Larceny</td>
<td>2696.88542</td>
<td>714.75023</td>
</tr>
<tr>
<td>Auto_Theft</td>
<td>383.97917</td>
<td>194.37033</td>
</tr>
</tbody>
</table>

Output 7.2.2

PROC PCA produces the scree plot as shown in Output 7.2.2 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.

Explained Variation of Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Prin1</th>
<th>Prin2</th>
<th>Prin3</th>
<th>Prin4</th>
<th>Prin5</th>
<th>Prin6</th>
<th>Prin7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Murder</td>
<td>0.37117</td>
<td>0.85539</td>
<td>0.87790</td>
<td>0.89562</td>
<td>0.97555</td>
<td>0.99143</td>
<td>1.00000</td>
</tr>
<tr>
<td>Rape</td>
<td>0.76242</td>
<td>0.79917</td>
<td>0.84059</td>
<td>0.84199</td>
<td>0.85065</td>
<td>0.99041</td>
<td>1.00000</td>
</tr>
<tr>
<td>Robbery</td>
<td>0.63783</td>
<td>0.64064</td>
<td>0.82164</td>
<td>0.92942</td>
<td>0.99788</td>
<td>0.99992</td>
<td>1.00000</td>
</tr>
<tr>
<td>Assault</td>
<td>0.63517</td>
<td>0.79127</td>
<td>0.79341</td>
<td>0.91781</td>
<td>0.98822</td>
<td>0.99513</td>
<td>1.00000</td>
</tr>
<tr>
<td>Burglary</td>
<td>0.78913</td>
<td>0.84414</td>
<td>0.88183</td>
<td>0.88207</td>
<td>0.88544</td>
<td>0.94800</td>
<td>1.00000</td>
</tr>
<tr>
<td>Larceny</td>
<td>0.51373</td>
<td>0.72178</td>
<td>0.93718</td>
<td>0.95479</td>
<td>0.95492</td>
<td>0.95530</td>
<td>1.00000</td>
</tr>
<tr>
<td>Auto_Theft</td>
<td>0.33638</td>
<td>0.65746</td>
<td>0.90481</td>
<td>0.96197</td>
<td>0.99623</td>
<td>0.99706</td>
<td>1.00000</td>
</tr>
</tbody>
</table>

Eigenvalues

<table>
<thead>
<tr>
<th>Eigenvalue</th>
<th>Difference</th>
<th>Proportion</th>
<th>Cumulative</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 4.045824</td>
<td>2.781795</td>
<td>0.5780</td>
<td>0.5780</td>
</tr>
<tr>
<td>2 1.264030</td>
<td>0.516529</td>
<td>0.1806</td>
<td>0.7586</td>
</tr>
<tr>
<td>3 0.747500</td>
<td>0.421175</td>
<td>0.1068</td>
<td>0.8653</td>
</tr>
<tr>
<td>4 0.326325</td>
<td>0.061119</td>
<td>0.0466</td>
<td>0.9120</td>
</tr>
<tr>
<td>5 0.265207</td>
<td>0.036843</td>
<td>0.0379</td>
<td>0.9498</td>
</tr>
<tr>
<td>6 0.228364</td>
<td>0.105613</td>
<td>0.0326</td>
<td>0.9825</td>
</tr>
<tr>
<td>7 0.122750</td>
<td>0.0175</td>
<td>1.0000</td>
<td></td>
</tr>
</tbody>
</table>

Loadings

<table>
<thead>
<tr>
<th>Variable</th>
<th>Prin1</th>
<th>Prin2</th>
<th>Prin3</th>
<th>Prin4</th>
<th>Prin5</th>
<th>Prin6</th>
<th>Prin7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Murder</td>
<td>0.30289</td>
<td>-0.61893</td>
<td>0.17353</td>
<td>-0.23308</td>
<td>0.54896</td>
<td>-0.26371</td>
<td>-0.26428</td>
</tr>
<tr>
<td>Rape</td>
<td>0.43410</td>
<td>-0.17053</td>
<td>-0.23539</td>
<td>0.06540</td>
<td>0.18075</td>
<td>0.78232</td>
<td>0.27946</td>
</tr>
<tr>
<td>Robbery</td>
<td>0.39705</td>
<td>0.04713</td>
<td>0.49208</td>
<td>-0.57470</td>
<td>-0.50808</td>
<td>0.09452</td>
<td>0.02497</td>
</tr>
<tr>
<td>Assault</td>
<td>0.39622</td>
<td>-0.35142</td>
<td>-0.05343</td>
<td>0.61744</td>
<td>-0.51525</td>
<td>-0.17395</td>
<td>-0.19921</td>
</tr>
<tr>
<td>Burglary</td>
<td>0.44164</td>
<td>0.20861</td>
<td>-0.22454</td>
<td>-0.02750</td>
<td>0.11273</td>
<td>-0.52340</td>
<td>0.65085</td>
</tr>
<tr>
<td>Larceny</td>
<td>0.35634</td>
<td>0.40570</td>
<td>-0.53681</td>
<td>-0.23231</td>
<td>0.02172</td>
<td>-0.04085</td>
<td>-0.60346</td>
</tr>
<tr>
<td>Auto_Theft</td>
<td>0.28834</td>
<td>0.50400</td>
<td>0.57524</td>
<td>0.41853</td>
<td>0.35939</td>
<td>0.06024</td>
<td>-0.15487</td>
</tr>
</tbody>
</table>
In addition to the scree plot, PROC PCA also produces the pattern profile plot. The following statements request this plot:

```
proc pca data=mycas.Crime method=nipals plots=patternprofile;
run;
```

Output 7.2.3 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 7.2.3 Component Pattern Profile Plot from the PCA Procedure

References


Chapter 8
The PLSMOD Procedure

Contents

Overview: PLSMOD Procedure ................................................................. 288
PROC PLSMOD Features ................................................................. 288
PROC PLSMOD Compared with Other SAS Procedures .......... 289
   PROC PLSMOD Compared with the HPPLS Procedure .......... 289
   PROC PLSMOD Compared with the PLS Procedure .......... 290
Using CAS Sessions and CAS Engine Librefs .................. 291
Getting Started: PLSMOD Procedure ........................................... 292
   Fitting a PLS Model ............................................................... 293
   Selecting the Number of Factors by Test Set Validation .... 295
   Predicting New Observations ................................................ 299
Syntax: PLSMOD Procedure ............................................................... 300
   PROC PLSMOD Statement ....................................................... 300
   CLASS Statement ................................................................. 303
   DISPLAY Statement .............................................................. 304
   DISPLAYOUT Statement ......................................................... 304
   EFFECT Statement ............................................................... 305
   MODEL Statement ............................................................... 307
   OUTPUT Statement .............................................................. 307
   PARTITION Statement .......................................................... 310
Details: PLSMOD Procedure .............................................................. 310
   Regression Methods ............................................................. 310
      Partial Least Squares ....................................................... 311
      SIMPLS ................................................................. 311
      Principal Component Regression ................................... 312
      Reduced Rank Regression ............................................. 312
      Relationships between Methods .................................... 312
   Test Set Validation ............................................................. 315
   Centering and Scaling ........................................................ 316
   Missing Values ................................................................. 317
   Displayed Output ............................................................... 317
      Model Information .......................................................... 317
      Dimensions ................................................................. 317
      Number of Observations ................................................ 317
      Class Level Information ............................................... 318
      Centering and Scaling Information ................................. 318
      Test Set Validation ........................................................ 318
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
The PLSMOD procedure 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).

The PLSMOD procedure 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 68 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 8.1.
Table 8.1  Comparison of PROC PLSMOD and PROC HPPLS

<table>
<thead>
<tr>
<th>Feature</th>
<th>PROC PLSMOD</th>
<th>PROC HPPLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supports general factor extraction methods</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Supports EFFECT statement</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Supports DISPLAY and DISPLAYOUT statements</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Supports BY and ID statements</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Supports MISSING=, VARSCALE, and PLOTS options</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Supports cross validation methods</td>
<td>Only test set validation by using the PARTITION statement</td>
<td>Only test set validation by using the PARTITION statement</td>
</tr>
<tr>
<td>Includes input variables in output data table</td>
<td>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.</td>
<td>Not by default, in order to avoid data duplication for large data sets. To include input variables, specify them in the ID statement.</td>
</tr>
<tr>
<td>Supports parameterization of classification variables</td>
<td>Full set of parameterizations (supports all values of the PARAM= option in the CLASS statement)</td>
<td>Supports only PARAM=GLM or PARAM=REFERENCE</td>
</tr>
<tr>
<td>Threading</td>
<td>Specifically designed for CAS; executes on multiple threads</td>
<td>Primarily designed for a distributed environment; executes on multiple threads</td>
</tr>
</tbody>
</table>

PROC PLSMOD Compared with the PLS Procedure

The PLSMOD procedure and the PLS procedure have the similarities and differences shown in Table 8.2.

Table 8.2  Comparison of PROC PLSMOD and PROC PLS

<table>
<thead>
<tr>
<th>Feature</th>
<th>PROC PLSMOD</th>
<th>PROC PLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supports general factor extraction methods</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Supports EFFECT statement</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Supports DISPLAY and DISPLAYOUT statements</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Supports BY and ID statements</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Supports MISSING=, VARSCALE, and PLOTS options</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Supports cross validation methods</td>
<td>Only test set validation by using the PARTITION statement</td>
<td>Various methods</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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:

```sas
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:

```sas
cas mysess;
libname mycas cas sessref=mysess;
```

The CAS statement creates the CAS session named `mysess`, and the LIBNAME statement creates the `mycas` 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:

```sas
Using CAS Sessions and CAS Engine Librefs  ♦  291

Table 8.2  continued

<table>
<thead>
<tr>
<th>Feature</th>
<th>PROC PLSMOD</th>
<th>PROC PLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax of OUTPUT statement</td>
<td>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.</td>
<td>You must provide keyword and prefix</td>
</tr>
<tr>
<td>Includes input variables in output data table</td>
<td>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.</td>
<td>Yes</td>
</tr>
<tr>
<td>Supports parameterization of classification variables</td>
<td>Full set of parameterizations (supports all values of the PARAM= option in the CLASS statement)</td>
<td>Only GLM parameterization</td>
</tr>
<tr>
<td>Threading</td>
<td>Specifically designed for CAS; executes on multiple threads</td>
<td>Executes on a single thread</td>
</tr>
</tbody>
</table>

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:

```sas
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:

```sas
cas mysess;
libname mycas cas sessref=mysess;
```

The CAS statement creates the CAS session named `mysess`, and the LIBNAME statement creates the `mycas` 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 8 in Chapter 2, “Shared Concepts.”

---

### Getting Started: PLSMOD Procedure

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 8 and “Loading a SAS Data Set onto a CAS Server” on page 9 in Chapter 2, “Shared Concepts.”

The following DATA step creates the mycas.Sample data table, which provides the calibration data, in your CAS session. These statements assume that your CAS engine libref is named mycas, but you can substitute any appropriately defined CAS engine libref.

``` SAS
data mycas.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
```

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=mycas.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 8.1 and Figure 8.2.

Figure 8.1 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 8.1** Model Information, Dimensions, and Number of Observations

<table>
<thead>
<tr>
<th>The PLSMOD Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Model Information</strong></td>
</tr>
<tr>
<td>Data Source</td>
</tr>
<tr>
<td>Factor Extraction Method</td>
</tr>
<tr>
<td>PLS Algorithm</td>
</tr>
<tr>
<td>Validation Method</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Dimensions</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Response Variables</td>
</tr>
<tr>
<td>Number of Effects</td>
</tr>
<tr>
<td>Number of Predictor Parameters</td>
</tr>
<tr>
<td>Number of Factors</td>
</tr>
</tbody>
</table>

| Number of Observations Read | 16  |
| Number of Observations Used | 16  |

Figure 8.2 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.
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:

```sas
proc plsmod data=mycas.sample;
   model ls ha dt = v1-v27;
   partition roleVar = Role(train='TRAIN' test='TEST');
run;
```

The resulting output is shown in Figure 8.3 through Figure 8.5.
Figure 8.3  Model Information, Dimensions, and Number of Observations with Test Set Validation

The PLSMOD Procedure

Model Information

<table>
<thead>
<tr>
<th>Data Source</th>
<th>SAMPLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Factor Extraction Method</td>
<td>Partial Least Squares</td>
</tr>
<tr>
<td>PLS Algorithm</td>
<td>NIPALS</td>
</tr>
<tr>
<td>Validation Method</td>
<td>Test Set Validation</td>
</tr>
</tbody>
</table>

Dimensions

<table>
<thead>
<tr>
<th>Number of Response Variables</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Effects</td>
<td>27</td>
</tr>
<tr>
<td>Number of Predictor Parameters</td>
<td>27</td>
</tr>
<tr>
<td>Maximum Number of Factors</td>
<td>9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Number of Observations Read</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Observations Used</td>
<td>16</td>
</tr>
<tr>
<td>Number of Observations Used for Training</td>
<td>9</td>
</tr>
<tr>
<td>Number of Observations Used for Testing</td>
<td>7</td>
</tr>
</tbody>
</table>

Figure 8.4  Test-Set-Validated PRESS Statistics for Number of Factors

The PLSMOD Procedure

<table>
<thead>
<tr>
<th>Test Set Validation for the Number of Extracted Factors</th>
<th>Root Mean PRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Extracted Factors</td>
<td>Root Mean PRESS</td>
</tr>
<tr>
<td>--------------------------------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>0</td>
<td>1.426362</td>
</tr>
<tr>
<td>1</td>
<td>1.276694</td>
</tr>
<tr>
<td>2</td>
<td>1.181752</td>
</tr>
<tr>
<td>3</td>
<td>0.656999</td>
</tr>
<tr>
<td>4</td>
<td>0.43457</td>
</tr>
<tr>
<td>5</td>
<td>0.420916</td>
</tr>
<tr>
<td>6</td>
<td>0.585031</td>
</tr>
<tr>
<td>7</td>
<td>0.576586</td>
</tr>
<tr>
<td>8</td>
<td>0.563935</td>
</tr>
<tr>
<td>9</td>
<td>0.563935</td>
</tr>
</tbody>
</table>

Minimum Root Mean PRESS 0.420916
Minimizing Number of Factors 5
In Figure 8.3, 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 8.4 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:

```sas
proc plsmod data=mycas.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 8.6 through Figure 8.8.

---

**Figure 8.5** PLS Variation Summary for Test-Set-Validated Model

<table>
<thead>
<tr>
<th>Number of Extracted Factors</th>
<th>Model Effects</th>
<th>Response Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Current</td>
<td>Total</td>
</tr>
<tr>
<td>1</td>
<td>95.92495</td>
<td>95.92495</td>
</tr>
<tr>
<td>2</td>
<td>3.86407</td>
<td>99.78903</td>
</tr>
<tr>
<td>3</td>
<td>0.10170</td>
<td>99.89073</td>
</tr>
<tr>
<td>4</td>
<td>0.08979</td>
<td>99.98052</td>
</tr>
<tr>
<td>5</td>
<td>0.01142</td>
<td>99.99194</td>
</tr>
</tbody>
</table>

---

**Figure 8.6** Model Information with Model Comparison Test

**The PLSMOD Procedure**

<table>
<thead>
<tr>
<th>Model Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Source</td>
</tr>
<tr>
<td>Factor Extraction Method</td>
</tr>
<tr>
<td>PLS Algorithm</td>
</tr>
<tr>
<td>Validation Method</td>
</tr>
<tr>
<td>Validation Testing Criterion</td>
</tr>
<tr>
<td>Number of Random Permutations</td>
</tr>
<tr>
<td>Random Number Seed for Permutation</td>
</tr>
</tbody>
</table>
Chapter 8: The PLSMOD Procedure

Figure 8.7 Testing Test Set Validation for Number of Factors

The PLSMOD Procedure

Test Set Validation for the Number of Extracted Factors

<table>
<thead>
<tr>
<th>Number of Extracted Factors</th>
<th>Root Mean PRESS</th>
<th>Prob &gt; T**2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.426362</td>
<td>5.191629</td>
</tr>
<tr>
<td>1</td>
<td>1.276694</td>
<td>6.174825</td>
</tr>
<tr>
<td>2</td>
<td>1.181752</td>
<td>4.60203</td>
</tr>
<tr>
<td>3</td>
<td>0.656999</td>
<td>3.09999</td>
</tr>
<tr>
<td>4</td>
<td>0.43457</td>
<td>4.980227</td>
</tr>
<tr>
<td>5</td>
<td>0.420916</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0.585031</td>
<td>2.05496</td>
</tr>
<tr>
<td>7</td>
<td>0.576586</td>
<td>3.009172</td>
</tr>
<tr>
<td>8</td>
<td>0.563935</td>
<td>2.416635</td>
</tr>
<tr>
<td>9</td>
<td>0.563935</td>
<td>2.416635</td>
</tr>
</tbody>
</table>

Minimum Root Mean PRESS 0.420916
Minimizing Number of Factors 5
Smallest Number of Factors with p > 0.15 3

Figure 8.8 PLS Variation Summary for Tested Test-Set-Validated Model

Percentage Variation Accounted for by Partial Least Squares Factors

<table>
<thead>
<tr>
<th>Number of Extracted Factors</th>
<th>Model Effects</th>
<th>Response Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current</td>
<td>Total</td>
<td>Current</td>
</tr>
<tr>
<td>1</td>
<td>95.92495</td>
<td>95.92495</td>
</tr>
<tr>
<td>2</td>
<td>3.86407</td>
<td>99.78903</td>
</tr>
<tr>
<td>3</td>
<td>0.10170</td>
<td>99.89073</td>
</tr>
</tbody>
</table>

The “Model Information” table in Figure 8.6 displays information about the options that are used in the model comparison test. In Figure 8.7, 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 8.8 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:

```sas
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:

```sas
data mycas.all;
  set mycas.sample newobs;
run;

proc plsmod data=mycas.all nfac=2;
  model ls ha dt = v1-v27;
  partition roleVar = Role(train='TRAIN' test='TEST');
  output out=mycas.result pred=p copyvars=(obsnam);
run;

proc print data=mycas.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 8.9.

**Figure 8.9** Predicted Concentrations for New Observations

<table>
<thead>
<tr>
<th>Obs</th>
<th>obsnam</th>
<th>p_ls</th>
<th>p_ha</th>
<th>p_dt</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>EM17</td>
<td>2.63326</td>
<td>0.22343</td>
<td>80.2027</td>
</tr>
<tr>
<td>2</td>
<td>EM25</td>
<td>0.69865</td>
<td>0.14308</td>
<td>98.9937</td>
</tr>
</tbody>
</table>
Syntax: PLSMOD Procedure

The following statements are available in the PLSMOD procedure:

```plaintext
PROC PLSMOD < options > ;
  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 = CAS-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

```plaintext
PROC PLSMOD < options > ;
```

The PROC PLSMOD statement invokes the PLSMOD procedure. Table 8.3 summarizes the options available in the PROC PLSMOD statement.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Basic Options</strong></td>
<td></td>
</tr>
<tr>
<td>DATA=</td>
<td>Specifies the CAS input data table</td>
</tr>
<tr>
<td><strong>Model Fitting Options</strong></td>
<td></td>
</tr>
<tr>
<td>CVTEST</td>
<td>Requests that van der Voet’s (1994) randomization-based model comparison test be performed</td>
</tr>
<tr>
<td>METHOD=</td>
<td>Specifies the general factor extraction method to be used</td>
</tr>
<tr>
<td>NFAC=</td>
<td>Specifies the number of factors to extract</td>
</tr>
<tr>
<td>NOCENTER</td>
<td>Suppresses centering of the responses and predictors before fitting</td>
</tr>
<tr>
<td>NOCVSTDIZE</td>
<td>Suppresses re-centering and rescaling of the responses and predictors when cross validating</td>
</tr>
<tr>
<td>NOSCALE</td>
<td>Suppresses scaling of the responses and predictors before fitting</td>
</tr>
<tr>
<td><strong>Output Options</strong></td>
<td></td>
</tr>
<tr>
<td>CENSCALE</td>
<td>Displays the centering and scaling information</td>
</tr>
<tr>
<td>DETAILS</td>
<td>Displays the details of the fitted model</td>
</tr>
<tr>
<td>NOCLPRINT</td>
<td>Limits or suppresses the display of class levels</td>
</tr>
</tbody>
</table>
Table 8.3  continued

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VARSS</td>
<td>Displays the amount of variation accounted for in each response and predictor</td>
</tr>
</tbody>
</table>

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 315. 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=CAS-libref.data-table**
names the input data table for PROC PLSMOD to use. The default is the most recently created data table. `CAS-libref.data-table` is a two-level name, where

  - **CAS-libref** refers to a collection of information that is defined in the LIBNAME statement and includes the caslib, 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 `CAS-libref`, see the section “Using CAS Sessions and CAS Engine Librefs” on page 291.

  - **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 317.

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 $X'Y$. This algorithm is the most accurate but least efficient approach.

EIG
bases the extraction on the eigenvalue decomposition of $Y'XX'Y$.

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 316.

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 316.

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 316.

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.

### 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 8.4 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 10 in Chapter 2, “Shared Concepts.”

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DESCENDING</td>
<td>Reverses the sort order</td>
</tr>
<tr>
<td>MISSING</td>
<td>Treats missing values as valid levels</td>
</tr>
<tr>
<td>ORDER=</td>
<td>Specifies the sort order for the levels</td>
</tr>
<tr>
<td>PARAM=</td>
<td>Specifies the parameterization of the variable</td>
</tr>
<tr>
<td>REF=</td>
<td>Specifies the reference level of the variable</td>
</tr>
<tr>
<td>SPLIT</td>
<td>Allows design columns for a variable to enter or leave the model independently</td>
</tr>
</tbody>
</table>
Chapter 8: The PLSMOD Procedure

**DISPLAY Statement**

```plaintext
DISPLAY < table-list > < / options > ;
```

The DISPLAY statement enables you to specify a list of ODS 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, all ODS tables are sent to the client and then the client creates a subset. If both DISPLAY and ODS statements are used together, the DISPLAY statement takes precedence over the ODS statements. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

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

- **CASESENSITIVE**
  - performs a case-sensitive comparison of table names in the `table-list` to ODS 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 ODS tables except those specified in the `table-list`.

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

- **TRACE**
  - displays the ODS table names, labels, and paths.

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

A path is a table name that is prefixed with dot-separated grouping information. For example, a SelectionSummary table that is produced by a procedure during a selection routine might have the path `Bygroup1.Summary.SelectionSummary`. A partial pathname does not include all groups; for example, `Selection-` and `Summary.SelectionSummary` are partial pathnames for `Bygroup1.Summary.SelectionSummary`.

When you specify a table name or partial pathname, all ODS 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 starts with a “/” or a “!”. 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.

**DISPLAYOUT Statement**

```plaintext
DISPLAYOUT table-spec-list < / options > ;
```
The DISPLAYOUT statement enables you to create CAS 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 CAS output tables to create. Each entry in the list has either a `key` or a `key=value` format:

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

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 (/):

- **NOREPLACE**
  - does not replace an existing CAS output table of the same name.
- **REPEATED**
  - replicates the CAS 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 48 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 8.5 summarizes the `options` available in the EFFECT statement.
Table 8.5  EFFECT Statement Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Collection Effects Options</strong></td>
<td></td>
</tr>
<tr>
<td>DETAILS</td>
<td>Displays the constituents of the collection effect</td>
</tr>
<tr>
<td><strong>Multimember Effects Options</strong></td>
<td></td>
</tr>
<tr>
<td>DETAILS</td>
<td>Displays the levels of the multimember effect</td>
</tr>
<tr>
<td>NOEFFECT</td>
<td>Specifies that observations whose levels are all missing for the multimember variables should have 0 values in the corresponding design matrix columns</td>
</tr>
<tr>
<td>STDIZE</td>
<td>Standardizes the design matrix entries so that each observation has a sum of 1</td>
</tr>
<tr>
<td>WEIGHT=</td>
<td>Specifies the weight variable for the contributions of each classification effect</td>
</tr>
<tr>
<td><strong>Polynomial Effects Options</strong></td>
<td></td>
</tr>
<tr>
<td>DEGREE=</td>
<td>Specifies the degree of the polynomial</td>
</tr>
<tr>
<td>DETAILS</td>
<td>Displays details of the specified polynomial</td>
</tr>
<tr>
<td>MDEGREE=</td>
<td>Specifies the maximum degree of any variable in a term of the polynomial</td>
</tr>
<tr>
<td>NOSEPARATE</td>
<td>Treats the polynomial as a single effect with multiple degrees of freedom</td>
</tr>
<tr>
<td>STANDARDIZE=</td>
<td>Specifies centering and scaling suboptions for the variables that define the polynomial</td>
</tr>
<tr>
<td><strong>Spline Effects Options</strong></td>
<td></td>
</tr>
<tr>
<td>BASIS=</td>
<td>Specifies the type of basis (B-spline basis or truncated power function basis) for the spline effect</td>
</tr>
<tr>
<td>DATABOUNDARY</td>
<td>Uses the extremes of the data as boundary knots for a B-spline basis</td>
</tr>
<tr>
<td>DEGREE=</td>
<td>Specifies the degree of the spline effect</td>
</tr>
<tr>
<td>DETAILS</td>
<td>Displays the knots and locations for each spline basis function</td>
</tr>
<tr>
<td>KNOTMAX=</td>
<td>Requests equally spaced right-side boundary knots starting at the variables’ maximum and ending at the KNOTMAX= value</td>
</tr>
<tr>
<td>KNOTMETHOD=</td>
<td>Specifies how to construct the knots for the spline effect</td>
</tr>
<tr>
<td>KNOTMIN=</td>
<td>Requests equally spaced left-side boundary knots starting at the KNOTMAX= value and ending at the variables’ minimum value</td>
</tr>
<tr>
<td>NATURALCUBIC</td>
<td>Specifies a natural cubic spline basis for the spline effect</td>
</tr>
<tr>
<td>SEPARATE</td>
<td>Treats the spline basis for each variable as a separate effect when multiple variables are specified</td>
</tr>
<tr>
<td>SPLIT</td>
<td>Treats each design matrix column as a separate effect for selection methods</td>
</tr>
</tbody>
</table>

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 17 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 45 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=CAS-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=CAS-libref.data-table

names the output data table for PROC PLSMOD to use. You must specify this option before any other options. CAS-libref.data-table is a two-level name, where

CAS-libref

refers to a collection of information that is defined in the LIBNAME statement and includes the caslib, 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 CAS-libref, see the section “Using CAS Sessions and CAS Engine Librefs” on page 291.

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**
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 8.6 shows the interpretation of this variable for each observation.

<table>
<thead>
<tr>
<th>Value</th>
<th>Observation Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not used</td>
</tr>
<tr>
<td>1</td>
<td>Training</td>
</tr>
<tr>
<td>2</td>
<td>Testing</td>
</tr>
</tbody>
</table>

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 315. For an illustration, see Example 8.1.

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 \( X = X_0 \) be the centered and scaled matrix of predictors, and let \( Y = Y_0 \) be the centered and scaled matrix of response values. The PLS method starts with a linear combination \( t = X_0 w \) of the predictors, where \( t \) is called a score vector and \( w \) is its associated weight vector. The PLS method predicts both \( X_0 \) and \( Y_0 \) by regression on \( t \):

\[
\hat{X}_0 = tp', \quad \text{where} \quad p' = (t't)^{-1}t'X_0
\]

\[
\hat{Y}_0 = tc', \quad \text{where} \quad c' = (t't)^{-1}t'Y_0
\]

The vectors \( p \) and \( c \) are called the X- and Y-loadings, respectively.

The specific linear combination \( t = X_0 w \) is the combination that has maximum covariance \( t' u \) with some response linear combination \( u = Y_0 q \). Another characterization is that the X-weight, \( w \), and the Y-weight, \( q \), are proportional to the first left- and right-singular vectors, respectively, of the covariance matrix \( X_0'Y_0 \) or, equivalently, the first eigenvectors of \( X_0'Y_0'X_0 \) and \( Y_0'X_0X_0'Y_0 \), respectively.

This accounts for how the first PLS factor is extracted. The second factor is extracted in the same way by replacing \( X_0 \) and \( Y_0 \) with the X- and Y-residuals from the first factor:

\[
X_1 = X_0 - \hat{X}_0
\]

\[
Y_1 = Y_0 - \hat{Y}_0
\]

These residuals are also called the deflated X and 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 \( X_i \) 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 \( t_i = Xr_i \) in terms of the original (centered and scaled) predictors \( X \). The SIMPLS X-weight vectors \( r_i \) are similar to the eigenvectors of \( SS' = X'YY'X \), but they satisfy a different orthogonality condition. The \( r_1 \) vector is just the first eigenvector \( e_1 \) (so that the first SIMPLS score is the same as the first PLS score). However, the second eigenvector maximizes

\[
e_1'ss'e_2 \quad \text{subject to} \quad e_1'e_2 = 0
\]

whereas the second SIMPLS weight \( r_2 \) maximizes

\[
r_1'ss'r_2 \quad \text{subject to} \quad r_1'X'Xr_2 = t_1't_2 = 0
\]

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 \( B \) because the scores are all defined in terms of \( X \):
Chapter 8: The PLSMOD Procedure

\[
\hat{Y} = \sum_i t_i^c c_i'
\]

\[
= \sum_i Xr_i^c c_i'
\]

\[
= XB, \text{ where } B = RC'
\]

Principal Component Regression

Like the SIMPLS method, principal component regression (PCR) defines all the scores in terms of the original (centered and scaled) predictors \(X\). 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 \(X\) as possible; equivalently, the X-weights for the PCR method are the eigenvectors of the predictor covariance matrix \(X'X\). 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 \(Y\) in terms of the original predictors \(X\).

Reduced Rank Regression

As discussed in the preceding sections, partial least squares depends on selecting the factors \(t = Xw\) of the predictors and \(u = Yq\) of the responses that have maximum covariance, whereas principal component regression effectively ignores \(u\) and selects \(t\) to have maximum variance, subject to orthogonality constraints. In contrast, reduced rank regression selects \(u\) 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, \(q_i\), are the eigenvectors of the covariance matrix of the predicted responses that are predicted by ordinary least squares regression, and the X-scores are the projections of the Y-scores, \(Yq_i\), onto the X space.

Relationships between Methods

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
```

```
Regression Methods

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 mycas.artData;
set artData;
run;

proc plsmod data=mycas.artData nfac=1 method=rrr;
   model y = x1 x2;
run;

proc plsmod data=mycas.artData nfac=1 method=pcr;
   model y = x1 x2;
run;

proc plsmod data=mycas.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 8.10 through Figure 8.12.

Figure 8.10 Variation Explained by the First Reduced Rank Regression Factor

The PLSMOD Procedure

<table>
<thead>
<tr>
<th>Number of Extracted Factors</th>
<th>Current</th>
<th>Total</th>
<th>Current</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>15.06605</td>
<td>15.06605</td>
<td>100.00000</td>
<td>100.00000</td>
</tr>
</tbody>
</table>

Figure 8.11 Variation Explained by the First Principal Component Regression Factor

The PLSMOD Procedure

<table>
<thead>
<tr>
<th>Number of Extracted Factors</th>
<th>Current</th>
<th>Total</th>
<th>Current</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>92.99959</td>
<td>92.99959</td>
<td>9.37874</td>
<td>9.37874</td>
</tr>
</tbody>
</table>
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 8.13 illustrates how partial least squares balances the goals of explaining response and predictor variation in this case.
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 $x_1 = x_2$ direction, the response changes most in the orthogonal $x_1 = -x_2$ 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:

```plaintext
proc plsmod data=mycas.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.

```plaintext
proc plsmod data=mycas.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 $j$th predicted residual for response $k$ for the model that has $i$ extracted factors. Then, the PRESS statistic is $\sum_{jk} R_{i,jk}^2$. Also, let $i_{\text{min}}$ 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:

$$D_{i,jk} = R_{i,jk}^2 - R_{i_{\text{min}},jk}^2$$

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 $i_{\text{min}}$ factors; alternatively, van der Voet suggests Hotelling’s $T^2$ statistic $C_i = d_i' S_i^{-1} d_i$, where $d_i$ is the sum of the vectors $d_i,j = \{D_i,j_1, \ldots, D_i,j_{N_j}\}'$ and $S_i$ is the sum of squares and crossproducts matrix,

$$S_i = \sum_j d_i,j d_i,j'$$

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 $R_{i,jk}^2$ and $R_{i_{\text{min}},jk}^2$. 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 310.) 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 $\text{Time}$ and $\text{Temp}$ are two of the predictors, then scaling says that a change of $\text{std}(\text{Time})$ in $\text{Time}$ is approximately equivalent to a change of $\text{std}(\text{Temp})$ in $\text{Temp}$.

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 \( x_1^i \) and \( x_2^i \), then the default standardized \( i \)th value of the crossproduct is

\[
\frac{x_1^i x_2^i - \text{mean}_j(x_1^j x_2^j)}{\text{std}_j(x_1^j x_2^j)}
\]

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 \texttt{MISSING} option in the \texttt{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 \texttt{PARTITION} statement and specify the \texttt{ROLEVAR=} option, observations that contain missing values for the \texttt{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 \texttt{CLASS} statement. If you use the \texttt{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 \texttt{FRACTION} option in the \texttt{PARTITION} statement and the \texttt{CVTEST} option in the \texttt{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 \texttt{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.

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 8.7.

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Required Statement and Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVResults</td>
<td>Results of test set validation</td>
<td>PARTITION statement</td>
</tr>
<tr>
<td>CenScaleParms</td>
<td>Parameter estimates for centered and scaled data</td>
<td>SOLUTION option in MODEL statement</td>
</tr>
<tr>
<td>ClassInfo</td>
<td>Level information from the CLASS statement</td>
<td>CLASS statement</td>
</tr>
<tr>
<td>CodedCoef</td>
<td>Coded regression coefficients</td>
<td>DETAILS option in PROC PLSMOD statement</td>
</tr>
<tr>
<td>Dimensions</td>
<td>Model dimensions</td>
<td>Default output</td>
</tr>
<tr>
<td>ModelInfo</td>
<td>Model information</td>
<td>Default output</td>
</tr>
<tr>
<td>NObs</td>
<td>Number of observations read and used</td>
<td>Default output</td>
</tr>
<tr>
<td>OutputCasTables</td>
<td>Library and name of output data tables, and number of rows and columns in the table</td>
<td>DISPLAYOUT statement or OUTPUT statement</td>
</tr>
<tr>
<td>ParameterEstimates</td>
<td>Parameter estimates for raw data</td>
<td>SOLUTION option in MODEL statement</td>
</tr>
<tr>
<td>PercentVariation</td>
<td>Predictor and response variation that are accounted for by each factor</td>
<td>Default output</td>
</tr>
<tr>
<td>ResidualSummary</td>
<td>Residual summary from test set validation</td>
<td>PARTITION statement</td>
</tr>
<tr>
<td>Timing</td>
<td>Absolute and relative times spent by tasks that are performed by the procedure</td>
<td>Default output</td>
</tr>
<tr>
<td>XEffectCenScale</td>
<td>Centering and scaling information for predictor effects</td>
<td>CENSCALE option in PROC PLSMOD statement</td>
</tr>
<tr>
<td>XLoadings</td>
<td>Loadings for predictor effects</td>
<td>DETAILS option in PROC PLSMOD statement</td>
</tr>
</tbody>
</table>
Table 8.7  continued

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Required Statement and Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>XPercentVariation</td>
<td>Variation that is accounted for by each factor for predictor effects</td>
<td>VARSS option in PROC PLSMOD statement</td>
</tr>
<tr>
<td>XWeights</td>
<td>Weights for predictor effects</td>
<td>DETAILS option in PROC PLSMOD statement</td>
</tr>
<tr>
<td>YPercentVariation</td>
<td>Variation that is accounted for by each factor for responses</td>
<td>VARSS option in PROC PLSMOD statement</td>
</tr>
<tr>
<td>YVariableCenScale</td>
<td>Centering and scaling information for responses</td>
<td>CENSCALE option in PROC PLSMOD statement</td>
</tr>
<tr>
<td>YWeights</td>
<td>Weights for responses</td>
<td>DETAILS option in PROC PLSMOD statement</td>
</tr>
</tbody>
</table>

Examples: PLSMOD 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 8 and “Loading a SAS Data Set onto a CAS Server” on page 9 in Chapter 2. “Shared Concepts.”

Example 8.1: Choosing a PLS Model by Test Set Validation

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 8.1.1 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 $10^{-6}$. 
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.

```sas
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
```
Chapter 8: The PLSMOD Procedure

-0.453 -0.549 -0.658 -0.797 -0.878 -0.954 -1.060 -1.266
-1.516 -5.294 -4.823 -3.858 -2.827 -1.683 -1.216
-0.878 -0.954 -1.060 -1.266 -1.516

19mix35 TRAIN 0.00003 3E-7
-0.351 -1.804 -2.044 -2.269 -2.496 -2.714
-5.516 -5.294 -4.823 -3.858 -2.827 -1.683 -1.216
-0.878 -0.954 -1.060 -1.266 -1.516

21mix35 TRAIN 0.00003 7.5E-7
-0.351 -1.804 -2.044 -2.269 -2.496 -2.714
-5.516 -5.294 -4.823 -3.858 -2.827 -1.683 -1.216
-0.878 -0.954 -1.060 -1.266 -1.516

... more lines ...

-5.138 -5.463 -5.461 -5.461 -5.461 -5.461
-5.138 -5.463 -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
-4.668 -4.668 -4.865 -4.865 -5.109 -5.111

; data mycas.exlData;
  set exlData;
run;

The following statements fit a PLS model that has 10 factors:

proc plsmod data=mycas.exlData nfac=10;
  model tot_log tyr_log try_log = f1-f30;
run;

The “Model Information” table in Output 8.1.2 shows that no validation method is used. The “Number of Observations” table confirms that all 33 sample observations are used in the analysis.

Output 8.1.2 Model Information and Number of Observations

The PLSMOD Procedure

<table>
<thead>
<tr>
<th>Model Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Source</td>
</tr>
<tr>
<td>Factor Extraction Method</td>
</tr>
<tr>
<td>PLS Algorithm</td>
</tr>
<tr>
<td>Validation Method</td>
</tr>
</tbody>
</table>

| Number of Observations Read | 33 |
| Number of Observations Used | 33 |

The table in Output 8.1.3 indicates that only four or five factors are required to explain almost all the variation in both the predictors and the responses.
Example 8.1: Choosing a PLS Model by Test Set Validation

Output 8.1.3 Amount of Variation Explained

<table>
<thead>
<tr>
<th>Number of Extracted Factors</th>
<th>Model Effects</th>
<th>Response Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current</td>
<td>Total</td>
<td>Current</td>
</tr>
<tr>
<td>1</td>
<td>77.67903</td>
<td>77.67903</td>
</tr>
<tr>
<td>2</td>
<td>20.62719</td>
<td>98.30622</td>
</tr>
<tr>
<td>3</td>
<td>1.00143</td>
<td>99.30766</td>
</tr>
<tr>
<td>4</td>
<td>0.24930</td>
<td>99.55696</td>
</tr>
<tr>
<td>5</td>
<td>0.13077</td>
<td>99.68773</td>
</tr>
<tr>
<td>6</td>
<td>0.08970</td>
<td>99.77742</td>
</tr>
<tr>
<td>7</td>
<td>0.05684</td>
<td>99.83426</td>
</tr>
<tr>
<td>8</td>
<td>0.06730</td>
<td>99.90156</td>
</tr>
<tr>
<td>9</td>
<td>0.01521</td>
<td>99.91676</td>
</tr>
<tr>
<td>10</td>
<td>0.02627</td>
<td>99.94304</td>
</tr>
</tbody>
</table>

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=mycas.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 8.1.4 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 8.1.4 Model Information and Number of Observations with Test Set Validation

The PLSMOD Procedure

<table>
<thead>
<tr>
<th>Model Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Source</td>
</tr>
<tr>
<td>Factor Extraction Method</td>
</tr>
<tr>
<td>PLS Algorithm</td>
</tr>
<tr>
<td>Validation Method</td>
</tr>
<tr>
<td>Validation Testing Criterion</td>
</tr>
<tr>
<td>Number of Random Permutations</td>
</tr>
<tr>
<td>Random Number Seed for Permutation</td>
</tr>
</tbody>
</table>

| 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 8.1.5 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 8.1.5 Test Set Validation for the Number of PLS Factors**

<table>
<thead>
<tr>
<th>Number of Extracted Factors</th>
<th>Root Mean PRESS</th>
<th>Prob &gt; PRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3.056797</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>1</td>
<td>2.630561</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>2</td>
<td>1.00706</td>
<td>0.0270</td>
</tr>
<tr>
<td>3</td>
<td>0.664603</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>4</td>
<td>0.521578</td>
<td>0.2820</td>
</tr>
<tr>
<td>5</td>
<td>0.500034</td>
<td>1.0000</td>
</tr>
<tr>
<td>6</td>
<td>0.513561</td>
<td>0.4840</td>
</tr>
<tr>
<td>7</td>
<td>0.501431</td>
<td>0.6470</td>
</tr>
<tr>
<td>8</td>
<td>1.055791</td>
<td>0.1860</td>
</tr>
<tr>
<td>9</td>
<td>1.435085</td>
<td>0.1860</td>
</tr>
<tr>
<td>10</td>
<td>1.720389</td>
<td>0.1510</td>
</tr>
</tbody>
</table>

Minimum Root Mean PRESS 0.500034
Minimizing Number of Factors 5
Smallest Number of Factors with p > 0.1 4

**Percentage Variation Accounted for by Partial Least Squares Factors**

<table>
<thead>
<tr>
<th>Number of Extracted Factors</th>
<th>Model Effects Current</th>
<th>Total</th>
<th>Response Variables Current</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>81.16545</td>
<td>81.16545</td>
<td>48.33854</td>
<td>48.33854</td>
</tr>
<tr>
<td>2</td>
<td>16.81131</td>
<td>97.97676</td>
<td>32.54654</td>
<td>80.88508</td>
</tr>
<tr>
<td>3</td>
<td>1.76391</td>
<td>99.74067</td>
<td>11.44380</td>
<td>92.32888</td>
</tr>
<tr>
<td>4</td>
<td>0.19507</td>
<td>99.93574</td>
<td>3.83631</td>
<td>96.16519</td>
</tr>
</tbody>
</table>

**References**


Chapter 9
The QTRSELECT Procedure

Contents
Overview: QTRSELECT Procedure ........................................ 328
PROC QTRSELECT Features ............................................. 328
PROC QTRSELECT Compared with Other SAS Procedures ........ 329
Using CAS Sessions and CAS Engine Librefs ..................... 330
Getting Started: QTRSELECT Procedure ......................... 331
Syntax: QTRSELECT Procedure ....................................... 339
PROC QTRSELECT Statement ........................................... 339
BY Statement ............................................................... 340
CLASS Statement ........................................................ 340
DISPLAY Statement ...................................................... 341
DISPLAYOUT Statement ............................................... 342
EFFECT Statement ....................................................... 342
MODEL Statement ........................................................ 344
OUTPUT Statement ....................................................... 346
PARTITION Statement ................................................... 348
SELECTION Statement .................................................. 349
WEIGHT Statement ...................................................... 349
Details: QTRSELECT Procedure ...................................... 350
Quantile Regression ....................................................... 350
Criteria Used in Model Selection .................................... 352
Diagnostic Statistics ..................................................... 355
Classification Variables and the SPLIT Option .................. 355
Using Validation and Test Data ...................................... 357
Computational Method .................................................. 360
Displayed Output ........................................................ 360
ODS Table Names ......................................................... 363
Examples: QTRSELECT Procedure ................................... 364
Example 9.1: Simulation Study ......................................... 364
Example 9.2: Growth Charts for Body Mass Index .............. 367
References ..................................................................... 369
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**
  - produces output data tables that contain predicted values, residuals, and confidence limits

The QTRSELECT procedure supports the following effect-selection methods. For a more detailed description of these methods, see the section “SELECTION Statement” on page 32 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 68 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 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. The QTRSELECT procedure is additionally capable of constructing complex effects, including univariate spline and polynomial expansions.

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 for parameter estimates and quantile predictions and a Wald test for significance level computation.

PROC QUANTSELECT provides LASSO and adaptive LASSO effect-selection methods and graphical summaries for effect selection, 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 and fit statistics 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, 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 mycas cas sessref=mysess;
```

The CAS statement creates the CAS session named mysess, and the LIBNAME statement creates the mycas 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 8 in Chapter 2, “Shared Concepts.”
Getting Started: QTRSELECT Procedure

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 9.1 shows the results.

```sas
proc contents varnum data=sashelp.baseball;
    ods select position;
run;
```

**Figure 9.1** Sashelp.Baseball Data Set

The CONTENTS Procedure

<table>
<thead>
<tr>
<th>Variables in Creation Order</th>
</tr>
</thead>
<tbody>
<tr>
<td># Variable</td>
</tr>
<tr>
<td>1 Name</td>
</tr>
<tr>
<td>2 Team</td>
</tr>
<tr>
<td>3 nAtBat</td>
</tr>
<tr>
<td>4 nHits</td>
</tr>
<tr>
<td>5 nHome</td>
</tr>
<tr>
<td>6 nRuns</td>
</tr>
<tr>
<td>7 nRBI</td>
</tr>
<tr>
<td>8 nBB</td>
</tr>
<tr>
<td>9 YrMajor</td>
</tr>
<tr>
<td>10 CrAtBat</td>
</tr>
<tr>
<td>11 CrHits</td>
</tr>
<tr>
<td>12 CrHome</td>
</tr>
<tr>
<td>13 CrRuns</td>
</tr>
<tr>
<td>14 CrRbi</td>
</tr>
<tr>
<td>15 CrBB</td>
</tr>
<tr>
<td>16 League</td>
</tr>
<tr>
<td>17 Division</td>
</tr>
<tr>
<td>18 Position</td>
</tr>
<tr>
<td>19 nOuts</td>
</tr>
<tr>
<td>20 nAssts</td>
</tr>
<tr>
<td>21 nError</td>
</tr>
<tr>
<td>22 Salary</td>
</tr>
<tr>
<td>23 Div</td>
</tr>
<tr>
<td>24 logSalary</td>
</tr>
</tbody>
</table>

**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 8 and “Loading a SAS Data Set onto a CAS Server” on page 9 in Chapter 2, “Shared Concepts.”
You can load the Sashelp.Baseball data set into your CAS session by using your CAS engine libref with the following DATA step:

```plaintext
data mycas.baseball;
    set sashelp.baseball;
run;
```

These statements assume that your CAS engine libref is named mycas, as in the section “Using CAS Sessions and CAS Engine Librefs” on page 330, 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:

```plaintext
proc qtrselect data=mycas.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 9.2** Number of Observations, Class Level Information, and Dimensions Tables

<table>
<thead>
<tr>
<th>The QTRSELECT Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Observations Read</td>
</tr>
<tr>
<td>Number of Observations Used</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class Level Information</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Class</strong></td>
</tr>
<tr>
<td>League</td>
</tr>
<tr>
<td>Division</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>Number of Effects</td>
</tr>
<tr>
<td>Number of Parameters</td>
</tr>
</tbody>
</table>

Figure 9.2 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 9.3** Fit Statistics

The QTRSELECT Procedure

Quantile Level 0.5

<table>
<thead>
<tr>
<th>Objective Function</th>
<th>25977</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>0.40584</td>
</tr>
<tr>
<td>Adj R1</td>
<td>0.36200</td>
</tr>
<tr>
<td>AIC</td>
<td>2453.81587</td>
</tr>
<tr>
<td>AICC</td>
<td>2456.94344</td>
</tr>
<tr>
<td>SBC</td>
<td>2521.68680</td>
</tr>
<tr>
<td>ACL</td>
<td>98.77118</td>
</tr>
</tbody>
</table>

**Figure 9.4** Parameter Estimates

<table>
<thead>
<tr>
<th>Parameter</th>
<th>DF</th>
<th>Estimate</th>
<th>Standard Error</th>
<th>t Value</th>
<th>Pr &gt;</th>
<th>t</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Intercept</td>
<td>1</td>
<td>-67.75322</td>
<td>39.95908</td>
<td>-1.70</td>
<td>0.0912</td>
<td></td>
<td></td>
</tr>
<tr>
<td>nAtBat</td>
<td>1</td>
<td>-1.57112</td>
<td>0.44700</td>
<td>-3.51</td>
<td>0.0005</td>
<td></td>
<td></td>
</tr>
<tr>
<td>nHits</td>
<td>1</td>
<td>8.82192</td>
<td>1.94990</td>
<td>4.52</td>
<td>&lt;0.0001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>nHome</td>
<td>1</td>
<td>-5.91757</td>
<td>4.91015</td>
<td>-1.21</td>
<td>0.2293</td>
<td></td>
<td></td>
</tr>
<tr>
<td>nRuns</td>
<td>1</td>
<td>-5.17076</td>
<td>2.14914</td>
<td>-2.41</td>
<td>0.0169</td>
<td></td>
<td></td>
</tr>
<tr>
<td>nRBI</td>
<td>1</td>
<td>0.77547</td>
<td>2.15469</td>
<td>0.36</td>
<td>0.7192</td>
<td></td>
<td></td>
</tr>
<tr>
<td>nBB</td>
<td>1</td>
<td>5.28866</td>
<td>1.67603</td>
<td>3.16</td>
<td>0.0018</td>
<td></td>
<td></td>
</tr>
<tr>
<td>YrMajor</td>
<td>1</td>
<td>6.61877</td>
<td>6.61798</td>
<td>1.00</td>
<td>0.3182</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CrAtBat</td>
<td>1</td>
<td>-0.04463</td>
<td>0.15485</td>
<td>-0.29</td>
<td>0.7734</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CrHits</td>
<td>1</td>
<td>0.07896</td>
<td>0.73594</td>
<td>0.11</td>
<td>0.9146</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CrHome</td>
<td>1</td>
<td>3.78231</td>
<td>1.90065</td>
<td>1.99</td>
<td>0.0477</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CrRuns</td>
<td>1</td>
<td>1.23105</td>
<td>0.77137</td>
<td>1.60</td>
<td>0.1118</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CrRbi</td>
<td>1</td>
<td>-0.70695</td>
<td>0.76888</td>
<td>-0.92</td>
<td>0.3588</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CrBB</td>
<td>1</td>
<td>-0.68911</td>
<td>0.41382</td>
<td>-1.67</td>
<td>0.0971</td>
<td></td>
<td></td>
</tr>
<tr>
<td>League American</td>
<td>1</td>
<td>-34.39136</td>
<td>24.37175</td>
<td>-1.41</td>
<td>0.1595</td>
<td></td>
<td></td>
</tr>
<tr>
<td>League National</td>
<td>0</td>
<td>0</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td></td>
</tr>
<tr>
<td>Division East</td>
<td>1</td>
<td>60.30856</td>
<td>27.28730</td>
<td>2.21</td>
<td>0.0280</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Division West</td>
<td>0</td>
<td>0</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td></td>
</tr>
<tr>
<td>nOuts</td>
<td>1</td>
<td>0.23273</td>
<td>0.12110</td>
<td>1.92</td>
<td>0.0558</td>
<td></td>
<td></td>
</tr>
<tr>
<td>nAssts</td>
<td>1</td>
<td>0.09824</td>
<td>0.18888</td>
<td>0.52</td>
<td>0.6035</td>
<td></td>
<td></td>
</tr>
<tr>
<td>nError</td>
<td>1</td>
<td>-0.81574</td>
<td>3.51436</td>
<td>-0.23</td>
<td>0.8166</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 9.3 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 350.

Figure 9.4 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 mycas.baseball data table:

```plaintext
proc qtrselect data=mycas.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 354.

**Figure 9.5 Selection Information**

**The QTRSELECT Procedure**

<table>
<thead>
<tr>
<th>Selection Information</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Selection Method</td>
<td>Forward</td>
</tr>
<tr>
<td>Select Criterion</td>
<td>Significance Level</td>
</tr>
<tr>
<td>Stop Criterion</td>
<td>Significance Level</td>
</tr>
<tr>
<td>Effect Hierarchy Enforced</td>
<td>None</td>
</tr>
<tr>
<td>Entry Significance Level (SLE)</td>
<td>0.1</td>
</tr>
<tr>
<td>Stop Horizon</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 9.5 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 9.6  Selection Summary

The QTRSELECT Procedure

Quantile Level 0.5
Selection Details

<table>
<thead>
<tr>
<th>Selection Summary</th>
<th>Effect Entered</th>
<th>Number Effects In</th>
<th>p Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 Intercept</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1 CrHome</td>
<td>2 &lt;.0001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 nHits</td>
<td>3 &lt;.0001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 CrHits</td>
<td>4 &lt;.0001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 nOuts</td>
<td>5 0.0185</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 nABat</td>
<td>6 0.0182</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 Division</td>
<td>7 0.0118</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7 nBB</td>
<td>8 0.0647</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8 nRuns</td>
<td>9 0.0558</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 9.6 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 9.7  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.

Selected Effects: Intercept nABat nHits nRuns nBB CrHits CrHome Division nOuts

Figure 9.7 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 9.8  Details of the Selected Model

The QTRSELECT Procedure

Quantile Level 0.5
Selected Model

<table>
<thead>
<tr>
<th>Objective Function</th>
<th>26568</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>0.39232</td>
</tr>
<tr>
<td>Adj R1</td>
<td>0.37318</td>
</tr>
<tr>
<td>AIC</td>
<td>2445.64547</td>
</tr>
<tr>
<td>AICC</td>
<td>2446.35693</td>
</tr>
<tr>
<td>SBC</td>
<td>2477.79485</td>
</tr>
<tr>
<td>ACL</td>
<td>101.01768</td>
</tr>
</tbody>
</table>
The “Fit Statistics” and “Parameter Estimates” tables in Figure 9.8 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 mycas.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:

```plaintext
proc qtrselect data=mycas.baseball alpha=0.1;
  class league division;
  model Salary = nAtBat nHits nHome nRuns nRBI nBB
eyrMajor crAtBat crHits crHome crRuns crRbi
crBB league division nOuts nAssts nError
   / quantile=0.1 0.9 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 of effect selection.

The “Fit Statistics” and “Parameter Estimates” tables in Figure 9.8 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 mycas.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:

```plaintext
proc qtrselect data=mycas.baseball alpha=0.1;
  class league division;
  model Salary = nAtBat nHits nHome nRuns nRBI nBB
eyrMajor crAtBat crHits crHome crRuns crRbi
crBB league division nOuts nAssts nError
   / quantile=0.1 0.9 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 of effect selection.
Figure 9.9 displays the “Selected Effects” and “Parameter Estimates” tables at quantile level 0.1.

**Figure 9.10** Parameter Estimates at Quantile Level 0.9

**Selected Effects:** Intercept nHits nBB CrAtBat crHits crHome crRbi League Division nOuts

| Parameter      | DF  | Estimate | Standard Error | 90% Confidence Limits | t Value | Pr > |t| |
|----------------|-----|----------|----------------|-----------------------|---------|------|---|
| Intercept      | 1   | 20.39804 | 58.17164       | -75.63745             | 0.35    | 0.7261 |
| nHits          | 1   | 2.30879  | 0.55640        | 1.39042               | 2.15    | 0.0329 |
| nBB            | 1   | 3.09799  | 1.44414        | 0.71386               | 4.15    | 0.0001 |
| CrAtBat        | 1   | -0.44914 | 0.14651        | -1.09602              | -0.0272 | 3.07  | 0.0024 |
| crHits         | 1   | 2.48064  | 0.51725        | 1.62672               | 4.80    | 0.0001 |
| crHome         | 1   | 6.29896  | 1.37134        | 4.03502               | 4.59    | 0.0001 |
| crRbi          | 1   | -2.12293 | 0.76546        | -3.86862              | -0.85923| 2.77  | 0.0060 |
| League American| 1   | -103.28955| 33.68480       | -158.89975            | -3.07   | 0.0024 |
| League National| 0   | 0        | .                | .                     | .       | .     |   |
| Division East  | 1   | 107.46694| 50.82797       | 23.55512              | 2.11    | 0.0355 |
| Division West  | 0   | 0        | .                | .                     | .       | .     |   |
| nOuts          | 1   | 0.39766  | 0.12820        | 0.18601               | 3.10    | 0.0021 |

Figure 9.9 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 mycas.baseball data table:

```plaintext
proc qtrselect data=mycas.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=s1 sls=0.1);
```
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 9.11** Top 10 Overpaid Baseball Players at Quantile Level 0.9

<table>
<thead>
<tr>
<th>Obs</th>
<th>Name</th>
<th>Salary</th>
<th>Overpaid</th>
<th>PredictedSalary</th>
<th>LCLM</th>
<th>UCLM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Smith, Ozzie</td>
<td>1940.0</td>
<td>1084.54</td>
<td>855.46</td>
<td>611.09</td>
<td>1099.83</td>
</tr>
<tr>
<td>2</td>
<td>Wiggins, Alan</td>
<td>700.0</td>
<td>220.74</td>
<td>479.26</td>
<td>345.93</td>
<td>612.60</td>
</tr>
<tr>
<td>3</td>
<td>Murray, Eddie</td>
<td>2460.0</td>
<td>213.10</td>
<td>2246.90</td>
<td>1982.06</td>
<td>2511.74</td>
</tr>
<tr>
<td>4</td>
<td>Strawberry, Darryl</td>
<td>1220.0</td>
<td>187.64</td>
<td>1032.36</td>
<td>914.99</td>
<td>1149.72</td>
</tr>
<tr>
<td>5</td>
<td>Gibson, Kirk</td>
<td>1300.0</td>
<td>177.24</td>
<td>1122.76</td>
<td>1011.97</td>
<td>1233.56</td>
</tr>
<tr>
<td>6</td>
<td>Trevino, Alex</td>
<td>512.5</td>
<td>149.70</td>
<td>362.80</td>
<td>273.17</td>
<td>452.43</td>
</tr>
<tr>
<td>7</td>
<td>Ramirez, Rafael</td>
<td>875.0</td>
<td>141.09</td>
<td>733.91</td>
<td>599.52</td>
<td>868.31</td>
</tr>
<tr>
<td>8</td>
<td>Romero, Ed</td>
<td>375.0</td>
<td>128.71</td>
<td>246.29</td>
<td>144.88</td>
<td>347.70</td>
</tr>
<tr>
<td>9</td>
<td>Mattingly, Don</td>
<td>1975.0</td>
<td>124.82</td>
<td>1850.18</td>
<td>1557.81</td>
<td>2142.55</td>
</tr>
<tr>
<td>10</td>
<td>Puhl, Terry</td>
<td>900.0</td>
<td>104.34</td>
<td>795.66</td>
<td>625.51</td>
<td>965.81</td>
</tr>
</tbody>
</table>

Output 9.11 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> ;
    DISPLAY <table-list> </options> ;
    DISPLAYOUT table-spec-list </options> ;
    EFFECT name=effect-type(variables <(options)}) ;
    MODEL dependent = <effects> </model-options> ;
    OUTPUT OUT=CAS-libref.data-table <keyword =name>> . . . <keyword =name>> </options> ;
    PARTITION partition-options ;
    SELECTION <METHOD=method <(method-options) >> <options> > ;
    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 9.1 summarizes the options in the PROC QTRSELECT statement by function.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic Options</td>
<td></td>
</tr>
<tr>
<td>DATA=</td>
<td>Specifies the input data table</td>
</tr>
<tr>
<td>Other Options</td>
<td></td>
</tr>
<tr>
<td>ALPHA=</td>
<td>Sets the significance level to be used for the construction of confidence intervals</td>
</tr>
<tr>
<td>COV=SPARSITY</td>
<td>Specifies the sparsity-function method for estimating the covariance matrix of the parameter estimates</td>
</tr>
<tr>
<td>NOCLPRINT</td>
<td>Limits or suppresses the display of class levels</td>
</tr>
</tbody>
</table>

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 STDPI 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 350.

DATA=CAS-libref.data-table
names the input data table for PROC QTRSELECT to use. The default is the most recently created
data table. CAS-libref.data-table is a two-level name, where

CAS-libref refers to a collection of information that is defined in the LIBNAME statement and
includes the caslib, 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 CAS-libref, see the section “Using CAS
Sessions and CAS Engine Librefs” on page 330.

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 SAS Language
Reference: Concepts.

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 9.2 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 10 in Chapter 2, “Shared Concepts.”

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DESCENDING</td>
<td>Reverses the sort order</td>
</tr>
<tr>
<td>MISSING</td>
<td>Treats missing values as valid levels</td>
</tr>
<tr>
<td>ORDER=</td>
<td>Specifies the sort order for the levels</td>
</tr>
<tr>
<td>PARAM=</td>
<td>Specifies the parameterization of the variable</td>
</tr>
<tr>
<td>REF=</td>
<td>Specifies the reference level of the variable</td>
</tr>
<tr>
<td>SPLIT</td>
<td>Allows design columns for a variable to enter or leave the model independently</td>
</tr>
</tbody>
</table>

**DISPLAY Statement**

```sas
DISPLAY <table-list> < / options> ;
```

The DISPLAY statement enables you to specify a list of ODS 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, all ODS tables are sent to the client and then the client creates a subset. If both DISPLAY and ODS statements are used together, the DISPLAY statement takes precedence over the ODS statements. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

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

- **CASESENSITIVE**
  
  performs a case-sensitive comparison of table names in the `table-list` to ODS 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 ODS tables except those specified in the `table-list`.

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

- **TRACE**
  
  displays the ODS table names, labels, and paths.

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

A path is a table name that is prefixed with dot-separated grouping information. For example, a SelectionSummary table that is produced by a procedure during a selection routine might have the path `Bygroup1.Summary.SelectionSummary`. A partial pathname does not include all groups; for example, `Selection-Summary` and `Summary_SelectionSummary` are partial pathnames for `Bygroup1.Summary.SelectionSummary`.  

DISP
When you specify a table name or partial pathname, all ODS 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 starts with a “/” or a “!”. 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.

**DISPLAYOUT Statement**

```
DISPLAYOUT table-spec-list / options;
```

The DISPLAYOUT statement enables you to create CAS 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 CAS output tables to create. Each entry in the list has either a `key` or a `key=value` format:

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

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 (/):

- **NOREPLACE**
  
  does not replace an existing CAS output table of the same name.

- **REPEATED**
  
  replicates the CAS 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 48 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 9.3 summarizes the options available in the EFFECT statement.

### Table 9.3 EFFECT Statement Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Collection Effects Options</strong></td>
<td></td>
</tr>
<tr>
<td>DETAILS</td>
<td>Displays the constituents of the collection effect</td>
</tr>
<tr>
<td><strong>Multimember Effects Options</strong></td>
<td></td>
</tr>
<tr>
<td>DETAILS</td>
<td>Displays the levels of the multimember effect</td>
</tr>
<tr>
<td>NOEFFECT</td>
<td>Specifies that observations whose levels are all missing for the multimember variables should have 0 values in the corresponding design matrix columns</td>
</tr>
<tr>
<td>STDIZE</td>
<td>Standardizes the design matrix entries so that each observation has a sum of 1</td>
</tr>
<tr>
<td>WEIGHT=</td>
<td>Specifies the weight variable for the contributions of each classification effect</td>
</tr>
<tr>
<td><strong>Polynomial Effects Options</strong></td>
<td></td>
</tr>
<tr>
<td>DEGREE=</td>
<td>Specifies the degree of the polynomial</td>
</tr>
<tr>
<td>DETAILS</td>
<td>Displays details of the specified polynomial</td>
</tr>
<tr>
<td>MDEGREE=</td>
<td>Specifies the maximum degree of any variable in a term of the polynomial</td>
</tr>
<tr>
<td>NOSEPARATE</td>
<td>Treats the polynomial as a single effect with multiple degrees of freedom</td>
</tr>
<tr>
<td>STANDARDIZE=</td>
<td>Specifies centering and scaling suboptions for the variables that define the polynomial</td>
</tr>
<tr>
<td><strong>Spline Effects Options</strong></td>
<td></td>
</tr>
<tr>
<td>BASIS=</td>
<td>Specifies the type of basis (B-spline basis or truncated power function basis) for the spline effect</td>
</tr>
<tr>
<td>DATABOUNDARY</td>
<td>Uses the extremes of the data as boundary knots for a B-spline basis</td>
</tr>
<tr>
<td>DEGREE=</td>
<td>Specifies the degree of the spline effect</td>
</tr>
<tr>
<td>DETAILS</td>
<td>Displays the knots and locations for each spline basis function</td>
</tr>
<tr>
<td>KNOTMAX=</td>
<td>Requests equally spaced right-side boundary knots starting at the variables’ maximum and ending at the KNOTMAX= value</td>
</tr>
<tr>
<td>KNOTMETHOD=</td>
<td>Specifies how to construct the knots for the spline effect</td>
</tr>
<tr>
<td>KNOTMIN=</td>
<td>Requests equally spaced left-side boundary knots starting at the variables’ minimum value</td>
</tr>
<tr>
<td>NATURALCUBIC</td>
<td>Specifies a natural cubic spline basis for the spline effect</td>
</tr>
</tbody>
</table>
Table 9.3  continued

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEPARATE</td>
<td>Treats the spline basis for each variable as a separate effect when multiple variables are specified</td>
</tr>
<tr>
<td>SPLIT</td>
<td>Treats each design matrix column as a separate effect for selection methods</td>
</tr>
</tbody>
</table>

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 17 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 45 in Chapter 2, “Shared Concepts.”

The model-options control other aspects of model formation and inference. Table 9.4 summarizes these options.

Table 9.4  MODEL Statement Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model Options</td>
<td>Requests confidence limits</td>
</tr>
<tr>
<td>CLB</td>
<td>Includes effects in all models for model selection</td>
</tr>
<tr>
<td>INCLUDE=</td>
<td>Models missing values by using extra indicator variables</td>
</tr>
<tr>
<td>INFORMATIVE</td>
<td>Suppresses the intercept</td>
</tr>
<tr>
<td>NOINT</td>
<td>Includes effects in the initial model for model selection</td>
</tr>
<tr>
<td>START=</td>
<td>Specifies quantile levels for the quantile regression</td>
</tr>
<tr>
<td>QUANTILES=</td>
<td></td>
</tr>
</tbody>
</table>

Model Options

You can specify the following options in the MODEL statement after a slash (/):
CLB
requests the 100(1 – α)% 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 65 in Chapter 2, “Shared Concepts.”

NOINT
suppresses the intercept term that is otherwise included in the model.

START=n
START=single-effect
START=(effects)
is used to begin 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.

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.

---

**OUTPUT Statement**

```
OUTPUT OUT=CAS-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:

```
OUTPUT OUT=CAS-libref.data-table
```

names the output data table for PROC QTRSELECT to use. You must specify this option before any other options. **CAS-libref.data-table** is a two-level name, where

- **CAS-libref** refers to a collection of information that is defined in the LIBNAME statement and includes the caslib, 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 **CAS-libref**, see the section “Using CAS Sessions and CAS Engine Librefs” on page 330.

- **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**\(\leq \text{name} \rangle\)
**PRED**\(\leq \text{name} \rangle\)
**P**\(\leq \text{name} \rangle\)
requests predicted values for the response variable. The default \text{name} is Pred.

**RESIDUAL**\(\leq \text{name} \rangle\)
**RESID**\(\leq \text{name} \rangle\)
**R**\(\leq \text{name} \rangle\)
requests the residual, calculated as ACTUAL – PREDICTED. The default \text{name} is Residual.

**ROLE**\(\leq \text{name} \rangle\)
requests a numeric variable that indicates the role played by each observation in fitting the model. The default \text{name} is _ROLE_. For each observation, the interpretation of this variable is shown in Table 9.5.

<table>
<thead>
<tr>
<th>Table 9.5 Role Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Value</strong></td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
</tbody>
</table>

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 QUANTILE= 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 mycas.out data table, which contains the two predicted quantile variables \text{p1} and \text{p2}:

```
proc qtrselect data=mycas.one;
  model y = x1-x4 /quantiles=0.5 0.3;
  output out=mycas.out pred=p;
run;
```

The variable \text{p1} is for quantile level 0.5, and the variable \text{p2} is for quantile level 0.3.

By using the SORT suboption in the QUANTILES option, the following statements generate the mycas.out data table in sorted order:

```
proc qtrselect data=mycas.one;
  model y = x1-x4 /quantiles(sort)=0.5 0.3;
  output out=mycas.out pred=p;
run;
```

The variable \text{p1} is for quantile level 0.3, and the variable \text{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 9.6 in the OUTPUT statement to obtain additional statistics. For computational formulas, see the section “Diagnostic Statistics” on page 355. 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 9.6** Keywords for OUTPUT Statement

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LCLM</td>
<td>Lower bound of a 100(1 − α)% confidence interval for the quantile prediction variable</td>
</tr>
<tr>
<td>STDP</td>
<td>Standard error of the quantile prediction variable</td>
</tr>
<tr>
<td>UCLM</td>
<td>Upper bound of a 100(1 − α)% confidence interval for the quantile prediction variable</td>
</tr>
</tbody>
</table>

**PARTITION Statement**

```plaintext
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 66 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 of the observations 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 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 32 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.

**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, all observations that are used in the analysis are assigned a weight of 1.
Details: QTRSELECT Procedure

Quantile Regression

This section describes the basic concepts and notations for quantile regression and quantile regression model selection.

Let \( \{(y_i, x_i) : i = 1, \ldots, n\} \) denote a data set of observations, where \( y_i \) are responses and \( x_i \) are regressors. Koenker and Bassett (1978) define the regression quantile at quantile level \( \tau \in (0, 1) \) as any solution to the minimization problem

\[
\min_{\beta \in \mathbb{R}^p} \sum_{i=1}^{n} \rho_\tau(y_i - x_i' \beta)
\]

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

\[
\min_{\beta \in \mathbb{R}^p} \sum_{i=1}^{n} w_i \rho_\tau(w_i (y_i - x_i' \beta))
\]

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 \( x_i \) follows the linear model,

\[
Y_i = x_i' \beta + \epsilon_i
\]

where \( \epsilon_i \) 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 352.

Linear-in-Parameter Model with Non-iid Settings

The more general form of a linear quantile regression model is

\[
Q_Y(\tau | x) = x' \beta(\tau)
\]

where the iid assumption is not necessary.
Let $s_i(\tau) = 1/f_i(F_i^{-1}(\tau))$ denote the sparsity function of the $i$th observation. Under some regularity conditions, the asymptotic distribution of the general form of quantile regression estimates is

$$\sqrt{n}(\hat{\beta}(\tau) - \beta(\tau)) \to N(0, \tau(1 - \tau)H^{-1} \Omega H^{-1})$$

where $H = \lim_{n \to \infty} n^{-1} \sum x_i x_i'$ and $\Omega = \lim_{n \to \infty} n^{-1} \sum x_i x_i'$.

The sparsity function of the $i$th 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 $\hat{F}_i^{-1}(\tau \pm h_n) = x_i' \hat{\beta}(\tau \pm h_n)$ are the quantile predictions of the $i$th observation at quantile levels $(\tau \pm h_n)$.

Accordingly, the covariance matrix of $\hat{\beta}(\tau)$ can be estimated as

$$\hat{\text{COV}}\left(\hat{\beta}(\tau)\right) = n^{-1}(1 - \tau)\hat{H}_n^{-1} \hat{\Omega}_n \hat{H}_n^{-1}$$

where $\hat{H}_n = n^{-1} \sum (x_i x_i' / \hat{s}_i(\tau))$ and $\hat{\Omega}_n = n^{-1} \sum x_i x_i'$.

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:
  $$h_n = n^{-1/5} (4.5 v^2(\tau))^{1/5}$$

- The Hall-Sheather bandwidth is based on Edgeworth expansions for studentized quantiles:
  $$h_n = n^{-1/3} z_\alpha^{2/3} (1.5 v(\tau))^{1/3}$$

  $z_\alpha$ 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 $j$th parameter estimate, and let $\hat{\beta}(\tau) = (\hat{\beta}_1(\tau), \ldots, \hat{\beta}_p(\tau))$ 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 9.7 summarizes these statistics for $\hat{\beta}_j(\tau)$.

### Table 9.7  More Statistics for $\hat{\beta}_j(\tau)$

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard error: $\hat{\sigma}_j$</td>
<td>$\sqrt{Cov(\hat{\beta}(\tau))_{jj}}$</td>
</tr>
<tr>
<td>$(1 - \alpha)%$ confidence limits</td>
<td>$\hat{\beta}<em>j(\tau) \pm t</em>{1,1-\frac{\alpha}{2}}\hat{\sigma}_j$</td>
</tr>
<tr>
<td>$t$ value</td>
<td>$\hat{\beta}_j(\tau)/\hat{\sigma}_j$</td>
</tr>
<tr>
<td>$Pr &gt;</td>
<td>t</td>
</tr>
</tbody>
</table>

Here $Cov(\hat{\beta}(\tau))_{jj}$ is the $(j, j)$ element of $Cov(\hat{\beta}(\tau))$, and $t_{1,1-\frac{\alpha}{2}}$ denotes the $(1 - \frac{\alpha}{2})$-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 9.8 provides formulas and definitions for these fit statistics.

### Table 9.8  Formulas and Definitions for Model Fit Summary Statistics for Single Quantile Effect Selection

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Definition or Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td>Number of observations</td>
</tr>
</tbody>
</table>
Table 9.8  continued

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Definition or Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p )</td>
<td>Number of parameters, including the intercept</td>
</tr>
<tr>
<td>( r_i(\tau) )</td>
<td>Residual for the ( i )th observation; ( r_i(\tau) = y_i - x_i \hat{\beta}(\tau) )</td>
</tr>
<tr>
<td>( D(\tau) )</td>
<td>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.</td>
</tr>
<tr>
<td>( D_0(\tau) )</td>
<td>Total sum of check losses for intercept-only model if the intercept is a forced-in effect; otherwise for empty model.</td>
</tr>
<tr>
<td>( ACL(\tau) )</td>
<td>Average check loss; ( ACL(\tau) = \frac{D(\tau)}{n} )</td>
</tr>
<tr>
<td>( R1(\tau) )</td>
<td>Counterpart of linear regression R square for quantile regression; ( R1(\tau) = 1 - \frac{D(\tau)}{D_0(\tau)} )</td>
</tr>
<tr>
<td>( ADJR1(\tau) )</td>
<td>Adjusted ( R1; ; 1 - \frac{(n - 1)D(\tau)}{(n - p)D_0(\tau)} ) if intercept is a forced-in effect; otherwise ( 1 - \frac{(n - p)D_0(\tau)}{nD(\tau)} ).</td>
</tr>
<tr>
<td>( AIC(\tau) )</td>
<td>( 2n \ln (ACL(\tau)) + 2p )</td>
</tr>
<tr>
<td>( AICC(\tau) )</td>
<td>( 2n \ln (ACL(\tau)) + \frac{2pn}{n - p - 1} )</td>
</tr>
<tr>
<td>( SBC(\tau) )</td>
<td>( 2n \ln (ACL(\tau)) + p \ln (n) )</td>
</tr>
</tbody>
</table>

The ADJR1(\( \tau \)) criterion is equivalent to the generalized approximate cross validation (GACV) criterion for quantile regression (Yuan 2006). The GACV criterion is defined as

\[
GACV(\tau) = D(\tau)/(n - p)
\]

which is proportional to \( 1 - ADJR1(\tau) \).

**Quasi-likelihood Information Criteria**

Given the quantile level \( \tau \), assume that the distribution of \( Y_i \) conditional on \( x_i \) follows the linear model

\[
Y_i = x'_i \beta + \epsilon_i
\]

where \( \epsilon_i \) 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 \( \sigma \) is the scale parameter. Then, the negative log-likelihood function is

\[
l_\tau(\beta, \sigma) = n \log(\sigma) + \sigma^{-1} \sum_{i=1}^{n} \rho_\tau(y_i - x'_i \beta) - n \log(\tau(1 - \tau))
\]

Under these settings, the maximum likelihood estimate (MLE) of \( \beta \) is the same as the relevant level-\( \tau \) quantile regression solution \( \hat{\beta}(\tau) \), and the MLE for \( \sigma \) is

\[
\hat{\sigma}(\tau) = n^{-1} \sum_{i=1}^{n} \rho_\tau(y_i - x'_i \hat{\beta}(\tau))
\]
where \( \hat{\sigma}(\tau) \) equals the level-\( \tau \) average check loss \( \text{ACL}(\tau) \) for the quantile regression solution.

Because the general form of Akaike’s information criterion (AIC) is \( \text{AIC} = (-2l + 2p) \), the quasi-likelihood AIC for quantile regression is

\[
\text{AIC}(\tau) = 2n \ln (\text{ACL}(\tau)) + 2p
\]

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:

\[
\text{AICC}(\tau) = 2n \ln (\text{ACL}(\tau)) + \frac{2pn}{n - p - 1}
\]

\[
\text{SBC}(\tau) = 2n \ln (\text{ACL}(\tau)) + p \ln(n)
\]

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 9.1: Simulation Study” on page 364. 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:

\[
Q_Y(\tau | x_1, x_2) = x_1^T \beta_1(\tau) + x_2^T \beta_2(\tau)
\]

At each step of an effect-selection process, a candidate effect can be represented as \( x_2 \), and the significance level of the candidate effect can be calculated by testing the null hypothesis: \( H_0 : \beta_2(\tau) = 0 \).

The QTRSELECT procedure supports the Wald test for computing candidate significance levels. Let \( \hat{\beta}(\tau) = \left( \hat{\beta}_1(\tau), \hat{\beta}_2(\tau) \right)' \) be the parameter estimates for the extended model, and denote the estimated covariance matrix of \( \hat{\beta}(\tau) \) 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 \( \hat{\beta}_2(\tau) \). Then the Wald test score is defined as

\[
\hat{\beta}_2^T(\tau) \hat{\Sigma}_{22}^{-1}(\tau) \hat{\beta}_2(\tau)
\]

Under the null hypothesis that the reduced model is the true model, the Wald score follows a \( \chi^2 \) 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, \( \delta_i(\tau) \)'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 SENTRY 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 $Q_Y(\tau|x) = x'\hat{\beta}(\tau)$, and the parameter estimate $\hat{\beta}(\tau)$ is the solution that minimizes $\sum_{i=1}^{n} \rho_\tau(y_i - x'_i\hat{\beta})$. The subscript $i$ is for the $i$th observation. The subscript $j$ is for the $j$th-smallest quantile level among all the specified QUANTILE= levels in the MODEL statement. $\hat{\Sigma}(\tau)$ denotes the covariance estimation for $\hat{\beta}(\tau)$.

The ALPHA= option in the PROC QTRSELECT statement sets the $\alpha$ value for the confidence limit statistics. The degrees of freedom for $t_\alpha$ are $n - p$.

Table 9.9 contains the diagnostic statistics and their formulas. Each statistic is computed for each observation.

<table>
<thead>
<tr>
<th>MODEL Option or Statistic</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREDj</td>
<td>$\hat{y}_{ji} = x'_i\hat{\beta}(\tau_j)$</td>
</tr>
<tr>
<td>RESj</td>
<td>$y_i - \hat{y}_{ji}$</td>
</tr>
<tr>
<td>STDPj</td>
<td>$\sqrt{x'_i\hat{\Sigma}(\tau_j)x_i}$</td>
</tr>
<tr>
<td>LCLMj</td>
<td>$\hat{y}<em>{ji} - t</em>\alpha STDPj_i$</td>
</tr>
<tr>
<td>UCLMj</td>
<td>$\hat{y}<em>{ji} + t</em>\alpha STDPj_i$</td>
</tr>
</tbody>
</table>

### 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:

```plaintext
data mycas.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=mycas.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 9.12 is produced by default whenever you specify a CLASS statement.

**Figure 9.12** Class Levels
The QTRSELECT Procedure

<table>
<thead>
<tr>
<th>Class Level Information</th>
<th>Class Levels</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>c1</td>
<td>6 1 2 3 4 5 6</td>
<td></td>
</tr>
<tr>
<td>c2</td>
<td>3 high medium low</td>
<td></td>
</tr>
</tbody>
</table>

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 9.13 shows that for this example the parameters that correspond only to levels 3 and 5 of c1 are in the selected model.

**Figure 9.13** Parameter Estimates

<table>
<thead>
<tr>
<th>Parameter</th>
<th>DF</th>
<th>Estimate</th>
<th>Standard Error</th>
<th>t Value</th>
<th>Pr &gt;</th>
<th>t</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Intercept</td>
<td>1</td>
<td>2.84133</td>
<td>0.11898</td>
<td>23.88</td>
<td>&lt;.0001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c1_3</td>
<td>1</td>
<td>10.15300</td>
<td>0.10121</td>
<td>100.31</td>
<td>&lt;.0001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c1_5</td>
<td>1</td>
<td>5.02683</td>
<td>0.09847</td>
<td>51.05</td>
<td>&lt;.0001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c2 high</td>
<td>1</td>
<td>-3.10623</td>
<td>0.09550</td>
<td>-32.52</td>
<td>&lt;.0001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c2 medium</td>
<td>1</td>
<td>-2.88624</td>
<td>0.11896</td>
<td>-24.26</td>
<td>&lt;.0001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c2 low</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x1</td>
<td>1</td>
<td>1.23941</td>
<td>0.14059</td>
<td>8.82</td>
<td>&lt;.0001</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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 mycas.inData data table, reserving 50% for training and 25% each for validation and testing:

```plaintext
proc qtrselect data=mycas.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 mycas.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.

```plaintext
proc qtrselect data=mycas.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:

```plaintext
%let seed=321;
%let n=600;
%let p=10;

data mycas.roleExample;
  array x{&p} x1-x&p;
```
length r $8;
drop i j k;

do i=1 to &n;
   do j=1 to &p;
      \textit{x}\{j\} = \textit{ranuni}(&seed);
   end;

y = x1 + x2 + x3 + \textit{ranuni}(&seed);

\textit{k} = \textit{mod}(i,3);
   \textit{if} \textit{k}=0 \textit{then} \textit{r} = 'train';
   \textit{else if} \textit{k}=1 \textit{then} \textit{r} = 'validate';
   \textit{else if} \textit{k}=2 \textit{then} \textit{r} = 'test';
   output;
end;
run;

\textit{proc qtrselect} data=mycas.roleExample;
   \textit{model y} = x1-x&p;
   \textit{selection method}=forward(select=validate stop=sbc);
   \textit{partition rolevar=r(train='train' validate='validate' test='test')};
run;

The “Number of Observations” table shown in \textit{Figure 9.14} displays the number of observations used for training, validation, and testing.

\textbf{Figure 9.14}  Number of Observations

\textbf{The QTRSELECT Procedure}

\begin{tabular}{|l|c|}
\hline
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 \\
\hline
\end{tabular}

The “Selection Summary” table shown in \textit{Figure 9.15} displays the validation ACL values for each step of the selection process.
The “Fit Statistics” table shown in Figure 9.16 displays the training ACL, the validation ACL, and the testing ACL for the final model.

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 68 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:

<table>
<thead>
<tr>
<th>Code</th>
<th>Stop Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>All eligible effects are in the model.</td>
</tr>
<tr>
<td>2</td>
<td>All eligible effects have been removed.</td>
</tr>
<tr>
<td>3</td>
<td>Specified maximum number of steps done.</td>
</tr>
<tr>
<td>4</td>
<td>The model contains the specified maximum number of effects.</td>
</tr>
<tr>
<td>5</td>
<td>The model contains the specified minimum number of effects (for backward selection).</td>
</tr>
<tr>
<td>6</td>
<td>The stopping criterion is at a local optimum.</td>
</tr>
<tr>
<td>7</td>
<td>No suitable add or drop candidate could be found.</td>
</tr>
<tr>
<td>8</td>
<td>Adding or dropping any effect does not improve the selection criterion.</td>
</tr>
<tr>
<td>9</td>
<td>No candidate meets the appropriate SLE or SLS significance level.</td>
</tr>
<tr>
<td>10</td>
<td>Stepwise selection is cycling.</td>
</tr>
<tr>
<td>11</td>
<td>The model is an exact fit.</td>
</tr>
<tr>
<td>12</td>
<td>Dropping an effect would result in an empty model.</td>
</tr>
</tbody>
</table>
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 9.8.

- $R_1$, 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 $R_1$, the adjusted $R_1$, a version of $R_1$ that has been adjusted for degrees of freedom. It is calculated as

$$\text{Adj } R_1 = 1 - \frac{n - i}{n - p} (1 - R_1)$$

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 9.8.

- 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.

**OutCASTblFull**

When you specify an OUTPUT statement or a DISPLAYOUT statement to create output tables on your CAS server, the “Output CAS Tables” table displays the names, the numbers of rows and columns, and the CAS libraries in which the table is created for every output table that you requested.

**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 9.10.

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Required Statement / Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Candidates</td>
<td>Swap candidates at step</td>
<td>SELECTION DETAILS=ALL</td>
</tr>
<tr>
<td>ClassInfo</td>
<td>Level information from the CLASS statement</td>
<td>CLASS</td>
</tr>
<tr>
<td>Dimensions</td>
<td>Model dimensions</td>
<td>Default output</td>
</tr>
<tr>
<td>EntryCandidates</td>
<td>Candidates for entry at step</td>
<td>SELECTION DETAILS=ALL</td>
</tr>
<tr>
<td>FitStatistics</td>
<td>Fit statistics</td>
<td>Default output</td>
</tr>
<tr>
<td>NObs</td>
<td>Number of observations read and used</td>
<td>Default output</td>
</tr>
</tbody>
</table>
Table 9.10 continued

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Required Statement / Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>OutCASTblFull</td>
<td>Library and name of the output data table, and number of rows and columns in the table</td>
<td>OUTPUT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DISPLAYOUT</td>
</tr>
<tr>
<td>ParameterEstimates</td>
<td>Solutions for the parameter estimates associated with effects in the MODEL statement</td>
<td>Default output</td>
</tr>
<tr>
<td>RemovalCandidates</td>
<td>Candidates for removal at step</td>
<td>SELECTION DETAILS=ALL</td>
</tr>
<tr>
<td>SelectedEffects</td>
<td>List of selected effects</td>
<td>SELECTION</td>
</tr>
<tr>
<td>SelectionInfo</td>
<td>Information about selection settings</td>
<td>Default output</td>
</tr>
<tr>
<td>SelectionReason</td>
<td>Reason for selecting the final model</td>
<td>SELECTION</td>
</tr>
<tr>
<td>SelectionSummary</td>
<td>Summary information about the model selection steps</td>
<td>SELECTION</td>
</tr>
<tr>
<td>StopReason</td>
<td>Reason selection was terminated</td>
<td>SELECTION</td>
</tr>
<tr>
<td>Timing</td>
<td>Timing breakdown by task</td>
<td>SELECTION DETAILS</td>
</tr>
</tbody>
</table>

Examples: QTRSELECT Procedure

Example 9.1: Simulation Study

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):

```sas
%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);
```
Example 9.1: Simulation Study

Variable $U$ in the data set indicates the true quantile level of the response $y$ conditional on $x = (x_1, \ldots, x_p)$. Let $Q_Y(\tau|x) = x\beta(\tau)$ denote the underlying quantile regression model, where $\beta(\tau) = (\beta_1(\tau), \ldots, \beta_p(\tau))'$.

Then, the true parameter functions are

$$
\begin{align*}
\beta_1(\tau) &= \tau - 0.1 \\
\beta_2(\tau) &= \tau^2 - 0.25 \\
\beta_3(\tau) &= \exp(\tau) - \exp(0.9) \\
\beta_4(\tau) &= \cdots = \beta_p(\tau) = 0
\end{align*}
$$

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 $x_2$ and $x_3$ and drop all the other effects in this data set at $\tau = 0.1$. By the same rationale, $x_1$ and $x_3$ should be selected at $\tau = 0.5$ with $\beta_1(0.5) = 0.4$ and $\beta_3(0.5) \approx -0.810882$, and $x_1$ and $x_2$ should be selected at $\tau = 0.9$ with $\beta_1(0.9) = 0.8$ and $\beta_2(0.9) = 0.56$.

The following statements load the analysisData data set into your CAS session by using the mycas engine libref:

```plaintext
data mycas.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.

```plaintext
proc qtrselect data=mycas.analysisData;
   model y= x1-x&p / quantile=0.1 0.5 0.9 clb;
   selection method=forward;
   output out=mycas.out p=pred;
run;
```

Output 9.1.1 shows that, by default, the CHOOSE= and STOP= options are both set to SBC.
Output 9.1.2, Output 9.1.3, and Output 9.1.4 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 9.1.2** Parameter Estimates at \( \tau = 0.1 \)

| 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 9.1.3** Parameter Estimates at \( \tau = 0.5 \)

| 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 9.1.4** Parameter Estimates at \( \tau = 0.9 \)

| 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 9.2: Growth Charts for Body Mass Index

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 9.2.1 shows the results.

```sas
proc contents varnum data=sashelp.BMIMen;
ods select position;
run;
```

**Output 9.2.1 Sashelp.BMIMen Data Set**

The CONTENTS Procedure

<table>
<thead>
<tr>
<th>Variables in Creation</th>
<th>Order</th>
<th># Variable</th>
<th>Type</th>
<th>Len</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age</td>
<td>1</td>
<td>Num</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>BMI</td>
<td>2</td>
<td>Num</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>
```

The following statements load the Sashelp.BMIMen data set into your CAS session by using the mycas engine libref:

```sas
data mycas.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 mycas.BMIMen data table at 10 quantile levels:

```sas
%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=mycas.BMIMen;
   model logBMI = InveAge SqrtAge Age SqrtAge*Age Age*Age*Age / quantile=&quantile;
   output out=mycas.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 `mycas.BMIOut` data table:

```sas
%let BMIcolor=red olive orange blue brown gray violet black gold green;

%macro plotBMI;
  data BMIPred;
    set mycas.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 9.2.2 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.
References


Chapter 10
The REGSELECT Procedure

Contents

Overview: REGSELECT Procedure .................................................. 372
PROC REGSELECT Features ......................................................... 372
PROC REGSELECT Compared with Other SAS Procedures ................. 373
Using CAS Sessions and CAS Engine Librefs ................................. 374
Getting Started: REGSELECT Procedure .................................... 375
Syntax: REGSELECT Procedure ...................................................... 381
  PROC REGSELECT Statement .................................................. 381
  BY Statement .................................................................. 382
  CLASS Statement ............................................................. 382
  CODE Statement ............................................................... 383
  DISPLAY Statement ............................................................ 383
  DISPLAYOUT Statement ...................................................... 384
  EFFECT Statement ............................................................. 385
  FREQ Statement ................................................................. 386
  MODEL Statement ............................................................... 387
  OUTPUT Statement ............................................................. 388
  PARTITION Statement ......................................................... 391
  SELECTION Statement ......................................................... 391
  WEIGHT Statement ............................................................. 392
Details: REGSELECT Procedure .................................................. 393
  Criteria Used in Model Selection .......................................... 393
  Diagnostic Statistics ............................................................ 395
  Classification Variables and the SPLIT Option ........................ 396
  Using Validation and Test Data ............................................. 397
  Computational Method ........................................................ 398
  Joint Tests and Type 3 Tests ................................................. 398
  Displayed Output .................................................................. 399
  ODS Table Names ................................................................ 403
Examples: REGSELECT Procedure ................................................. 404
  Example 10.1: Model Selection with Validation ....................... 404
References ................................................................................. 411
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
  - supports stopping and selection rules based on external validation and leave-one-out cross validation

- **Display and output**
  - produces output data tables that contain predicted values, residuals, studentized residuals, confidence limits, and influence statistics

The REGSELECT procedure supports the following effect-selection methods. For a more detailed description of these methods, see the section “SELECTION Statement” on page 32 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 $R^2$, this method is the same as the MAXR method in the REG procedure in SAS/STAT software.

• Least angle regression, 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.

Hybrid versions of the LAR and LASSO methods are also supported. They use 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 68 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 and HPREG procedures offer the same methods of effect selection, 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 and LASSO methods), the ability to use external validation data and cross validation as selection criteria, and extensive options to customize the selection process. 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.

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 and LASSO methods, the ability to use external validation data and cross validation as selection criteria, and extensive options to customize the selection process. PROC REGSELECT does not support the all-subset-based methods that you find in PROC REG, nor does it support the MINR method.

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:

```sas
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:

```sas
cas mysess;
libname mycas cas sessref=mysess;
```

The CAS statement creates the CAS session named `mysess`, and the LIBNAME statement creates the `mycas` 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:

```sas
cas mysess terminate;
```

For more information about the CAS and LIBNAME statements, see the section “Introduction to Shared Concepts” on page 8 in Chapter 2, “Shared Concepts.”

---

**Getting Started: REGSELECT Procedure**

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 (*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 10.1) the variables in the data set:

```sas
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.

**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 8 and “Loading a SAS Data Set onto a CAS Server” on page 9 in Chapter 2, “Shared Concepts.”

You can load the Sashelp.Baseball data set into your CAS session by using your CAS engine libref with the following DATA step:

```sas
data mycas.baseball;
  set sashelp.baseball;
run;
```

These statements assume that your CAS engine libref is named Mycas, as in the section “Using CAS Sessions and CAS Engine Librefs” on page 374, but you can substitute any appropriately defined CAS engine libref.
The following statements select a model by using the default settings for stepwise selection:

```plaintext
proc regselect data=mycas.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;
run;
```

The default output from this analysis is presented in Figure 10.2 through Figure 10.6.

**Figure 10.2** Selection Information, Number of Observations, Class Level Information, and Dimensions

<table>
<thead>
<tr>
<th>The REGSELECT Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selection Information</td>
</tr>
<tr>
<td>Selection Method</td>
</tr>
<tr>
<td>Select Criterion</td>
</tr>
<tr>
<td>Stop Criterion</td>
</tr>
<tr>
<td>Effect Hierarchy Enforced</td>
</tr>
<tr>
<td>Stop Horizon</td>
</tr>
</tbody>
</table>

**Number of Observations Read:** 322  
**Number of Observations Used:** 263

<table>
<thead>
<tr>
<th>Class Level Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
</tr>
<tr>
<td>League</td>
</tr>
<tr>
<td>Division</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Effects</td>
</tr>
<tr>
<td>Number of Parameters</td>
</tr>
</tbody>
</table>

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 10.2 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 10.3 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 10.3 Selection Summary Table**

<table>
<thead>
<tr>
<th>Selection Summary</th>
<th>Effect Entered</th>
<th>Effect Removed</th>
<th>Number Effects In</th>
<th>SBC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 0</td>
<td>Intercept</td>
<td></td>
<td>1</td>
<td>-57.2041</td>
</tr>
<tr>
<td>1</td>
<td>CrRuns</td>
<td>2</td>
<td>-194.3166</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>nHits</td>
<td>3</td>
<td>-252.5794</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>YrMajor</td>
<td>4</td>
<td>-262.7322</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>CrRuns</td>
<td>3</td>
<td>-262.8353</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>nBB</td>
<td>4</td>
<td>-269.7804*</td>
<td></td>
</tr>
</tbody>
</table>

* Optimal Value Of Criterion

Figure 10.4 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 10.4 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 “Analysis of Variance,” “Fit Statistics,” and “Parameter Estimates” tables shown in Figure 10.5 display details of the selected model.
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.

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:
The VIF and CLB options in the **MODEL** statement request variance inflation factors and 95% confidence limits, respectively, for the parameter estimates. Figure 10.7 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 10.7** Parameter Estimates with Additional Statistics

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:

```sas
proc print data=mycas.baseballOut(obs=5);
run;
```
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=CAS-libref.data-table <keyword =<name>> . . . <keyword =<name>> < options >;
   PARTITION partition-options;
   SELECTION < METHOD=method <(method-options) > >< options >;
   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 10.1 summarizes the options in the PROC REGSELECT statement by function.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic Options</td>
<td></td>
</tr>
<tr>
<td>DATA=</td>
<td>Specifies the input data set</td>
</tr>
</tbody>
</table>
Table 10.1  continued

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Other Options</strong></td>
<td></td>
</tr>
<tr>
<td>NOCLPRINT</td>
<td>Limits or suppresses the display of class levels</td>
</tr>
<tr>
<td>ALPHA=</td>
<td>Sets the significance level to be used for the construction of confidence intervals</td>
</tr>
</tbody>
</table>

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=**CAS-libref.data-table
names the input data table for PROC REGSELECT to use. The default is the most recently created data table. **CAS-libref** specifies the CAS library where the input data table resides, and **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 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 SAS Language Reference: Concepts.

**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 10.2 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 10 in Chapter 2, “Shared Concepts.”
Table 10.2  CLASS Statement Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DESCENDING</td>
<td>Reverses the sort order</td>
</tr>
<tr>
<td>MISSING</td>
<td>Treats missing values as valid levels</td>
</tr>
<tr>
<td>ORDER=</td>
<td>Specifies the sort order for the levels</td>
</tr>
<tr>
<td>PARAM=</td>
<td>Specifies the parameterization of the variable</td>
</tr>
<tr>
<td>REF=</td>
<td>Specifies the reference level of the variable</td>
</tr>
<tr>
<td>SPLIT</td>
<td>Allows design columns for a variable to enter or leave the model independently</td>
</tr>
</tbody>
</table>

**CODE Statement**

```
CODE < options > ;
```

The CODE statement writes SAS DATA step code for computing predicted values of the fitted model either to a file or to a catalog entry. This code can then be included in a DATA step to score new data.

Table 10.3 summarizes the options available in the CODE statement.

Table 10.3  CODE Statement Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMMENT</td>
<td>Adds comments to the generated code</td>
</tr>
<tr>
<td>FILE=</td>
<td>Names the file where the generated code is saved</td>
</tr>
<tr>
<td>FORMATWIDTH=</td>
<td>Specifies the numeric format width for the regression coefficients</td>
</tr>
<tr>
<td>INDENTSIZE=</td>
<td>Specifies the number of spaces to indent the generated code</td>
</tr>
<tr>
<td>LABELID=</td>
<td>Specifies a number used to construct names and labels</td>
</tr>
<tr>
<td>LINESIZE=</td>
<td>Specifies the line size for the generated code</td>
</tr>
<tr>
<td>NOTRIM</td>
<td>Compares formatted values, including blank padding</td>
</tr>
<tr>
<td>PCATALL</td>
<td>Generates probabilities for all levels of categorical response variables</td>
</tr>
</tbody>
</table>

For more information about the syntax of the CODE statement, see the section “CODE Statement” on page 14 in Chapter 2, “Shared Concepts.”

**DISPLAY Statement**

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

The DISPLAY statement enables you to specify a list of ODS 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, all ODS tables are sent to the client and then the client creates a subset. If both DISPLAY and ODS statements are used
together, the DISPLAY statement takes precedence over the ODS statements. For more information about ODS, see SAS Output Delivery System: Procedures Guide.

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

**CASESENSITIVE**
performs a case-sensitive comparison of table names in the table-list to ODS 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 ODS tables except those specified in the table-list.

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

**TRACE**
displays the ODS table names, labels, and paths.

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

A path is a table name that is prefixed with dot-separated grouping information. For example, a SelectionSummary table that is produced by a procedure during a selection routine might have the path Bygroup1.Summary.SelectionSummary. A partial pathname does not include all groups; for example, Selection- Summary and Summary.SelectionSummary are partial pathnames for Bygroup1.Summary.SelectionSummary.

When you specify a table name or partial pathname, all ODS 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 starts with a “/” or a “!”. 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.

---

**DISPLAYOUT Statement**

DISPLAYOUT table-spec-list < / options > ;

The DISPLAYOUT statement enables you to create CAS 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 CAS output tables to create. Each entry in the list has either a key or a key=value format:

- **key=value** specifies key as the ODS table name, path, or partial pathname, and specifies value as the CAS output table name.
- **key** specifies key as the ODS table name and also as the CAS output table name.
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 (/):

**NOREPLACE**

does not replace an existing CAS output table of the same name.

**REPEATED**

replicates the CAS output tables on all nodes.

---

**EFFECT Statement**

\[
\text{EFFECT name=}\text{effect-type (variables } < \text{/ options}> \text{)} ;
\]

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 48 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 10.4 summarizes the **options** available in the EFFECT statement.

<table>
<thead>
<tr>
<th><strong>Table 10.4</strong> EFFECT Statement Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Collection Effects Options</strong></td>
<td></td>
</tr>
<tr>
<td>DETAILS</td>
<td>Displays the constituents of the collection effect</td>
</tr>
<tr>
<td><strong>Multimember Effects Options</strong></td>
<td></td>
</tr>
<tr>
<td>DETAILS</td>
<td>Displays the levels of the multimember effect</td>
</tr>
<tr>
<td>NOEFFECT</td>
<td>Specifies that observations whose levels are all missing for the multimember variables should have 0 values in the corresponding design matrix columns</td>
</tr>
<tr>
<td>STDIZE</td>
<td>Standardizes the design matrix entries so that each observation has a sum of 1</td>
</tr>
</tbody>
</table>
### Table 10.4 continued

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WEIGHT=</td>
<td>Specifies the weight variable for the contributions of each classification effect</td>
</tr>
</tbody>
</table>

**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 KNOTMAX= 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 17 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 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 the FREQ statement is not specified, 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 45 in Chapter 2, “Shared Concepts.”

The model-options control other aspects of model formation and inference. Table 10.5 summarizes these options.

<table>
<thead>
<tr>
<th>Model Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLB</td>
<td>Requests confidence limits</td>
</tr>
<tr>
<td>INCLUDE=</td>
<td>Includes effects in all models for model selection</td>
</tr>
<tr>
<td>INFORMATIVE</td>
<td>Models missing values by using extra indicator variables</td>
</tr>
<tr>
<td>NOINT</td>
<td>Suppresses the intercept</td>
</tr>
<tr>
<td>SS3</td>
<td>Displays the Type 3 or joint tests of effects</td>
</tr>
<tr>
<td>START=</td>
<td>Includes effects in the initial model for model selection</td>
</tr>
<tr>
<td>STB</td>
<td>Displays standardized estimates</td>
</tr>
<tr>
<td>TOL</td>
<td>Displays tolerance values for the estimates</td>
</tr>
<tr>
<td>VIF</td>
<td>Displays variance inflation factors for the estimates</td>
</tr>
</tbody>
</table>

Model Options

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

CLB

requests the 100(1 − α)% 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 supported when you specify METHOD=LAR or METHOD=LASSO in the SELECTION statement.

INCLUDE=n

INCLUDED=single-effect

INCLUDED=(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. The INCLUDE=
option is not available when you specify METHOD=LAR or METHOD=LASSO in the SELECTION statement.

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 65 in Chapter 2, “Shared Concepts.”

NOINT
suppresses the intercept term that is otherwise included in the model.

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 398.

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 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, or METHOD=LASSO 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.

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 supported when you specify METHOD=LAR or METHOD=LASSO in the SELECTION statement.

VIF
produces variance inflation factors along with the parameter estimates. Variance inflation is the reciprocal of tolerance. The VIF option is not supported when you specify METHOD=LAR or METHOD=LASSO in the SELECTION statement.

OUTPUT Statement

OUTPUT OUT=CAS-libref.data-table
  < COPYVARS=variables >
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=CAS-libref.data-table
```

names the output data table for PROC LOGSELECT to use. You must specify this option before any other options. *CAS-libref.data-table* is a two-level name, where

- **CAS-libref** refers to a collection of information that is defined in the LIBNAME statement and includes the *caslib*, 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 *CAS-libref*, see the section “Using CAS Sessions and CAS Engine Librefs” on page 374.

- **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>> . . . <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 10.6.

Table 10.6 Role Interpretation

<table>
<thead>
<tr>
<th>Value</th>
<th>Observation Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not used</td>
</tr>
<tr>
<td>1</td>
<td>Training</td>
</tr>
<tr>
<td>2</td>
<td>Validation</td>
</tr>
<tr>
<td>3</td>
<td>Testing</td>
</tr>
</tbody>
</table>

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 10.7 in the OUTPUT statement to obtain additional statistics. These statistics are not available if you specify METHOD=LAR or METHOD=LASSO in the SELECTION statement. For computational formulas, see the section “Diagnostic Statistics” on page 395. 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 10.7 Keywords for OUTPUT Statement

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COOKD</td>
<td>Cook’s $D$ influence statistic</td>
</tr>
<tr>
<td>COVRATIO</td>
<td>Standard influence of observation on covariance of betas</td>
</tr>
<tr>
<td>DFFIT</td>
<td>Standard influence of observation on predicted value</td>
</tr>
<tr>
<td>H</td>
<td>Leverage, $x_i(X'X)^{-1}x_i'$</td>
</tr>
<tr>
<td>LCL</td>
<td>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.</td>
</tr>
<tr>
<td>LCLM</td>
<td>Lower bound of a $100(1 - \alpha)$% confidence interval for the expected value (mean) of the dependent variable</td>
</tr>
<tr>
<td>LIKEDIST</td>
<td>Likelihood displacement, which measures the change in the error sum of squares when the current observation is deleted</td>
</tr>
<tr>
<td>PRESS</td>
<td>$i$th residual divided by $(1 - h)$, where $h$ is the leverage and where the model has been refit without the $i$th observation</td>
</tr>
<tr>
<td>RSTUDENT</td>
<td>A studentized residual with the current observation deleted</td>
</tr>
<tr>
<td>STDERR</td>
<td>Standard error of the individual predicted value</td>
</tr>
<tr>
<td>STDPM</td>
<td>Standard error of the mean predicted value</td>
</tr>
<tr>
<td>STDR</td>
<td>Standard error of the residual</td>
</tr>
<tr>
<td>STUDENT</td>
<td>Studentized residuals, which are the residuals divided by their standard errors</td>
</tr>
</tbody>
</table>
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 66 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 of the observations 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 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 32 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.
- **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 Classification Variables and the SPLIT Option.
- **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 Classification Variables and the SPLIT Option.
- **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 “ANOVA,” “Fit Statistics,” and “Parameter Estimates” tables, which provide information about the model that is selected at each step of the selection process.

---

**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, 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. 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

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 $\text{RSS}_p$, add an effect with $k$ degrees of freedom, and denote the residual sum of squares of the resulting model by $\text{RSS}_{p+k}$, then the $F$ statistic for entry with $k$ numerator degrees of freedom and $n - (p + k) - 1$ denominator degrees of freedom is given by

$$F = \frac{(\text{RSS}_p - \text{RSS}_{p+k})/k}{\text{RSS}_{p+k}/(n - (p + k) - 1)}$$

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 32 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 $\text{RSS}_{p-k}$, then the $F$ statistic for removal with $k$ numerator degrees of freedom and $n - p - k$ denominator degrees of freedom is given by

$$F = \frac{(\text{RSS}_{p-k} - \text{RSS}_p)/k}{\text{RSS}_p/(n - p - k)}$$

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 32 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 10.8 provides formulas and definitions for these fit statistics.

**Table 10.8** Formulas and Definitions for Model Fit Summary Statistics

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Definition or Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td>Number of observations</td>
</tr>
<tr>
<td>$p$</td>
<td>Number of parameters, including the intercept</td>
</tr>
<tr>
<td>$\hat{\sigma}^2$</td>
<td>Estimate of pure error variance from fitting the full model</td>
</tr>
<tr>
<td>SST</td>
<td>Total sum of squares corrected for the mean for the dependent variable</td>
</tr>
<tr>
<td>SSE</td>
<td>Error sum of squares</td>
</tr>
<tr>
<td>ASE</td>
<td>$\frac{n}{SSE}$</td>
</tr>
<tr>
<td>MSE</td>
<td>$\frac{n - p}{SSE}$</td>
</tr>
<tr>
<td>$R^2$</td>
<td>$1 - \frac{SSE}{SST}$</td>
</tr>
<tr>
<td>ADJRSQ</td>
<td>$1 - \frac{n}{n-p}(1-R^2)$</td>
</tr>
<tr>
<td>AIC</td>
<td>$n \ln \left( \frac{SSE}{n} \right) + 2p$</td>
</tr>
<tr>
<td>AICC</td>
<td>$1 + \ln \left( \frac{SSE}{n} \right) + \frac{2(p + 1)}{n - p - 2}$</td>
</tr>
<tr>
<td>CP ($C_p$)</td>
<td>$\frac{SSE}{\hat{\sigma}^2} + 2p - n$</td>
</tr>
<tr>
<td>PRESS</td>
<td>$\sum_{i=1}^{n} \frac{r_i^2}{1 - h_i}$ where $r_i$ = residual at observation $i$ and $h_i$ = leverage of observation $i = x_i(X'X)^{-1}x_i'$</td>
</tr>
<tr>
<td>RMSE</td>
<td>$\sqrt{\text{MSE}}$</td>
</tr>
<tr>
<td>SBC</td>
<td>$n \ln \left( \frac{SSE}{n} \right) + p \ln(n)$</td>
</tr>
</tbody>
</table>
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 $Y = X\beta + \epsilon$, and the parameter estimate is denoted by $b = (X'X)^{-1}X'Y$. The subscript $i$ denotes values for the $i$th observation, and the parenthetical subscript $(i)$ means that the statistic is computed by using all observations except the $i$th observation.

The ALPHA= option in the PROC REGSELECT statement is used to set the $\alpha$ value for the confidence limit statistics.

Table 10.9 contains the diagnostic statistics and their formulas. Each statistic is computed for each observation.

<table>
<thead>
<tr>
<th>MODEL Option or Statistic</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRED ($\hat{Y}_i$)</td>
<td>$X_i b$</td>
</tr>
<tr>
<td>RES ($r_i$)</td>
<td>$Y_i - \hat{Y}_i$</td>
</tr>
<tr>
<td>H ($h_i$)</td>
<td>$x_i(X'X)^{-1}x'_i$</td>
</tr>
<tr>
<td>STDP</td>
<td>$\sqrt{h_i\hat{\sigma}^2}$</td>
</tr>
<tr>
<td>STDI</td>
<td>$\sqrt{(1 + h_i)\hat{\sigma}^2}$</td>
</tr>
<tr>
<td>STDR</td>
<td>$\sqrt{(1 - h_i)\hat{\sigma}^2}$</td>
</tr>
<tr>
<td>LCL</td>
<td>$\hat{Y}<em>i - t</em>{\frac{\alpha}{2}}$STDI</td>
</tr>
<tr>
<td>LCLM</td>
<td>$\hat{Y}<em>i - t</em>{\frac{\alpha}{2}}$STDP</td>
</tr>
<tr>
<td>UCL</td>
<td>$\hat{Y}<em>i + t</em>{\frac{\alpha}{2}}$STDI</td>
</tr>
<tr>
<td>UCLM</td>
<td>$\hat{Y}<em>i + t</em>{\frac{\alpha}{2}}$STDP</td>
</tr>
<tr>
<td>STUDENT</td>
<td>$\frac{STDR_i}{r_i}$</td>
</tr>
<tr>
<td>RSTUDENT</td>
<td>$\frac{\hat{\sigma}(i)\sqrt{1 - h_i}}{STDP}$</td>
</tr>
<tr>
<td>COOKD</td>
<td>$\frac{1}{p}\frac{\text{STUDENT}^2 \text{STDP}^2}{\text{STDR}^2}$</td>
</tr>
<tr>
<td>COVRATIO</td>
<td>$\frac{\det(\hat{\sigma}^2(X'X)^{-1})}{\hat{\sigma}(i)(X'x_i)(x_i')^{-1}}$</td>
</tr>
<tr>
<td>DFFITS</td>
<td>$\frac{(\hat{\sigma}(i)\sqrt{h_i})}{r_i}$</td>
</tr>
<tr>
<td>PRESS(predr$_i$)</td>
<td>$\frac{1 - h_i}{1}$</td>
</tr>
</tbody>
</table>
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:

```r
data mycas.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;
```

```r
proc regselect data=mycas.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 10.9 is produced by default whenever you specify a CLASS statement.

### Figure 10.9 Class Levels

<table>
<thead>
<tr>
<th>Class Level Information</th>
<th>Class Levels</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>c1</td>
<td>6 1 2 3 4 5 6</td>
<td></td>
</tr>
<tr>
<td>c2</td>
<td>3 high medium low</td>
<td></td>
</tr>
</tbody>
</table>

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 10.10 shows that for this example the parameters that correspond only to levels 3 and 5 of `c1` are in the selected model.
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:

```bash
proc regselect data=mycas.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=mycas.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 68 in Chapter 2, “Shared Concepts.”

Joint Tests and Type 3 Tests

Linear hypotheses for \( \beta \) are expressed in matrix form as

\[
H_0 : L\beta = c
\]

where \( L \) is a matrix of coefficients for the linear hypotheses and \( c \) is a vector of constants. The vector of regression coefficients \( \beta \) includes slope parameters and intercept parameters. The \( F \) statistic for testing \( H_0 \) is computed as

\[
F = (L\hat{\beta} - c)'[L\hat{V}(\beta)L']^{-1}(L\hat{\beta} - c)/r
\]

where \( \hat{V}(\beta) = \hat{\sigma}^2(X'WX)^{-1} \) is the estimated covariance matrix of \( \hat{\beta} \), 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 \( L \).
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.

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.

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:
<table>
<thead>
<tr>
<th>Code</th>
<th>Stop Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>All eligible effects are in the model.</td>
</tr>
<tr>
<td>2</td>
<td>All eligible effects have been removed.</td>
</tr>
<tr>
<td>3</td>
<td>Specified maximum number of steps done.</td>
</tr>
<tr>
<td>4</td>
<td>The model contains the specified maximum number of effects.</td>
</tr>
<tr>
<td>5</td>
<td>The model contains the specified minimum number of effects (for backward selection).</td>
</tr>
<tr>
<td>6</td>
<td>The stopping criterion is at a local optimum.</td>
</tr>
<tr>
<td>7</td>
<td>No suitable add or drop candidate could be found.</td>
</tr>
<tr>
<td>8</td>
<td>Adding or dropping any effect does not improve the selection criterion.</td>
</tr>
<tr>
<td>9</td>
<td>No candidate meets the appropriate SLE or SLS significance level.</td>
</tr>
<tr>
<td>10</td>
<td>Stepwise selection is cycling.</td>
</tr>
<tr>
<td>11</td>
<td>The model is an exact fit.</td>
</tr>
<tr>
<td>12</td>
<td>Dropping an effect would result in an empty model.</td>
</tr>
</tbody>
</table>

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

$$\hat{R}^2 = 1 - \frac{(n - i)(1 - R^2)}{n - p}$$

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 10.8 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 398.

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.

OutCASTblFull

When you specify an OUTPUT statement or a DISPLAYOUT statement to create output tables on your CAS server, the “Output CAS Tables” table displays the names, the numbers of rows and columns, and the CAS libraries in which the table is created for every output table that you requested.

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 ODS statements. These names are listed in Table 10.10.

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Required Statement / Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANOVA</td>
<td>Selected model ANOVA table</td>
<td>Default output</td>
</tr>
<tr>
<td>Candidates</td>
<td>Swap candidates at step</td>
<td>SELECTION DETAILS=ALL</td>
</tr>
<tr>
<td>ClassInfo</td>
<td>Level information from the CLASS statement</td>
<td>CLASS</td>
</tr>
<tr>
<td>Dimensions</td>
<td>Model dimensions</td>
<td>Default output</td>
</tr>
</tbody>
</table>
Table 10.10  continued

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Required Statement / Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>EntryCandidates</td>
<td>Candidates for entry at step</td>
<td>SELECTION DETAILS=ALL</td>
</tr>
<tr>
<td>FitStatistics</td>
<td>Fit statistics</td>
<td>Default output</td>
</tr>
<tr>
<td>ModelAnova</td>
<td>Model analysis of variance (Type III)</td>
<td>MODEL SS3</td>
</tr>
<tr>
<td>NObs</td>
<td>Number of observations read and used</td>
<td>Default output</td>
</tr>
<tr>
<td>OutCASTblFull</td>
<td>Library and name of the output data table, and number of rows and columns in the table</td>
<td>OUTPUT DISPLAYOUT</td>
</tr>
<tr>
<td>ParameterEstimates</td>
<td>Solutions for the parameter estimates associated with effects in the MODEL statement</td>
<td>Default output</td>
</tr>
<tr>
<td>RemovalCandidates</td>
<td>Candidates for removal at step</td>
<td>SELECTION DETAILS=ALL</td>
</tr>
<tr>
<td>SelectedEffects</td>
<td>List of selected effects</td>
<td>SELECTION</td>
</tr>
<tr>
<td>SelectionInfo</td>
<td>Information about selection settings</td>
<td>Default output</td>
</tr>
<tr>
<td>SelectionReason</td>
<td>Reason for selecting the final model</td>
<td>SELECTION</td>
</tr>
<tr>
<td>SelectionSummary</td>
<td>Summary information about the model selection steps</td>
<td>SELECTION</td>
</tr>
<tr>
<td>StopReason</td>
<td>Reason selection was terminated</td>
<td>SELECTION</td>
</tr>
<tr>
<td>Timing</td>
<td>Timing breakdown by task</td>
<td>SELECTION DETAILS</td>
</tr>
</tbody>
</table>

**Examples: 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 8 and “Loading a SAS Data Set onto a CAS Server” on page 9 in Chapter 2, “Shared Concepts.”

**Example 10.1: Model Selection with Validation**

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.

```plaintext
DATA mycas.analysisData;
  DROP i j c3Num;
  LENGTH c3$ 7;
  ARRAY x{20} x1-x20;
  DO i=1 TO 15000;
    DO j=1 TO 20;
      x{j} = ranuni(1);
    END;
    c1 = 1 + MOD(i,8);
    c2 = ranbin(1,3,.6);
    IF i < 50 THEN c3 = 'tiny'; c3Num=1; END;
    ELSE IF i < 250 THEN c3 = 'small'; c3Num=1; END;
    ELSE IF i < 600 THEN c3 = 'average'; c3Num=2; END;
    ELSE IF i < 1200 THEN c3 = 'big'; c3Num=3; END;
    ELSE c3 = 'huge'; c3Num=5; END;
    yTrue = 10 + x1 + 2*x5 + 3*x10 + 4*x20 + 3*x1*x7 + 8*x6*x7
       + 5*(c1=3)*c3Num + 8*(c1=7);
    ERROR = 5*rannor(1);
    y = yTrue + error;
    IF MOD(i,3)=1 THEN Role = 'TRAIN';
    ELSE IF MOD(i,3)=2 THEN Role = 'VAL';
    ELSE Role = 'TEST';
    OUTPUT;
  END;
RUN;
```

By construction, the true model consists of the main effects \textit{x1}, \textit{x5}, \textit{x10}, \textit{x20}, and \textit{c1} and the interaction effects \textit{x1*x7}, \textit{x6*x7}, and \textit{c1*c3}. Furthermore, you can see that only levels 3 and 7 of the classification variable \textit{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. \textbf{Output 10.1.1} shows the ASE for each role that you can compute with the following statements:

```plaintext
PROC SUMMARY DATA=mycas.analysisData;
  CLASS role;
  WAYS 1;
  VAR error;
  OUTPUT OUT=ASE USU=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 10.1.1  Oracle ASE Values by Role

<table>
<thead>
<tr>
<th>Role</th>
<th>Oracle ASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEST</td>
<td>25.5784</td>
</tr>
<tr>
<td>TRAI N</td>
<td>25.4008</td>
</tr>
<tr>
<td>VAL</td>
<td>25.8993</td>
</tr>
</tbody>
</table>

The ASE values shown Output 10.1.1 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:

```sas
proc regselect data=mycas.analysisData;
  partition roleVar=role(train='TRAIN' validate='VAL' test='TEST');
  class c1 c2 c3;
  model y = c1|c2|c3|x1|x2|x3|x4|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;
```

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, 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 10.1.2 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 661 parameters.

**Output 10.1.2  Number of Observations, Class Levels, and Dimensions**

**The REGSELECT Procedure**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Observations Read</td>
<td>15000</td>
</tr>
<tr>
<td>Number of Observations Used</td>
<td>15000</td>
</tr>
<tr>
<td>Number of Observations Used for Training</td>
<td>5000</td>
</tr>
<tr>
<td>Number of Observations Used for Validation</td>
<td>5000</td>
</tr>
<tr>
<td>Number of Observations Used for Testing</td>
<td>5000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class Level Information</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Class Levels Values</td>
<td></td>
</tr>
<tr>
<td>c1</td>
<td>8 1 2 3 4 5 6 7 8</td>
</tr>
<tr>
<td>c2</td>
<td>4 0 1 2 3</td>
</tr>
<tr>
<td>c3</td>
<td>5 average big huge small tiny</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dimensions</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Effects</td>
<td>278</td>
</tr>
<tr>
<td>Number of Parameters</td>
<td>661</td>
</tr>
</tbody>
</table>

Output 10.1.3 shows the “Selection Summary” table. You see that 18 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 18 is not used as the selected model. Instead the model at step 10 (where the validation ASE achieves a local minimum value) is selected. The “Stop Reason,” “Selection Reason,” and “Selected Effects” in Output 10.1.4 tables provide this information.
Output 10.1.3 Selection Summary

The REGSELECT Procedure

Selection Details

<table>
<thead>
<tr>
<th>Step</th>
<th>Effect Entered</th>
<th>Effect Number</th>
<th>Validation ASE</th>
<th>p Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Intercept</td>
<td>1</td>
<td>98.3895</td>
<td>.</td>
</tr>
<tr>
<td>1</td>
<td>c1</td>
<td>2</td>
<td>34.8572</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>2</td>
<td>x7</td>
<td>3</td>
<td>32.5531</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>3</td>
<td>x6</td>
<td>4</td>
<td>31.0646</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>4</td>
<td>x20</td>
<td>5</td>
<td>29.7078</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>5</td>
<td>x6*x7</td>
<td>6</td>
<td>29.2210</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>6</td>
<td>x10</td>
<td>7</td>
<td>28.6683</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>7</td>
<td>x1</td>
<td>8</td>
<td>28.3250</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>8</td>
<td>x5</td>
<td>9</td>
<td>27.9766</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>9</td>
<td>c3</td>
<td>10</td>
<td>27.8288</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>10</td>
<td>c1*c3</td>
<td>11</td>
<td>25.9701*</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>11</td>
<td>x10*c1</td>
<td>12</td>
<td>26.0696</td>
<td>0.0109</td>
</tr>
<tr>
<td>12</td>
<td>x4</td>
<td>13</td>
<td>26.1594</td>
<td>0.0128</td>
</tr>
<tr>
<td>13</td>
<td>x4*x10</td>
<td>14</td>
<td>26.1814</td>
<td>0.0035</td>
</tr>
<tr>
<td>14</td>
<td>x20*c1</td>
<td>15</td>
<td>26.3294</td>
<td>0.0156</td>
</tr>
<tr>
<td>15</td>
<td>x1*c3</td>
<td>16</td>
<td>26.3945</td>
<td>0.0244</td>
</tr>
<tr>
<td>16</td>
<td>x1*x7</td>
<td>17</td>
<td>26.3632</td>
<td>0.0270</td>
</tr>
<tr>
<td>17</td>
<td>x7*x10</td>
<td>18</td>
<td>26.4120</td>
<td>0.0313</td>
</tr>
<tr>
<td>18</td>
<td>x1*x20</td>
<td>19</td>
<td>26.4330</td>
<td>0.0871</td>
</tr>
</tbody>
</table>

* Optimal Value Of Criterion

Output 10.1.4 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 10 is selected where Validation ASE is 25.9701.

Selected Effects: Intercept c1 c3 c1*c3 x1 x5 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.

Output 10.1.5 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 oracle values for the true model, as shown in Output 10.1.1.
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 10.1.6** shows these fit statistics for the final model at step 18. 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 10.

**Output 10.1.5** Fit Statistics for the Selected Model

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root MSE</td>
<td>5.03976</td>
</tr>
<tr>
<td>R-Square</td>
<td>0.74483</td>
</tr>
<tr>
<td>Adj R-Sq</td>
<td>0.74246</td>
</tr>
<tr>
<td>AIC</td>
<td>21222</td>
</tr>
<tr>
<td>AICC</td>
<td>21223</td>
</tr>
<tr>
<td>SBC</td>
<td>16527</td>
</tr>
<tr>
<td>ASE (Train)</td>
<td>25.16041</td>
</tr>
<tr>
<td>ASE (Validate)</td>
<td>25.97010</td>
</tr>
<tr>
<td>ASE (Test)</td>
<td>25.83436</td>
</tr>
</tbody>
</table>

**Output 10.1.6** Fit Statistics for the Model at Step 18

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root MSE</td>
<td>5.01386</td>
</tr>
<tr>
<td>R-Square</td>
<td>0.74862</td>
</tr>
<tr>
<td>Adj R-Sq</td>
<td>0.74510</td>
</tr>
<tr>
<td>AIC</td>
<td>21194</td>
</tr>
<tr>
<td>AICC</td>
<td>21196</td>
</tr>
<tr>
<td>SBC</td>
<td>16648</td>
</tr>
<tr>
<td>ASE (Train)</td>
<td>24.78688</td>
</tr>
<tr>
<td>ASE (Validate)</td>
<td>26.43304</td>
</tr>
<tr>
<td>ASE (Test)</td>
<td>26.07078</td>
</tr>
</tbody>
</table>

Output 10.1.7 shows part of the “Parameter Estimates” table for the selected model at step 10 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 10.1.7** Part of the Parameter Estimates Table for the Selected Model

| Parameter | DF | Estimate | Standardized Estimate | Standard Error | t Value | Pr > |t| |
|-----------|----|----------|-----------------------|----------------|---------|------|------|
| Intercept | 1  | 8.558668 | 0                     | 3.581926       | 2.39    | 0.0169|
| c1 1      | 1  | 13.177235| 0.438872              | 5.043039       | 2.61    | 0.0090|
| c1 2      | 1  | -2.800921| -0.093285             | 4.602421       | -0.61   | 0.5428|
| c1 3      | 1  | 9.689426 | 0.322709              | 5.043363       | 1.92    | 0.0548|
| c1 4      | 1  | 8.182340 | 0.272515              | 5.042136       | 1.62    | 0.1047|
| c1 5      | 1  | 3.097216 | 0.103154              | 5.041119       | 0.61    | 0.5390|
| c1 6      | 1  | 0.543642 | 0.018106              | 5.043466       | 0.11    | 0.9142|
| c1 7      | 1  | 11.857267| 0.394910              | 5.042602       | 2.35    | 0.0187|
| c1 8      | 0  | 0        | 0                     | 0              | .       | .    |
The magnitudes of the standardized estimates and the \( t \) statistics of the parameters of the effect \( c_1 \) 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 \( c_1 \) 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:

```sas
proc regselect data=mycas.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|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 10.1.8 shows the “Dimensions” table. You can see that because the columns in the design matrix that correspond to the levels of \( c_1 \) are treated as separate effects, the selection is now from 439 effects, even though the number of parameters is unchanged.

**Output 10.1.8** Dimensions with \( c_1 \) Split

<table>
<thead>
<tr>
<th>Dimensions</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Effects</td>
<td>439</td>
</tr>
<tr>
<td>Number of Parameters</td>
<td>661</td>
</tr>
</tbody>
</table>

Output 10.1.9 shows the selected effects. You can see that, as anticipated, the selected model now depends on only levels 3 and 7 of \( c_1 \).

**Output 10.1.9** Selected Effects with \( c_1 \) Split

**Selected Effects:** Intercept \( c_1.3\) \( c_1.7\) \( c_1.3\)\( c_3\) \( x_1\) \( x_5\) \( x_6\) \( x_7\) \( x_6x_7\) \( x_{10}\) \( x_{20}\)

Finally, the fit statistics for the selected model are shown in **Output 10.1.10**.

**Output 10.1.10** Fit Statistics for the Selected Model with \( c_1 \) Split

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root MSE</td>
<td>5.04060</td>
</tr>
<tr>
<td>R-Square</td>
<td>0.74325</td>
</tr>
<tr>
<td>Adj R-Sq</td>
<td>0.74238</td>
</tr>
<tr>
<td>AIC</td>
<td>21195</td>
</tr>
<tr>
<td>AICC</td>
<td>21195</td>
</tr>
<tr>
<td>SBC</td>
<td>16311</td>
</tr>
<tr>
<td>ASE (Train)</td>
<td>25.31622</td>
</tr>
<tr>
<td>ASE (Validate)</td>
<td>25.98055</td>
</tr>
<tr>
<td>ASE (Test)</td>
<td>25.76059</td>
</tr>
</tbody>
</table>

If you compare the ASE values for this model in **Output 10.1.10** with the oracle values in **Output 10.1.1** and the values for the model without splitting \( c_1 \) in **Output 10.1.5**, you see that this more parsimonious model produces the best predictive performance for the test data of all the models considered in this example.
References


Chapter 11
The TREESPLIT Procedure

Contents

Overview: TREESPLIT Procedure ................................................................. 414
PROC TREESPLIT Features ................................................................. 414
Using CAS Sessions and CAS Engine Librefs .................................. 415
Getting Started: TREESPLIT Procedure ........................................... 416
Syntax: TREESPLIT Procedure ................................................................. 421
  PROC TREESPLIT Statement .......................................................... 422
  AUTOTUNE Statement ................................................................. 427
  CLASS Statement ....................................................................... 431
  CODE Statement ............................................................... 431
  FREQ Statement ................................................................. 431
  GROW Statement ................................................................. 431
  MODEL Statement ................................................................. 433
  OUTPUT Statement ................................................................. 433
  PARTITION Statement ............................................................. 434
  PRUNE Statement ................................................................. 435
Details: TREESPLIT Procedure ................................................................. 436
  Building a Decision Tree .......................................................... 436
  Splitting Criteria ................................................................. 439
  Primary and Surrogate Splitting Rules ..................................... 441
  Pruning .......................................................... 442
  Scoring .......................................................... 446
  Measures of Model Fit .......................................................... 447
  Variable Importance ............................................................. 447
  Parameter Tuning ................................................................. 449
  ODS Table Names ................................................................. 450
  ODS Graphics ................................................................. 451
  INPUT and TARGET Statement Syntax ................................... 451
Examples: TREESPLIT Procedure ............................................................ 453
  Example 11.1: Creating a Binary Classification Tree with Validation Data .... 453
  Example 11.2: Creating a Regression Tree ......................................... 459
  Example 11.3: Assessing Variable Importance .................................. 462
References ................................................................. 463
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 primary 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 primary splitting rules, surrogate 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 436.

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, including surrogate splits.

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
provides various methods of handling missing values, including surrogate rules
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 TRESPLIT procedure uses ODS Graphics to create plots as part of its output. For information about the statistical graphics available with the TRESPLIT procedure, see the PLOTS options in the PROC TRESPLIT statement and the section “ODS Graphics” on page 451.

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 mycas cas sessref=mysess;

The CAS statement creates the CAS session named mysess, and the LIBNAME statement creates the mycas 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:
Getting Started: TREENSPLIT 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 8 and “Loading a SAS Data Set onto a CAS Server” on page 9 in Chapter 2, “Shared Concepts.”

This example demonstrates basic features of the TREENSPLIT procedure that you can use to build a classification tree. The data are measurements of four physical attributes in 150 samples of iris flowers. Each flower is from one of three species of iris, and the goal of the analysis is a model that classifies iris samples by species.

The following statements load the mycas.iris data into your CAS session. For this example, the statements assume that your CAS engine libref is named mycas, but you can substitute any appropriately defined CAS engine libref.

```plaintext
data mycas.iris;
  set sashelp.iris;
run;
```

The following statement prints the first 10 observations of mycas.iris, as shown in Figure 11.1:

```plaintext
proc print data=mycas.iris(obs=10); run;
```

![Figure 11.1 Partial Listing of mycas.iris](image)

The variable Species is a nominal categorical variable with the levels Setosa, Versicolor, and Virginica (the species names), and the four attribute variables are continuous.

The following statements use the TREENSPLIT procedure to create a classification tree:
ods graphics on;

proc treesplit data=mycas.iris maxdepth=2;
  class Species;
  model Species = PetalLength PetalWidth SepalLength SepalWidth;
  grow entropy;
  prune none;
run;

The MAXDEPTH= option specifies the maximum depth of the tree to be grown. The MODEL statement specifies Species as the response variable and the variables to the right of the equal sign as the predictor variables. The inclusion of Species 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 439.

The default output includes several informational tables, which are shown in Figure 11.2 through Figure 11.6. The “Model Information” table in Figure 11.2 provides information about the model and the methods that are used to grow and prune the tree. In this example, the tree depth and the number of leaves are the same before and after pruning, because the tree is not pruned.

**Figure 11.2** Model Information

<table>
<thead>
<tr>
<th>Model Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Split Criterion</td>
</tr>
<tr>
<td>Pruning Method</td>
</tr>
<tr>
<td>Max Branches per Node</td>
</tr>
<tr>
<td>Max Tree Depth</td>
</tr>
<tr>
<td>Tree Depth Before Pruning</td>
</tr>
<tr>
<td>Tree Depth After Pruning</td>
</tr>
<tr>
<td>Number of Leaves Before Pruning</td>
</tr>
<tr>
<td>Number of Leaves After Pruning</td>
</tr>
</tbody>
</table>

The “Observation Information” table in Figure 11.3 provides the numbers of observations that are read and used.

**Figure 11.3** Observation Information

<table>
<thead>
<tr>
<th>Training</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Observations Read</td>
</tr>
<tr>
<td>Number of Observations Used</td>
</tr>
</tbody>
</table>
The tree diagram in Figure 11.4, which is produced by default when ODS Graphics is enabled, provides an overview of the tree as a classifier.

**Figure 11.4** Overview Diagram of Final Tree

The tree is constructed starting with all the observations in the root node (labeled 0). This node is split into one internal node and one leaf node (1 and 2, respectively), and node 1 is further split into leaf nodes (3 and 4).

The color of the bar in each leaf node indicates the most frequent level of *Species* among the observations in that node; this is also the classification level 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 11.5 provides more detail about the nodes and splits.
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 150 samples. Because no MAXBRANCH= option is 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 150 observations such that all samples for which PetalLength \( \geq 21.8 \) are assigned to node 1 and all samples for which PetalLength < 21.8 are assigned to node 2. This is the primary splitting rule for node 0. In this training phase, 100 samples are assigned to node 1 and 50 samples are assigned to node 2.
Figure 11.5 also indicates that node 1 contains 0 samples with level 3. The legend shows that level 3 corresponds to Species=Setosa, so node 1 contains no samples of the species Setosa. Similarly, the diagram indicates that node 2 contains 0 samples with level 1 or level 2, and the legend shows that level 1 corresponds to Species=Versicolor and level 2 corresponds to Species=Virginica, so node 2 contains no samples of the species Versicolor or Virginica.

The primary splitting rule for node 1 consists of the variable PetalWidth and the split value 17.8. All samples for which PetalWidth ≥ 17.8 are assigned to node 3 (which then contains 46 samples), and all samples for which PetalWidth < 17.8 are assigned to node 4 (which then contains 54 samples).

The resulting classification tree yields simple rules for predicting the species. For example, a sample for which PetalLength ≥ 21.8 and PetalLength < 17.8 is predicted to be from the species Versicolor (node 4 indicates that level 1 has the highest proportion of observations, and the legend shows that level 1 corresponds to Species=Versicolor).

Figure 11.5 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 11.6 displays fit statistics for the tree model.

**Figure 11.6 Fit Statistics**

<table>
<thead>
<tr>
<th>The TREESPLIT Procedure</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Fit Statistics for Selected Tree</td>
<td></td>
</tr>
<tr>
<td>Number of Leaves</td>
<td>Misclassification Rate</td>
</tr>
<tr>
<td>Training</td>
<td>3</td>
</tr>
</tbody>
</table>

The misclassification rate is the total proportion of the 150 iris samples that were misclassified. The following numbers of samples were misclassified in the terminal nodes:

- node 2: 0
- node 4: 1
- node 5: 5

So the total misclassification rate is \((0 + 1 + 5) / 150 = 0.04\).
Syntax: TREESPLIT Procedure

The following statements and options are available in the TREESPLIT procedure:

```
PROC TREESPLIT < options > ;
   AUTOTUNE < options > ;
   CLASS variables ;
   CODE FILE=filename ;
   FREQ variable ;
   GROW criterion < options > ;
   MODEL response = variable . . . ;
   OUTPUT OUT=CAS-libref.data-table output-options ;
   PARTITION < partition-options > ;
   PRUNE prune-method < (prune-options) > ;
```

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 equal 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 11.1 summarizes the options in the PROC TREESPLIT statement.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Basic Options</strong></td>
<td></td>
</tr>
<tr>
<td>NUMBIN=</td>
<td>Specifies the number of bins to use for continuous variables</td>
</tr>
<tr>
<td>NOPRINT</td>
<td>Suppresses ODS output</td>
</tr>
<tr>
<td>NSURROGATES=</td>
<td>Specifies the number of surrogate rules to create</td>
</tr>
<tr>
<td>OUTMODEL=</td>
<td>Specifies the data table to which the decision tree model is to be saved</td>
</tr>
<tr>
<td>PLOTS=</td>
<td>Specifies options for plots</td>
</tr>
<tr>
<td>PRUNINGTABLE</td>
<td>Creates a table that contains the pruning information</td>
</tr>
<tr>
<td>SEED=</td>
<td>Specifies the random number seed to use for autotuning</td>
</tr>
<tr>
<td><strong>Splitting Options</strong></td>
<td></td>
</tr>
<tr>
<td>ASSIGNMISSING=</td>
<td>Specifies how to handle missing values in a predictor variable</td>
</tr>
<tr>
<td>MAXBRANCH=</td>
<td>Specifies the maximum number of child nodes per parent node</td>
</tr>
<tr>
<td>MAXDEPTH=</td>
<td>Specifies the maximum tree depth</td>
</tr>
<tr>
<td>MINLEAFSIZE=</td>
<td>Specifies the minimum number of observations per child node</td>
</tr>
<tr>
<td>MINUSEINSEARCH=</td>
<td>Specifies the minimum number of observations to use with the USEINSEARCH policy for handling missing values</td>
</tr>
<tr>
<td>SPLITONCE</td>
<td>Specifies that a variable can be used to split only once per branch</td>
</tr>
</tbody>
</table>

You can specify the following options:

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 after all surrogate rules have been applied during scoring. 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.

Both this option and the NSURROGATES= option affect how missing values are handled in model training and scoring. For more information about how these rules are applied, see the section “Primary and Surrogate Splitting Rules” on page 441.

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 predictor variable, then the observation is assigned to the child node that contains the most training observations.

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.

DATA=\texttt{CAS-libref.data-table}

names the input data table for PROC TREESPLIT to use. \texttt{CAS-libref.data-table} is a two-level name, where
CAS-libref refers to a collection of information that is defined in the LIBNAME statement and includes the caslib, 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 CAS-libref, see the section “Using CAS Sessions and CAS Engine Librefs” on page 415.

data-table specifies the name of the input data table.

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 one. The default is calculated by the following equation, where b is the value of the MAXBRANCH= option in the PROC TREESPLIT statement:

\[
\text{MaxDepth} = \left\lfloor \frac{10}{\log_2 (b)} \right\rfloor
\]

This value can be tuned with the AUTOTUNE statement.

MINLEAFSIZE=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.

MINUSERINSEARCH=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.

NOPRINT
suppresses the generation of ODS output.

NUMBIN=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=20. This value can be tuned with the AUTOTUNE statement.
**NSURROGATES=** number

specifies the number of surrogate rules to create for each splitting rule, where number is an integer greater than 0. Surrogate rules are backup splitting rules that are used when the variable that corresponds to the primary splitting rule is missing.

Both this option and the ASSIGNMISSING= affect training and scoring. For more information, see the section “Primary and Surrogate Splitting Rules” on page 441.

By default, NSURROGATES=0.

**OUTMODEL=** CAS-libref.data-table

names the output data table to which to save the decision tree model. CAS-libref.data-table is a two-level name, where

- **CAS-libref** refers to a collection of information that is defined in the LIBNAME statement and includes the caslib, 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 CAS-libref, see the section “Using CAS Sessions and CAS Engine Librefs” on page 415.

- **data-table** specifies the name of the input data table.

The data table that results from this option contains information about each node in the decision tree, including the splitting variables, the child nodes, the number of observations at each node, and the predicted response at each node.

**PLOTS < (global-plot-option) > < = 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.

- **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 TREPESPLIT 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

PRUNINGTABLE outputs a table of the pruning results.

SEED=number specifies the initial seed for random number generation for cross validation. 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.

AUTOTUNE Statement

AUTOTUNE < options > ;

The AUTOTUNE statement searches for the best combination of values of the MAXDEPTH= and NUMBIN= options in the PROC TREPESPLIT statement and the criterion in the GROW statement. Trees created as a result of using autotuning are not pruned.

You can specify the following options:

FRACTION=number specifies the fraction of all data to be used 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). 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 validation partition is usually sufficient for estimating error; a single partition is more efficient than cross validation in terms of the total execution time.
You cannot specify this option in combination with the KFOLD= option. If a PARTITION statement is specified, the validation partition defined in that statement is used, and this option is ignored.

By default, FRACTION=0.3.

\textbf{KFOLD=number}

specifies the number of partition folds in the cross validation process, where \textit{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 \textit{number} of training executions (on \textit{number}–1 data folds) and \textit{number} of scoring executions (on 1 hold-out fold). Thus, the evaluation time is increased by approximately a factor of \textit{number}. For small to medium data tables or for unbalanced data tables, cross validation provides on average a better representation of error across the entire data table (a better generalization error).

You cannot specify this option in combination with the FRACTION= option. If a PARTITION statement is specified, the validation partition defined in that statement is used, and this option is ignored.

\textbf{MAXEVALS=number}

specifies the maximum number of configuration evaluations allowed for the tuner, where \textit{number} must be an integer greater than or equal to 3. When the \textit{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 that either a single model be trained and scored on a validation partition or that a model undergo a number of training and scoring executions equal to the KFOLD= option for cross validation. The MAXEVALS= option might lead to termination before the value of the MAXITER= or MAXTIME= option is reached.

By default, MAXEVALS=50.

\textbf{MAXITER=number}

specifies the maximum number of iterations of the optimization tuner, where \textit{number} must be greater than or equal to 1. Each iteration usually involves a number of objective evaluations up to the value of the POPSIZE= option. The MAXITER= option might lead to termination before the value of the MAXEVALS= or MAXTIME= option is reached.

By default, MAXITER=5.

\textbf{MAXTIME=number}

specifies the maximum time (in seconds) allowed for the tuner, where \textit{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 \textit{number}. The MAXTIME= option might lead to termination before the value of the MAXEVALS= or MAXITER= option is reached.

By default, MAXTIME=36000.

\textbf{POPSIZE=number}

specifies the maximum number of evaluations in one iteration (population), where \textit{number} must be greater than or equal to 1. In some cases, the tuner algorithm might generate a number of new configurations smaller than \textit{number}.

By default, POPSIZE=10.
TUNINGPARAMETERS=(suboption | ... | < suboption >)
TUNEPMRS=(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, and 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.

**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 hyperparameters to tune.

You can specify the following tuning-parameter-options:

**STANDARD**
tunes using the default bounds and initial values for all of the 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.

By default, USEPARAMETERS=COMBINED.
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 43.

You can specify only one CLASS statement.

NOTE: All classification levels are padded or truncated to 32 characters.

CODE Statement

CODE FILE=filename;

The CODE statement converts the final tree to SAS DATA step code that you can use for scoring. You must specify the following option:

FILE=filename

specifies the name of the file to write the SAS score code to.

If you do not specify this statement, no SAS DATA step code is output.

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 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 the FREQ statement is not specified, 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:
CHAI < (options) >
for categorical predictor variables, CHAI uses the value (as specified in the ALPHA= option) of a
d 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, CHAI 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**: 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.

The value of criterion can be tuned with the AUTOTUNE statement.

---

**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=CAS-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=CAS-libref.data-table
names the output data table for PROC TREESPLIT to use. You must specify this option before any other options. CAS-libref.data-table is a two-level name, where

CAS-libref refers to a collection of information that is defined in the LIBNAME statement and includes the caslib, 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 CAS-libref, see the section “Using CAS Sessions and CAS Engine Librefs” on page 415.

data-table specifies the name of the output data table.

You can also specify the following option:

COPYVARS=(variables)
lists one or more variables from the input data table to be transferred to the output data table.

**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 66 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 of the observations 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 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.
The PRUNE statement specifies the pruning method and related options.

You can specify the following `prune-methods`. The default `prune-method` is OFF.

**C45**

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 442. 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**

requests cost-complexity pruning (Breiman et al. 1984; Quinlan 1987; Zhang and Singer 2010). You can specify this pruning method for both classification trees (which have a categorical response) and regression trees (which have a continuous response).

PROC TREESPLIT uses the validation data for subtree selection. The `PARTITION` statement is required. PROC TREESPLIT generates a cost-complexity pruning plot that shows the error metric for the training and validation data. The error metric is the misclassification rate for classification trees and the average square error (ASE) for regression trees.

You can specify the following `prune-options`:

**ALPHA=**`number`

selects the subtree whose cost-complexity value $\alpha$ is equal to `number`.

**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`. When LEAVES=ALL, the largest tree is selected.

**OFF**

turns off pruning completely. No pruning is performed, and no pruning plots are generated.

**REDUCEDERROR**

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 442.
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. When LEAVES=ALL, the largest tree is selected.

By default, PRUNE=OFF.

---

**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 439.

The best predictor variable and split value (or set of split values) is called the primary splitting rule. You can also request one or more surrogate splitting rules, which are determined in a fashion similar to the primary splitting rule and are applied when the primary splitting rule’s variable is missing. Each nonterminal node has a primary splitting rule and zero or more surrogate splitting rules. For more information about surrogate splitting rules, see the section “Primary and Surrogate Splitting Rules” on page 441.

Each region, which corresponds to a terminal node, is described by a set of primary and surrogate 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 11.7, Figure 11.8, and Figure 11.9 illustrate this process. Figure 11.7 shows a classification tree of depth 2.
Figure 11.7 First Two Splits for the mycas.hmeq Data Table

Classification Tree for BAD

BAD  0  1
Chapter 11: The TREESPLIT Procedure

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 primary splitting rule for node 0. Observations in which Debtinc < 43.77156 are assigned to node 2, and observations in which Debtinc ≥ 43.77156 are assigned to node 1.

Figure 11.8 presents a scatter plot of the predictor space for the primary 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.

Next, the algorithm determines the primary 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 11.9 presents a scatter plot of the predictor space for the primary 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.
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 436, 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 $\tau$ 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 \( \tau_b \) denotes the \( b \)th child node, \( p(\tau_b|\tau) \) is the proportion of observations in \( \tau \) that are assigned to \( \tau_b \), and \( B \) is the number of branches after splitting \( \tau \).

**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 \( \tau \) 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 \( j \)th 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 \( \tau \), \( Y_i \) is the response value of observation \( i \), and \( \overline{Y} \) is the average response of the observations in \( \tau \).

**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

$$F = \frac{SS_{between}/(B - 1)}{SS_{within}/(N(\tau) - B)}$$

where

$$SS_{between} = \sum_{b=1}^{B} N(\tau_b)(\bar{Y}(\tau_b) - \bar{Y}(\tau))^2$$

$$SS_{within} = \sum_{b=1}^{B} \sum_{i=1}^{N(\tau_b)} (Y_{bi} - \bar{Y}(\tau_b))^2$$

**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 $\alpha$ is the significance level of a test and $m$ is the number of independent tests.

**Primary and Surrogate Splitting Rules**

During training, PROC TREESPLIT creates a primary splitting rule for each nonterminal node by examining all the node’s observations’ variables and their values and calculating which variable and which of its values to split on such that the observations within each resulting child node are as similar as possible. If you specify a value, $x$, for the NSURROGATES= option, then PROC TREESPLIT creates $x$ surrogate rules in a similar fashion, each time using a different variable that has a similar predictive ability. The TREESPLIT procedure uses the method of Breiman et al. (1984) to determine surrogate rules. Surrogate rules are ordered by agreement, which is measured by the proportion of observations that have a nonmissing value in the rule’s predictor variable that two rules assign to the same branch. For example, if the primary splitting rule assigns 100 observations to a child node, surrogate rule A assigns 40 observations to the same child node, and surrogate rule B assigns 50 observations to the same child node, then surrogate rule B is applied before surrogate rule A. Surrogate rules enable you to make better use of the data.

During scoring, the primary and surrogate splitting rules are applied as follows:

1. The primary splitting rule is applied if the primary rule’s variable is not missing; otherwise,

2. The surrogate rule that has the highest agreement with the primary rule is applied if the first surrogate rule’s variable is not missing; otherwise,
3. Each subsequent splitting rule (ordered by agreement) is applied if the preceding rule has a missing value.

4. If all of the surrogate rules’ variables are missing, then the default splitting rule (as specified in the ASSIGNMISSING= option) is used. The default splitting rule enables all data to be scored even if the primary rule and all surrogate rules cannot be used on a particular observation.

Table 11.2 shows an example set of rules for a particular node in a decision tree where X, Y, and Z are three continuous predictors.

<table>
<thead>
<tr>
<th>Rule</th>
<th>Assign to Branch 1</th>
<th>Assign to Branch 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primary</td>
<td>X ≥ 1</td>
<td>X &lt; 1</td>
</tr>
<tr>
<td>Surrogate 1</td>
<td>X is missing, Y &lt; 0</td>
<td>X is missing, Y ≥ 0</td>
</tr>
<tr>
<td>Surrogate 2</td>
<td>X and Y are missing, Z ≥ 100</td>
<td>X and Y are missing, Z &lt; 100</td>
</tr>
<tr>
<td>Default</td>
<td>None</td>
<td>X, Y, and Z are missing</td>
</tr>
</tbody>
</table>

By default, NSURROGATES=0. If a variable is used as a surrogate, you can see exactly how it is used in the SAS DATA step code that is generated when you specify the CODE statement.

Pruning

The TREESPLIT procedure creates a classification or regression tree by first growing a tree as described in the section “Splitting Criteria” on page 439. 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 11.10 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.
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

$$CC(T) = R(T) + \alpha |T|$$

where $R(T)$ represents $T$’s error rate, $|T|$ represents the number of leaves on $T$, and the complexity parameter $\alpha$ 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 $\alpha$, there is a subtree of $T$ that minimizes cost complexity. When $\alpha = 0$, this is the full tree, $T_0$. As $\alpha$ increases, the corresponding subtree becomes progressively smaller, and the subtrees are in fact nested. Then, at some value of $\alpha$, the root node has the minimal cost complexity for any $\alpha$ 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 $\alpha$; that is,

- $[0, \alpha_1) =$ interval where $T_0$ (the full tree) has minimal cost complexity
- $[\alpha_1, \alpha_2) =$ interval where $T_1$ has minimal cost complexity
- ... 
- $[\alpha_m, \infty) =$ interval where $T_m$ (the root node) has minimal cost complexity

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 $\alpha$. This determination is performed by using the validation partition, which you specify in the required PARTITION statement to reserve a validation holdout sample. The subtree in the pruning sequence that has the lowest validation error rate is selected as the final tree.

You can use the LEAVES= option in the PRUNE statement to select a tree that has a specified number of leaves. Alternatively, you can use the ALPHA= option in the PRUNE statement to select a tree that has a corresponding value of the cost-complexity value $\alpha$.

In order to use cost-complexity pruning, you must specify the PARTITION statement. In the cost-complexity pruning plot that it produces, PROC TREESPLIT displays the error rate $R(T)$ as a function of the number of leaves $|T|$ for both the training and validation data. This plot uses a vertical reference line to indicate the final selected tree, the tree that has the minimum $R(T)$ for the validation data. This plot can help you identify a smaller tree that has only a slightly higher validation error rate. You could then use the LEAVES= option in a subsequent run of PROC TREESPLIT to obtain the final selected tree that has the number of leaves that you specify. For an example of this plot, see Output 11.1.5 in “Example 11.1: Creating a Binary Classification Tree with Validation Data” on page 453.

### 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 $l$):

$$
\alpha = 1 - \frac{\Gamma (N_l + 1)}{\Gamma (F_l + 1) \Gamma (N_l - F_l) \int_0^{p_l} v^{F_l} (1 - v)^{N_l - F_l + 1} dv}
$$

Here the confidence level $\alpha$ 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

$$
\Gamma (x) = \int_0^\infty v^{x-1} e^{-v} dv
$$

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 (`mycas.hmeq`) is scored after the tree model has been created:

```
proc treesplit data=mycas.hmeq;
  class Bad Delinq Derog Job Ninq Reason;
  model Bad = Delinq Derog Job Ninq Reason;
  output out=mycas.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:

```sas
proc treesplit data=mycas.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:

  \[
  \text{Misc} = \frac{1}{N_0} \sum \left\{ \begin{array}{ll}
  0 & \text{if prediction is correct} \\
  1 & \text{otherwise}
  \end{array} \right. 
  \]

- 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 predictor variables are useful for predicting the response variable, and others are not. You can use the TREESPLIT procedure to select the most useful predictor variables based on variable importance. (See “Example 11.3: Assessing Variable Importance” on page 462.) Variable importance is an indication of which predictor variables are most useful for predicting the response variable.

The most important variables might not be the ones near the top of the tree. PROC TREESPLIT measures variable importance based on the following metrics:
• Count-based variable importance simply counts the number of times in the tree that a particular variable is used in a split.

• Surrogate-count-based variable importance tallies the number of times that a variable is used in a surrogate splitting rule.

• RSS-based variable importance measures variable importance based on the change of RSS when a split is found at a node. The change is

\[ \Delta_d = \text{RSS}_d - \sum_i \text{RSS}^d_i \]

where \( d \) denotes the node, \( i \) denotes the index of a child of this node, \( \text{RSS}_d \) is the RSS if the node is treated as a leaf, and \( \text{RSS}^d_i \) is the RSS of the node after it has been split. RSS is defined as follows:

- For classification trees, the RSS is defined as

\[ \text{RSS} = \sum_{\lambda} \sum_{\Phi} N^\lambda_\Phi \left[ \sum_{\tau \neq \Phi} \left( P^\lambda_\tau \right)^2 + \left( 1 - P^\lambda_\Phi \right)^2 \right] \]

where \( \Phi \) is the actual response level, \( N^\lambda_\Phi \) is the number of observations in leaf \( \lambda \) that have the response level \( \Phi \), \( P^\lambda_\tau \) is the posterior probability for the response level \( \tau \) on leaf \( \lambda \), and \( P^\lambda_\Phi \) is the posterior probability for the actual response level \( \Phi \) on leaf \( \lambda \).

- For regression trees, the RSS is defined as

\[ \text{RSS} = \sum_{\lambda} \sum_{i \in \lambda} \left( y_i - \hat{y}^T_\lambda \right)^2 \]

where \( i \) is an observation in leaf \( \lambda \), \( y_i \) is the predicted value of the response variable of the observation \( i \), and \( \hat{y}^T_\lambda \) is the actual value of the response variable on leaf \( \lambda \).

If the change in RSS is negative (which is possible when you use validation data), then the change is set to 0.

If surrogate rules are in effect, they are also credited with a portion of the change in RSS. The credit is proportional to the agreement between the primary and surrogate splitting rules at the node. The agreement at node \( d \), \( \kappa_d \), is defined as

\[ \kappa_d = \sum_i \frac{N_i}{N_d} \]

where \( N_d \) is the number of nonmissing observations and \( N_i \) is the number of observations that were assigned to \( i \) by both the primary and surrogate rules.

The change in RSS from the surrogate rules is defined as

\[ \Delta_d = \kappa_d \left( \text{RSS}_d - \sum_i \text{RSS}^d_i \right) \]

The RSS-based importance is then defined as the following, where \( D \) is the total number of nodes:

\[ \sum_{d=1}^D \Delta_d \]
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.

Parameter Tuning

The quality of the predictive model that PROC TREESPLIT 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. The AUTOTUNE statement engages the optimization algorithm (tuner), which searches for the best possible combination of values of these select hyperparameters while trying to minimize the objective function. The objective function is a validation error estimate (misclassification error for nominal targets or average square error for interval targets). The tuning process includes multiple iterations; each iteration usually involves multiple objective function evaluations. Each objective function evaluation can consist of one or several training and scoring executions as follows:

- If you specify the PARTITION statement, the tuner uses a single-partition validation set as defined in that statement. 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.

- If you specify the FRACTION= option, the tuner uses a single-partition validation set. 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.

- If you specify KFOLD=k, the tuner uses k-fold cross validation. In this process, the tuner partitions all the data into k-fold subsets (folds). For each fold, a new model is trained on each of the (k-1) folds and then validated using the selected (holdout) fold. The objective function value is averaged over each set of training and scoring executions to obtain a single error estimate value.

The optimization tuner algorithm is based on a genetic algorithm (GA), which applies the principles of natural selection and evolution to find an improved configuration. The tuner performs the following sequence of actions:

1. A default model configuration (default values of select model tuning parameters) 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. The initial set of configurations, also called a “population,” is generated using a technique called random Latin hypercube sampling (LHS). In a Latin hypercube sample, each configuration of hyperparameters is evaluated, and their objective function values are again recorded for comparison. This becomes Iteration 1.

3. The best model configurations from the initial population are used to generate the next population of model configurations, Iteration 2, which are then evaluated. This process is repeated for the remaining iterations, as long as the maximum number of evaluations or the maximum time is not reached.

4. The best model configuration is reevaluated by executing a single training and model 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 ODS table, as described in the section “ODS Table Names” on page 450.

You can tune the following hyperparameter values when you specify the AUTOTUNE 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

### 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 11.3 lists these names.

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Statement</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>BestConfiguration</td>
<td>Hyperparameters and objective function values for the best configuration</td>
<td>AUTOTUNE</td>
<td>Default</td>
</tr>
<tr>
<td>CostComplexity</td>
<td>Information about number of leaves and the error for cost-complexity pruning</td>
<td>PROC TREESPLIT</td>
<td>PRUNINGTABLE</td>
</tr>
<tr>
<td>ModellInfo</td>
<td>Information about the modeling environment</td>
<td>PROC TREESPLIT</td>
<td>Default</td>
</tr>
<tr>
<td>NObs</td>
<td>Observation information</td>
<td>PROC TREESPLIT</td>
<td>Default</td>
</tr>
<tr>
<td>ReducedError</td>
<td>Information about number of leaves and the error for reduced-error pruning</td>
<td>PROC TREESPLIT</td>
<td>PRUNINGTABLE</td>
</tr>
</tbody>
</table>
Table 11.3 continued

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Statement</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>TreePerformance</td>
<td>Fit statistics for the selected tree</td>
<td>PROC TREESPLIT</td>
<td>Default</td>
</tr>
<tr>
<td>TunerInfo</td>
<td>Setup values used by the tuner</td>
<td>AUTOTUNE</td>
<td>Default</td>
</tr>
<tr>
<td>TunerResults</td>
<td>Values of the hyperparameters and the objective function for the default configuration (Iteration 0) and up to 10 best configurations found</td>
<td>AUTOTUNE</td>
<td>Default</td>
</tr>
<tr>
<td>TunerSummary</td>
<td>Statistics about the tuning process</td>
<td>AUTOTUNE</td>
<td>Default</td>
</tr>
<tr>
<td>TunerTiming</td>
<td>Total time spent on different tasks while tuning</td>
<td>AUTOTUNE</td>
<td>Default</td>
</tr>
<tr>
<td>VariableImportance</td>
<td>Variable importance</td>
<td>PROC TREESPLIT</td>
<td>Default</td>
</tr>
</tbody>
</table>

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 11.4, along with the relevant PLOTS= options.

Table 11.4 Graphs Produced by PROC TREESPLIT

<table>
<thead>
<tr>
<th>ODS Graph Name</th>
<th>Plot Description</th>
<th>PLOTS=</th>
</tr>
</thead>
<tbody>
<tr>
<td>PruningPlot</td>
<td>Plot of the error sum of squares, misclassification rate, or cost complexity when it is used for final tree selection</td>
<td>PRUNEUNTIL</td>
</tr>
<tr>
<td>WholeTreePlot</td>
<td>Overview plot of final tree</td>
<td>WHOLETREE</td>
</tr>
<tr>
<td>ZoomedTreePlot</td>
<td>Detailed plot of portion of final tree</td>
<td>ZOOMEDTREE</td>
</tr>
</tbody>
</table>

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 <l option> ;
TARGET variable <l option> ;
```

**INPUT Statement**

```
INPUT variables <l 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 <l 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: TRESPLIT 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 8 and “Loading a SAS Data Set onto a CAS Server” on page 9 in Chapter 2, “Shared Concepts.”

Example 11.1: Creating a Binary Classification Tree with Validation Data

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 11.5 describes the variables in Hmeq.

Table 11.5 Variables in the Home Equity (Hmeq) Data Table

<table>
<thead>
<tr>
<th>Variable</th>
<th>Role</th>
<th>Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bad</td>
<td>Response</td>
<td>Binary</td>
<td>1 = applicant defaulted on the loan or is seriously delinquent 0 = applicant paid off the loan</td>
</tr>
<tr>
<td>CLAge</td>
<td>Predictor</td>
<td>Interval</td>
<td>Age of oldest credit line in months</td>
</tr>
<tr>
<td>CLNo</td>
<td>Predictor</td>
<td>Interval</td>
<td>Number of credit lines</td>
</tr>
<tr>
<td>DebtInc</td>
<td>Predictor</td>
<td>Interval</td>
<td>Debt-to-income ratio</td>
</tr>
<tr>
<td>Delinq</td>
<td>Predictor</td>
<td>Interval</td>
<td>Number of delinquent credit lines</td>
</tr>
<tr>
<td>Derog</td>
<td>Predictor</td>
<td>Interval</td>
<td>Number of major derogatory reports</td>
</tr>
<tr>
<td>Job</td>
<td>Predictor</td>
<td>Nominal</td>
<td>Occupational category</td>
</tr>
<tr>
<td>Loan</td>
<td>Predictor</td>
<td>Interval</td>
<td>Requested loan amount</td>
</tr>
<tr>
<td>MortDue</td>
<td>Predictor</td>
<td>Interval</td>
<td>Amount due on mortgage</td>
</tr>
<tr>
<td>nlnq</td>
<td>Predictor</td>
<td>Interval</td>
<td>Number of recent credit inquiries</td>
</tr>
<tr>
<td>Reason</td>
<td>Predictor</td>
<td>Binary</td>
<td>'DebtCon' = debt consolidation 'HomeImp' = home improvement</td>
</tr>
<tr>
<td>Value</td>
<td>Predictor</td>
<td>Interval</td>
<td>Value of property</td>
</tr>
<tr>
<td>YoJ</td>
<td>Predictor</td>
<td>Interval</td>
<td>Years at present job</td>
</tr>
</tbody>
</table>

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 mycas.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 mycas, but you can substitute any appropriately defined CAS engine libref.
Chapter 11: The TREESPLIT Procedure

/* Convert variable names to mixed case */
data mycas.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=mycas.hmeq(obs=10); run;

Output 11.1.1 shows the first 10 observations of mycas.hmeq.

Output 11.1.1 Partial Listing of the mycas.hmeq Data

<table>
<thead>
<tr>
<th>Obs</th>
<th>Bad</th>
<th>Loan</th>
<th>MortDue</th>
<th>Value</th>
<th>Reason</th>
<th>Job</th>
<th>YoJ</th>
<th>Derog</th>
<th>Delinq</th>
<th>CLAge</th>
<th>nInq</th>
<th>CLNo</th>
<th>DebtInc</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1100</td>
<td>25860</td>
<td>39025</td>
<td>HomeImp</td>
<td>Other</td>
<td>10.5</td>
<td>0</td>
<td>0</td>
<td>94.367</td>
<td>1</td>
<td>9</td>
<td>.</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1500</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.</td>
<td>.</td>
<td>.</td>
<td></td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1800</td>
<td>48649</td>
<td>57037</td>
<td>HomeImp</td>
<td>Other</td>
<td>5.0</td>
<td>3</td>
<td>2</td>
<td>77.100</td>
<td>1</td>
<td>17</td>
<td>.</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>2000</td>
<td>.</td>
<td>62250</td>
<td>HomeImp</td>
<td>Sales</td>
<td>16.0</td>
<td>0</td>
<td>0</td>
<td>115.800</td>
<td>0</td>
<td>13</td>
<td>.</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>2000</td>
<td>45000</td>
<td>55000</td>
<td>HomeImp</td>
<td>Other</td>
<td>3.0</td>
<td>0</td>
<td>0</td>
<td>86.067</td>
<td>2</td>
<td>25</td>
<td>.</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>2200</td>
<td>24280</td>
<td>34687</td>
<td>HomeImp</td>
<td>Other</td>
<td>.</td>
<td>0</td>
<td>1</td>
<td>300.867</td>
<td>0</td>
<td>8</td>
<td>.</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>2300</td>
<td>28192</td>
<td>40150</td>
<td>HomeImp</td>
<td>Other</td>
<td>4.5</td>
<td>0</td>
<td>0</td>
<td>54.600</td>
<td>1</td>
<td>16</td>
<td>.</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>2400</td>
<td>50000</td>
<td>73395</td>
<td>HomeImp</td>
<td>ProfEx</td>
<td>5.0</td>
<td>1</td>
<td>0</td>
<td>.</td>
<td>1</td>
<td>0</td>
<td>.</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>2400</td>
<td>.</td>
<td>17180</td>
<td>HomeImp</td>
<td>Other</td>
<td>.</td>
<td>0</td>
<td>0</td>
<td>14.567</td>
<td>3</td>
<td>4</td>
<td>.</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>2500</td>
<td>15000</td>
<td>20200</td>
<td>HomeImp</td>
<td></td>
<td>18.0</td>
<td>0</td>
<td>0</td>
<td>136.067</td>
<td>1</td>
<td>19</td>
<td>.</td>
</tr>
</tbody>
</table>

The following statements use the TREESPLIT procedure to create a classification tree:

ods graphics on;

proc treesplit data=mycas.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 equal 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 11.1.2 provides an overview of the full tree.

**Output 11.1.2** Overview Diagram of Final Tree

The observations in terminal nodes 9, C, D, F, and I are assigned a prediction of Bad=0, and those in terminal nodes 3, 8, A, G, and H are assigned a prediction of Bad=1. Node C contains the most observations, as indicated by the thickness of the link from its parent node.

The tree diagram in Output 11.1.3 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.
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. 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. Note that the legend shows what actual value of the response variable is represented by the value shown in the node. For example, in node 6, 93.39% of the 3,208 observations in the training data have an observed response value of Bad=0, as indicated by the value 2.
Example 11.1: Creating a Binary Classification Tree with Validation Data

Output 11.1.4 displays the fit statistics for the final tree.

**Output 11.1.4 Tree Performance**

The TREESPLIT Procedure

<table>
<thead>
<tr>
<th>Fit Statistics for Selected Tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Leaves</td>
</tr>
<tr>
<td>Training</td>
</tr>
<tr>
<td>Validation</td>
</tr>
</tbody>
</table>

Output 11.1.5 displays the pruning plot.

**Output 11.1.5 Pruning Plot**

This plot displays the misclassification rates for the training and validation data as the tree is pruned. The tree with 10 leaves is selected as the final tree because it 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 11.1.6 shows a partial listing of Scored.

Output 11.1.6  Partial Listing of the Scored Hmeq Data

<table>
<thead>
<tr>
<th>Obs</th>
<th>BAD</th>
<th>LOAN</th>
<th>MORTDUE</th>
<th>VALUE</th>
<th>REASON</th>
<th>JOB</th>
<th>YOJ</th>
<th>DEROG</th>
<th>DELIQ</th>
<th>CLAGE</th>
<th>NINQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1100</td>
<td>25860</td>
<td>39025</td>
<td>HomeImp Other</td>
<td>10.5</td>
<td>0</td>
<td>0</td>
<td>94.367</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1300</td>
<td>70053</td>
<td>68400</td>
<td>HomeImp Other</td>
<td>7.0</td>
<td>0</td>
<td>2</td>
<td>121.833</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1500</td>
<td>13500</td>
<td>16700</td>
<td>HomeImp Other</td>
<td>4.0</td>
<td>0</td>
<td>0</td>
<td>149.467</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1500</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1700</td>
<td>97800</td>
<td>112000</td>
<td>HomeImp Office</td>
<td>3.0</td>
<td>0</td>
<td>0</td>
<td>93.333</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1700</td>
<td>30548</td>
<td>40320</td>
<td>HomeImp Other</td>
<td>9.0</td>
<td>0</td>
<td>0</td>
<td>101.466</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1800</td>
<td>48649</td>
<td>57037</td>
<td>HomeImp Other</td>
<td>5.0</td>
<td>0</td>
<td>2</td>
<td>77.100</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>1800</td>
<td>28502</td>
<td>43034</td>
<td>HomeImp Other</td>
<td>11.0</td>
<td>0</td>
<td>0</td>
<td>88.766</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>2000</td>
<td>32700</td>
<td>46740</td>
<td>HomeImp Other</td>
<td>3.0</td>
<td>0</td>
<td>2</td>
<td>216.933</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>2000</td>
<td>.</td>
<td>62250</td>
<td>HomeImp Sales</td>
<td>16.0</td>
<td>0</td>
<td>0</td>
<td>115.800</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Obs</th>
<th>CLNO</th>
<th>DEBTINC</th>
<th>leaf_id</th>
<th>I_Bad</th>
<th>P_Bad1</th>
<th>P_Bad0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9</td>
<td>.</td>
<td>9 1</td>
<td>.</td>
<td>0.58906</td>
<td>0.41094</td>
</tr>
<tr>
<td>2</td>
<td>14</td>
<td>.</td>
<td>9 1</td>
<td>.</td>
<td>0.58906</td>
<td>0.41094</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>.</td>
<td>9 1</td>
<td>.</td>
<td>0.58906</td>
<td>0.41094</td>
</tr>
<tr>
<td>4</td>
<td>.</td>
<td>.</td>
<td>5 1</td>
<td>.</td>
<td>0.93421</td>
<td>0.06579</td>
</tr>
<tr>
<td>5</td>
<td>14</td>
<td>.</td>
<td>9 1</td>
<td>.</td>
<td>0.58906</td>
<td>0.41094</td>
</tr>
<tr>
<td>6</td>
<td>8</td>
<td>37.1136</td>
<td>10 0</td>
<td>.</td>
<td>0.06568</td>
<td>0.93432</td>
</tr>
<tr>
<td>7</td>
<td>17</td>
<td>.</td>
<td>9 1</td>
<td>.</td>
<td>0.58906</td>
<td>0.41094</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>36.8849</td>
<td>10 0</td>
<td>.</td>
<td>0.06568</td>
<td>0.93432</td>
</tr>
<tr>
<td>9</td>
<td>12</td>
<td>.</td>
<td>9 1</td>
<td>.</td>
<td>0.58906</td>
<td>0.41094</td>
</tr>
<tr>
<td>10</td>
<td>13</td>
<td>.</td>
<td>9 1</td>
<td>.</td>
<td>0.58906</td>
<td>0.41094</td>
</tr>
</tbody>
</table>

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 11.2: Creating a Regression Tree

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:

```plaintext
data mycas.baseball;
   set sashelp.baseball;
run;
```

The following statements create a regression tree model:

```plaintext
ods graphics on;
proc treesplit data=mycas.baseball maxdepth=3;
   class league division;
   model logSalary = nAtBat nHits nHome nRuns nRBI nBB
crMajor crAtBat crHits crHome crRuns crRbi
crBB league division nOuts nAssts nError;
   output out=mycas.treesplout;
run;
```

Because no GROW statement is specified, the tree is grown using the RSS criterion by default. Because no PRUNE statement is included, no pruning is performed. The OUTPUT statement requests generation of the data table mycas.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 11.2.1 displays the full 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 E has the lowest predicted response value, indicated by the lightest shade of blue, and node 7 has the highest, indicated by the dark shade.

Output 11.2.2 shows details of the first three levels of the tree, including the root node.
As in Output 11.1.3, 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 11.2.3 displays the fit statistic for the final regression tree (the only fit statistic provided for a regression tree is the ASE).

**Output 11.2.3** Regression Tree Performance

The **TREESPLIT Procedure**

<table>
<thead>
<tr>
<th>Fit Statistics for Selected Tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Leaves</td>
</tr>
<tr>
<td>-------------------</td>
</tr>
<tr>
<td>Training</td>
</tr>
</tbody>
</table>
Output 11.2.4 is a partial display of the mycas.treesplout data table that is created when you specify the OUTPUT statement.

**Output 11.2.4** Scored Predictor Data Table

<table>
<thead>
<tr>
<th>Obs</th>
<th>P_logSalary</th>
<th>DT_PredStd</th>
<th>LeafID</th>
<th>Residual</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4.6555457426</td>
<td>0.3064936468</td>
<td>14</td>
<td>.</td>
</tr>
<tr>
<td>2</td>
<td>6.8573201516</td>
<td>0.4410266912</td>
<td>7</td>
<td>0.6427120531</td>
</tr>
<tr>
<td>3</td>
<td>4.6555457426</td>
<td>0.3064936468</td>
<td>14</td>
<td>0.4070505006</td>
</tr>
<tr>
<td>4</td>
<td>6.8573201516</td>
<td>0.4410266912</td>
<td>7</td>
<td>-0.145745307</td>
</tr>
<tr>
<td>5</td>
<td>5.7477251685</td>
<td>0.3269101703</td>
<td>11</td>
<td>-0.56219311</td>
</tr>
<tr>
<td>6</td>
<td>4.6555457426</td>
<td>0.3064936468</td>
<td>14</td>
<td>.</td>
</tr>
<tr>
<td>7</td>
<td>6.8573201516</td>
<td>0.4410266912</td>
<td>7</td>
<td>.</td>
</tr>
<tr>
<td>8</td>
<td>4.6555457426</td>
<td>0.3064936468</td>
<td>14</td>
<td>-0.089386386</td>
</tr>
<tr>
<td>9</td>
<td>6.8573201516</td>
<td>0.4410266912</td>
<td>7</td>
<td>0.2023084645</td>
</tr>
<tr>
<td>10</td>
<td>6.35986721</td>
<td>0.5351772705</td>
<td>8</td>
<td>-0.260205997</td>
</tr>
</tbody>
</table>

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 11.3: Assessing Variable Importance**

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 mycas.MBE_Data, which contains measurements for 20 devices:

```plaintext
data mycas.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
```

The following statements create the tree model:

```plaintext
proc treesplit data=mycas.MBE_Data maxdepth=6;
   class Usable Dopant;
   model Usable = gTemp aTemp Rot Dopant;
   prune none;
run;
```

Output 11.3.1 shows the “Variable Importance” table.

```
Output 11.3.1  Variable Importance

The TREESPLIT Procedure

<table>
<thead>
<tr>
<th>Variable</th>
<th>Std Dev</th>
<th>Relative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dopant</td>
<td>2.7000</td>
<td>1.0000</td>
</tr>
<tr>
<td>aTemp</td>
<td>1.0500</td>
<td>0.3889</td>
</tr>
</tbody>
</table>
```

This table shows that the predictor gTemp has the largest importance value. This means that the growth temperature of the substrate is the most important consideration in determining the usability of the sample.

References


Part III

Utility
Chapter 12
The ASSESS Procedure

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 highly 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:

```plaintext
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:

```plaintext
cas mysess;
libname mycas cas sessref=mysess;
```

The CAS statement creates the CAS session named `mysess`, and the LIBNAME statement creates the `mycas` 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 8 in Chapter 2, “Shared Concepts.”

---

**Getting Started: ASSESS 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 8 and “Loading a SAS Data Set onto a CAS Server” on page 9 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 `mycas.score`, which is similar to score data from a linear regression, and loads the table into your CAS session:

```sas
data mycas.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 `mycas`, 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.

```sas
proc assess data=mycas.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 12.1* shows the minimum, maximum, and mean of the actual and predicted values for `_PartInd_=0.*
Figure 12.1  Lift Regression Information for the First Partition

The ASSESS Procedure

<table>
<thead>
<tr>
<th><em>PartInd</em> =0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lift Regression Information</td>
</tr>
<tr>
<td>Variable Depth</td>
</tr>
<tr>
<td>p_good 50</td>
</tr>
<tr>
<td>p_good 100</td>
</tr>
</tbody>
</table>

Figure 12.2 shows the error metrics of the interval target for _PartInd_ =0.

Figure 12.2  Fit Statistics for the First Partition

<table>
<thead>
<tr>
<th><em>PartInd</em> =0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fit Statistics</td>
</tr>
<tr>
<td>Squared Error</td>
</tr>
<tr>
<td>Number of Observations</td>
</tr>
<tr>
<td>5</td>
</tr>
</tbody>
</table>

Figure 12.3 shows the minimum, maximum, and mean of the actual and predicted values for _PartInd_ =1.

Figure 12.3  Lift Regression Information for the Second Partition

The ASSESS Procedure

<table>
<thead>
<tr>
<th><em>PartInd</em> =1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lift Regression Information</td>
</tr>
<tr>
<td>Variable Depth</td>
</tr>
<tr>
<td>p_good 50</td>
</tr>
<tr>
<td>p_good 100</td>
</tr>
</tbody>
</table>

Figure 12.4 shows the error metrics for the interval target for _PartInd_ =1.

Figure 12.4  Fit Statistics for the Second Partition

<table>
<thead>
<tr>
<th><em>PartInd</em> =1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fit Statistics</td>
</tr>
<tr>
<td>Squared Error</td>
</tr>
<tr>
<td>Number of Observations</td>
</tr>
<tr>
<td>5</td>
</tr>
</tbody>
</table>
Syntax: ASSESS Procedure

The following statements are available in the ASSESS procedure:

```
PROC ASSESS < options > ;
   INPUT variable ;
   TARGET variables < /options > ;
   FITSTAT PVAR=variables / PEVENT="event-list" < DLM="character"> ;
   FREQ variable ;
   BY 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=** `CAS-libref.data-table`
  names the input data table for PROC ASSESS to use. The default is the most recently created data table. `CAS-libref.data-table` is a two-level name, where

  - `CAS-libref` refers to a collection of information that is defined in the LIBNAME statement and includes the `caslib`, 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 `CAS-libref`, see the section “Using CAS Sessions and CAS Engine Librefs” on page 468.

  - `data-table` specifies the name of the input data table.

- **FITSTATOUT=** `CAS-libref.data-table`
  specifies the name of the fit statistics results table. `CAS-libref.data-table` is a two-level name, where `CAS-libref` refers to the `caslib` and session identifier, 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 468.

  If you specify this option, all ODS tables are suppressed. This option is valid only if you also include a FITSTAT statement.

- **LIFTOUT=** `CAS-libref.data-table`
  specifies the name of the lift results table. `CAS-libref.data-table` is a two-level name, where `CAS-libref` refers to the `caslib` and session identifier, 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 468.
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.

NBINS=integer

specifies the number of bins to be used 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 be used 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.

NTHREADS=number

specifies the number of threads that are used 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.

ROCOUT=CAS-libref.data-table

specifies the name of the ROC results table. CAS-libref.data-table is a two-level name, where CAS-libref refers to the caslib and session identifier, 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 468.

If you specify this option, all ODS tables are suppressed. This option is valid only for a nominal target (LEVEL=NOMINAL in the TARGET statement).

**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, the following metrics are created without specifying the FITSTAT statement: average square
You must specify the following arguments:

**PVAR=variables**

specifies the posterior probability for each level in model prediction except the variable specified 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 option. You must specify the PEVENT= and PVAR= options one-to-one in the same order.

You can also specify the following option:

**DELIMITER="character"**

specifies the delimiter that is used to separate events that are specified in the PEVENT= option. A delimiter is used when event names contain embedded spaces (for example, “Fuel Oil”) or special characters (for example, “;” or “*”). You must specify the 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 a delimiter other than “ ” is specified, PROC ASSESS treats any spaces (including empty spaces) to be 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 data set 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 the FREQ statement is not specified, 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:

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.

EVENT="string"
specifies the formatted value of 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.

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. 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 based on the event in each cutoff point. More terminology and the derivation of a confusion matrix is available in Fawcett (2006), Powers (2011), and Derby (2013). The Kolmogorov-Smirnov chart and other metrics are available in Derby (2013). The Youden Index can be found in Youden (1950). The F0.5 score is extended from F1 and is defined as follows:

\[
(1 + \beta^2) \frac{pr}{\beta^2 p + r}
\]

where \( p = \frac{tp}{tp + fp} \), \( r = \frac{tp}{tp + fn} \), \( \beta = 0.5 \), \( tp \) is true positive, \( fp \) is false positive, and \( fn \) is false negative. For more information about F0.5, see Kaggle Inc. (2015).

**Fit Statistics**

The following notation is necessary for discussion of fit statistics:

- \( n \) is the number of observations
- \( N \) is the sum of observation frequencies in the data
- \( w_i \) are the observation frequencies, so
  \[
  N = \sum_{i=1}^{n} w_i
  \]
- \( y_i \) are the actual values
- \( \hat{y}_i \) are the predicted values
- \( t \) is the level from the label
- \( \hat{t} \) is the predicted level

PROC ASSESS calculates the average square error, root average square error, and divisor of average square error for both interval targets and nominal targets.

For interval targets, PROC ASSESS also calculates the following metrics:

- mean square logarithmic error
  \[
  \text{MSLE} = \frac{1}{N} \sum_{i=1}^{n} \left( \log(\hat{y}_i + 1) - \log(y_i + 1) \right)^2 w_i
  \]
- mean absolute error
  \[
  \text{MAE} = \frac{1}{N} \sum_{i=1}^{n} |y_i - \hat{y}_i| w_i
  \]
For nominal targets, PROC ASSESS also calculates the following metrics:

- mean consequential error
  \[ \text{MCE} = \frac{1}{N} \sum_{t_i \neq \hat{t}_i} w_i \]

- multi-class log loss
  \[ \text{logloss} = -\frac{1}{N} \sum_{i=1}^{n} \sum_{j=1}^{m} y_{i,j} \log(p_{i,j}) w_i \]

where \( n \) is the sum of observation frequencies in the data, \( m \) is the total number of levels in the target, \( y_{i,j} \) is 1 if observation \( i \) is assigned to level \( j \), and \( p_{i,j} \) is the predicted probability that observation \( i \) is assigned to \( j \)

**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 12.1.

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Statement</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>FitStat</td>
<td>Fit statistics</td>
<td>FITSTAT</td>
<td>PVAR= and PEVENT= for a nominal target;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>default for interval target</td>
</tr>
<tr>
<td>LIFTInfo</td>
<td>Lift information</td>
<td>PROC</td>
<td>Default for nominal target</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ASSESS</td>
<td></td>
</tr>
<tr>
<td>LIFTRegInfo</td>
<td>Lift regression information</td>
<td>PROC</td>
<td>Default for interval target</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ASSESS</td>
<td></td>
</tr>
<tr>
<td>ROCInfo</td>
<td>ROC information</td>
<td>PROC</td>
<td>Default for nominal target</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ASSESS</td>
<td></td>
</tr>
</tbody>
</table>

Examples: ASSESS 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 8 and “Loading a SAS Data Set onto a CAS Server” on page 9 in Chapter 2, “Shared Concepts.”

Example 12.1: Assess a Model That Has a Binary Target

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.

```sas
data mycas.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:

```plaintext
proc assess data=mycas.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 12.1.1 shows the lift and gain for each bin in the first partition.
Example 12.1: Assess a Model That Has a Binary Target

**Output 12.1.1** Lift Regression Information for the First Partition

**The ASSESS Procedure**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Event</th>
<th>Depth</th>
<th>Number of Observations</th>
<th>Number of Events</th>
<th>Lift</th>
<th>Response Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_good</td>
<td>good</td>
<td>20</td>
<td>3</td>
<td>3</td>
<td>27.27</td>
<td>1.363636</td>
</tr>
<tr>
<td></td>
<td></td>
<td>40</td>
<td>3</td>
<td>2</td>
<td>18.18</td>
<td>0.909091</td>
</tr>
<tr>
<td></td>
<td></td>
<td>60</td>
<td>3</td>
<td>3</td>
<td>27.27</td>
<td>1.363636</td>
</tr>
<tr>
<td></td>
<td></td>
<td>80</td>
<td>3</td>
<td>3</td>
<td>27.27</td>
<td>1.363636</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100</td>
<td>3</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Output 12.1.2 shows the ROC information, including the confusion matrix and its derivations, for each bin in the first partition.

**Output 12.1.2** ROC Information for the First Partition

<table>
<thead>
<tr>
<th>Variable</th>
<th>Event</th>
<th>Cutoff</th>
<th>TP</th>
<th>FP</th>
<th>FN</th>
<th>TN</th>
<th>FPR</th>
<th>FDR</th>
<th>TPR</th>
<th>TNR</th>
<th>ACC</th>
<th>KS</th>
<th>Youden Index</th>
<th>F1 Score</th>
<th>F0.5 Score</th>
<th>AUC</th>
<th>Gini</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_good</td>
<td>good</td>
<td>0.20</td>
<td>11</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.266667</td>
<td>1</td>
<td>0.733333</td>
<td>0</td>
<td>0</td>
<td>0.846154</td>
<td>0.774648</td>
<td>0.806818</td>
<td>0.613636</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.40</td>
<td>11</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>0.250000</td>
<td>0.083333</td>
<td>1</td>
<td>0.750000</td>
<td>0.933333</td>
<td>1</td>
<td>0.956522</td>
<td>0.932203</td>
<td>0.806818</td>
<td>0.613636</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.60</td>
<td>5</td>
<td>1</td>
<td>6</td>
<td>3</td>
<td>0.250000</td>
<td>0.166667</td>
<td>0.454545</td>
<td>0.750000</td>
<td>0.533333</td>
<td>0</td>
<td>0.204545</td>
<td>0.588235</td>
<td>0.714286</td>
<td>0.806818</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.80</td>
<td>0</td>
<td>0</td>
<td>11</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.266667</td>
<td>0</td>
<td>0</td>
<td>0.806818</td>
<td>0.613636</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Output 12.1.3** shows the error metrics of the nominal target in the first partition.

**Output 12.1.3** Fit Statistics for the First Partition

<table>
<thead>
<tr>
<th>Squared Error</th>
<th>Number of Observations</th>
<th>Divisor of Average</th>
<th>Root Average</th>
<th>Mean Consequential Error</th>
<th>Multiclass Log Loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>15</td>
<td>15</td>
<td>0.338253</td>
<td>0.581595</td>
<td>0.266667</td>
</tr>
<tr>
<td>Multiclass</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.514728</td>
</tr>
</tbody>
</table>
Output 12.1.4 shows the lift and gain for each bin in the second partition.

**Output 12.1.4** Lift Regression Information for the Second Partition

<table>
<thead>
<tr>
<th>Variable</th>
<th>Event Depth</th>
<th>Number of Observations</th>
<th>Number of Events</th>
<th>Lift</th>
<th>Response Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_good</td>
<td>3</td>
<td>3</td>
<td>27.27</td>
<td>1.363636</td>
<td>100.00</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>3</td>
<td>27.27</td>
<td>1.363636</td>
<td>100.00</td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>3</td>
<td>27.27</td>
<td>1.363636</td>
<td>100.00</td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>3</td>
<td>0.00</td>
<td>81.82</td>
<td>1.022727</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>2</td>
<td>18.18</td>
<td>100.00</td>
<td>90.9091</td>
</tr>
</tbody>
</table>

Output 12.1.5 shows the ROC information, including the confusion matrix and its derivations, for each bin in the second partition.

**Output 12.1.5** ROC Information for the Second Partition

<table>
<thead>
<tr>
<th>Variable</th>
<th>Event</th>
<th>Cutoff</th>
<th>TP</th>
<th>FP</th>
<th>FN</th>
<th>TPR</th>
<th>FDR</th>
<th>TNR</th>
<th>ACC</th>
<th>KS</th>
<th>Youden Index</th>
<th>F1 Score</th>
<th>F0.5 Score</th>
<th>AUC</th>
<th>Gini</th>
<th>Gamma</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_good</td>
<td>good</td>
<td>0.200000</td>
<td>10</td>
<td>1</td>
<td>0</td>
<td>0.230769</td>
<td>0.714286</td>
<td>0.642857</td>
<td>0.6909091</td>
<td>0.571429</td>
<td>0.454545</td>
<td>0.672727</td>
<td>0.755610</td>
<td>0.272727</td>
<td>0.428571</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.400000</td>
<td>9</td>
<td>2</td>
<td>0</td>
<td>0.250000</td>
<td>0.818182</td>
<td>0.642857</td>
<td>0.909091</td>
<td>0.755610</td>
<td>0.272727</td>
<td>0.428571</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.600000</td>
<td>5</td>
<td>6</td>
<td>3</td>
<td>0.454545</td>
<td>0.571429</td>
<td>0.642857</td>
<td>0.818182</td>
<td>0.755610</td>
<td>0.272727</td>
<td>0.428571</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.800000</td>
<td>3</td>
<td>8</td>
<td>3</td>
<td>0.272727</td>
<td>0.428571</td>
<td>0.642857</td>
<td>0.755610</td>
<td>0.272727</td>
<td>0.428571</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Output 12.1.6 shows the error metrics of the nominal target in the second partition.
Output 12.1.6  Fit Statistics for the Second Partition

<table>
<thead>
<tr>
<th>Number of Observations</th>
<th>Divisor Average</th>
<th>Root Average</th>
<th>Mean Error</th>
<th>Multiclass Log Loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>14</td>
<td>0.443175</td>
<td>0.665714</td>
<td>0.142857</td>
</tr>
</tbody>
</table>

References


Chapter 13
The BINNING Procedure

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 interval 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 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.

**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.

**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. For more information, see the section “Binning Computation and Formulas” on page 491.

**Cutpoint Binning**

The cutpoint binning method enables you to create bins by specifying the bin upper bound.

**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
- 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:

```sas
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:

```sas
cas mysess;
libname mycas cas sessref=mysess;
```

The CAS statement creates the CAS session named `mysess`, and the LIBNAME statement creates the `mycas` 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:

```sas
cas mysess terminate;
```

For more information about the CAS and LIBNAME statements, see the section “Introduction to Shared Concepts” on page 8 in Chapter 2, “Shared Concepts.”

Getting Started: BINNING Procedure

Bucket Binning

**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 8 and “Loading a SAS Data Set onto a CAS Server” on page 9 in Chapter 2, “Shared Concepts.”
The following DATA step creates the data table `mycas.bucket`, which consists of 1,000 observations of an ID variable (`id`) and three continuous variables (`x1`–`x3`), in your CAS session:

```sas
data mycas.bucket;
  length id 8;
  do id=1 to 1000;
    x1 = ranuni(101);
    x2 = 10*ranuni(201);
    x3 = 100*ranuni(301);
    output;
  end;
run;
```

These statements assume that your CAS engine libref is named `mycas`, 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:

```sas
proc binning data=mycas.bucket numbin=10 method=bucket;
  input x1-x3;
  output out=mycas.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 13.1 displays the “Binning 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).
### Bin Details

<table>
<thead>
<tr>
<th>Variable</th>
<th>Bin ID</th>
<th>Lower Bound</th>
<th>Upper Bound</th>
<th>Bin Width</th>
<th>Number of Observations</th>
<th>Mean</th>
<th>Standard Deviation</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>Missing</td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>-Infy</td>
<td>0.0999</td>
<td></td>
<td>108</td>
<td>0.0472</td>
<td>0.0289</td>
<td>0.0005</td>
<td>0.0995</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.0999</td>
<td>0.1994</td>
<td>0.0995</td>
<td>94</td>
<td>0.1523</td>
<td>0.0313</td>
<td>0.1002</td>
<td>0.1993</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.1994</td>
<td>0.2989</td>
<td>0.0995</td>
<td>90</td>
<td>0.2477</td>
<td>0.0302</td>
<td>0.2003</td>
<td>0.2986</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0.2989</td>
<td>0.3983</td>
<td>0.0995</td>
<td>95</td>
<td>0.3455</td>
<td>0.0284</td>
<td>0.2994</td>
<td>0.3965</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>0.3983</td>
<td>0.4978</td>
<td>0.0995</td>
<td>90</td>
<td>0.4508</td>
<td>0.0282</td>
<td>0.3984</td>
<td>0.4977</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>0.4978</td>
<td>0.5973</td>
<td>0.0995</td>
<td>115</td>
<td>0.5506</td>
<td>0.0311</td>
<td>0.4998</td>
<td>0.5969</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>0.5973</td>
<td>0.6968</td>
<td>0.0995</td>
<td>116</td>
<td>0.6461</td>
<td>0.0282</td>
<td>0.5980</td>
<td>0.6951</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>0.6968</td>
<td>0.7962</td>
<td>0.0995</td>
<td>105</td>
<td>0.7509</td>
<td>0.0290</td>
<td>0.6983</td>
<td>0.7950</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>0.7962</td>
<td>0.8957</td>
<td>0.0995</td>
<td>106</td>
<td>0.8478</td>
<td>0.0260</td>
<td>0.8003</td>
<td>0.8940</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.8957</td>
<td>Infy</td>
<td></td>
<td>81</td>
<td>0.9421</td>
<td>0.0262</td>
<td>0.8978</td>
<td>0.9952</td>
</tr>
<tr>
<td>x2</td>
<td>Missing</td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>-Infy</td>
<td>1.0215</td>
<td></td>
<td>99</td>
<td>0.4865</td>
<td>0.2964</td>
<td>0.0277</td>
<td>1.0152</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1.0215</td>
<td>2.0154</td>
<td>0.9939</td>
<td>110</td>
<td>1.5061</td>
<td>0.2823</td>
<td>1.0232</td>
<td>1.9713</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2.0154</td>
<td>3.0093</td>
<td>0.9939</td>
<td>99</td>
<td>2.4672</td>
<td>0.2840</td>
<td>2.0172</td>
<td>3.0041</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>3.0093</td>
<td>4.0031</td>
<td>0.9939</td>
<td>85</td>
<td>3.4925</td>
<td>0.3028</td>
<td>3.0129</td>
<td>4.0023</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>4.0031</td>
<td>4.9970</td>
<td>0.9939</td>
<td>109</td>
<td>4.4883</td>
<td>0.2898</td>
<td>4.0033</td>
<td>4.9736</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>4.9970</td>
<td>5.9909</td>
<td>0.9939</td>
<td>98</td>
<td>5.4785</td>
<td>0.2910</td>
<td>5.0073</td>
<td>5.9874</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>5.9909</td>
<td>6.9847</td>
<td>0.9939</td>
<td>100</td>
<td>6.5686</td>
<td>0.2797</td>
<td>6.0550</td>
<td>6.9823</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>6.9847</td>
<td>7.9786</td>
<td>0.9939</td>
<td>94</td>
<td>7.4833</td>
<td>0.2893</td>
<td>6.9975</td>
<td>7.9784</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>7.9786</td>
<td>8.9725</td>
<td>0.9939</td>
<td>112</td>
<td>8.4685</td>
<td>0.2947</td>
<td>7.9802</td>
<td>8.9567</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>8.9725</td>
<td>Infy</td>
<td></td>
<td>94</td>
<td>9.4733</td>
<td>0.2697</td>
<td>8.9760</td>
<td>9.9663</td>
</tr>
<tr>
<td>x3</td>
<td>Missing</td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>-Infy</td>
<td>10.174</td>
<td></td>
<td>97</td>
<td>5.2253</td>
<td>2.5781</td>
<td>0.2042</td>
<td>10.051</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>30.112</td>
<td>40.082</td>
<td>9.9694</td>
<td>94</td>
<td>35.979</td>
<td>2.7377</td>
<td>30.274</td>
<td>40.076</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>40.082</td>
<td>50.051</td>
<td>9.9694</td>
<td>102</td>
<td>44.474</td>
<td>2.9092</td>
<td>40.153</td>
<td>49.758</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>50.051</td>
<td>60.021</td>
<td>9.9694</td>
<td>99</td>
<td>55.331</td>
<td>2.6685</td>
<td>50.111</td>
<td>59.963</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>60.021</td>
<td>69.990</td>
<td>9.9694</td>
<td>107</td>
<td>64.927</td>
<td>2.7211</td>
<td>60.307</td>
<td>69.680</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>69.990</td>
<td>79.959</td>
<td>9.9694</td>
<td>92</td>
<td>74.900</td>
<td>2.9279</td>
<td>69.996</td>
<td>79.880</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>79.959</td>
<td>89.929</td>
<td>9.9694</td>
<td>91</td>
<td>84.836</td>
<td>2.9643</td>
<td>79.975</td>
<td>89.848</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>89.929</td>
<td>Infy</td>
<td></td>
<td>115</td>
<td>95.022</td>
<td>3.0062</td>
<td>90.078</td>
<td>99.898</td>
</tr>
</tbody>
</table>
Chapter 13: The BINNING Procedure

Syntax: BINNING Procedure

The following statements are available in the BINNING procedure:

```
PROC BINNING <options> ;
  CODE FILE=filename ;
  FREQ variable ;
  INPUT variables </option> ;
  OUTPUT OUT=CAS-libref.data-table <option> ;
  TARGET variable / EVENT="category" ;
```

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).

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 13.1 summarizes important options in the PROC BINNING statement by function.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Basic Options</strong></td>
<td></td>
</tr>
<tr>
<td>DATA=</td>
<td>Specifies the input data table</td>
</tr>
<tr>
<td><strong>Binning Level Options</strong></td>
<td></td>
</tr>
<tr>
<td>NUMBIN=</td>
<td>Specifies the global number of bins for all binning variables</td>
</tr>
<tr>
<td><strong>Binning Method Options</strong></td>
<td></td>
</tr>
<tr>
<td>METHOD=</td>
<td>Specifies which binning method to use</td>
</tr>
<tr>
<td><strong>Weight-of-Evidence Options</strong></td>
<td></td>
</tr>
<tr>
<td>WOE(WOEAJUST=)</td>
<td>Computes the weight of evidence and information values</td>
</tr>
</tbody>
</table>
You can specify the following `options`:

**DATA=** `CAS-libref.data-table`

names the input data table for PROC BINNING to use. The default is the most recently created data table. `CAS-libref.data-table` is a two-level name, where

- `CAS-libref` refers to a collection of information that is defined in the LIBNAME statement and includes the `caslib`, 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 `CAS-libref`, see the section “Using CAS Sessions and CAS Engine Librefs” on page 485.
- `data-table` specifies the name of the input data table.

**METHOD=** `BUCKET | QUANTILE | WINSOR(RATE=) | CUTPTS(numlist)`

specifies which binning method to use.

You can specify one of the following methods:

- **BUCKET**
  - specifies the bucket binning method.

- **QUANTILE**
  - specifies the quantile binning method.

- **WINSOR(RATE=rate)**
  - specifies the Winsorized binning method and the `rate` that it uses. You must specify a `rate` between 0 and 0.5, exclusive.

- **CUTPTS(numlist)**
  - specifies the cutpoints binning method and the lower and upper bounds of the bin.

**NUMBIN=** `integer`

specifies the global number of binning levels for all binning variables. The value of `integer` can be any integer between 2 and 1,000, inclusive.

By default, `NUMBIN=16`.

**WOE(WOEADJUST=number)**

enables computation of the weight of evidence and information values. 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 data set 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 the FREQ statement is not specified, each observation is assigned a frequency of 1.

**INPUT Statement**

INPUT variables < / option > ;

The INPUT statement names one or more variables as input variables for binning. The specified variables must be interval variables. If classification variables are provided, PROC BINNING stops and returns an error message. 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.

You can specify the following option in each INPUT statement:

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).

**OUTPUT Statement**

OUTPUT OUT=CAS-libref.data-table < option > ;

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. 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.”

**NOTE:** If an input variable value is missing, then the binning output level value is 0.

You must specify the following option:

```plaintext
OUT=\text{CAS-libref.data-table}
```

names the output data table for PROC BINNING to use. You must specify this option before any other options. \text{CAS-libref.data-table} is a two-level name, where

- \text{CAS-libref} refers to a collection of information that is defined in the LIBNAME statement and includes the \text{caslib}, 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 \text{CAS-libref}, see the section “Using CAS Sessions and CAS Engine Librefs” on page 485.

- \text{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:

```plaintext
COPYVAR=\text{variable}
COPYVARS=(\text{variables})
```

lists one or more \text{variables} from the input data table to be transferred to the output data table.

---

**TARGET Statement**

```plaintext
TARGET \text{variable} / EVENT="category";
```

The TARGET statement names the \text{variable} that PROC BINNING uses to calculate the weight of evidence and information value.

You must specify the following option after a slash (/):

```plaintext
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.

---

**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 \( \text{numbin} \) is the value of the \textsc{numbin=} option in the \textsc{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 \( k = 1, 2, \ldots, \text{numbin} - 1 \).
  When the data are evenly distributed on the cloud, the time complexity for bucket binning is \( O(n/\text{nodes} \times \text{nThreads}) \), where \( n \) is the number of observations, \( \text{nodes} \) is the number of computer nodes in the cloud, and \( \text{nThreads} \) is the number of CPUs on each node.

- For quantile binning, \textsc{proc binning} calculates a quantile table \( P \). Let \( P = \{p_k\} \), where \( k = 1, 2, \ldots, \text{numbin} \). Then \( p_k \) is described as
  \[
  p_k = \begin{cases} 
  1.0/\text{numbin} + p_{k-1} & \text{if } 0 < k < \text{numbin} \\
  1.0 & \text{if } k = \text{numbin}
  \end{cases}
  \]
  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 \textsc{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, \textsc{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 \textsc{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 \text{rate=} suboption in the \textsc{method=winsor} option in the \textsc{proc binning} statement) by the lower and upper thresholds. For variable \( x \), \textsc{proc binning} uses a simple bucket sorting method to obtain the basic information. Let \( N \) be the number of buckets. For each bucket \( B_i \), \( i = 1, 2, \ldots, N \), \textsc{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 \( B_i \)
\[- \sum x_i^2: \text{sum of } x^2 \text{ in } B_i\]

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 \( \text{ceil}(\text{Rate} \times n) \), and find the smallest \( I \) such that \( \sum_{i=1}^{I} c_i \geq wc \). Then the left tail count is \( \text{lwc} = \sum_{i=1}^{I} c_i \). Find the next \( I_l \) such that \( \sum_{i=1}^{I_l} c_i > \text{lwc} \). Therefore, the minimum value is \( \text{WinsorMin} = \text{min} I_l \). Similarly, find the largest \( I \) such that \( \sum_{i=1}^{N} c_i \geq wc \). The right tail count is \( \text{rwc} = \sum_{i=1}^{N} c_i \). Find the next \( I_r \) such that \( \sum_{i=1}^{I_r} c_i > \text{rwc} \). Then the maximum value is \( \text{WinsorMax} = \text{max} I_r \). The mean is calculated by the formula

\[
\text{WinsorMean} = \frac{\text{lwc} \times \text{WinsorMin} + \sum_{i=I_l}^{I} x_i + \text{rwc} \times \text{WinsorMax}}{n}
\]

The trimmed mean is calculated by the formula

\[
\text{TrimmedMean} = \frac{\sum_{i=I_l}^{I_r} x_i}{n - \text{lwc} - \text{rwc}}
\]

**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.

---

**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:

\[
\text{WOE}_{\text{attribute}} = \ln \frac{P_{\text{attr}(\text{nonevent})}}{P_{\text{attr}(\text{event})}} = \ln \frac{N_{\text{attr}(\text{nonevent})}/N_{\text{tot}(\text{nonevent})}}{N_{\text{attr}(\text{event})}/N_{\text{tot}(\text{event})}}
\]

The definitions of the quantities in the preceding formula are as follows:

- \( N_{\text{attr}(\text{nonevent})} \): the number of nonevent records that exhibit the attribute
- \( N_{\text{tot}(\text{nonevent})} \): the total number of nonevent records
- \( N_{\text{attr}(\text{event})} \): the number of event records that exhibit the attribute
- \( N_{\text{tot}(\text{event})} \): the total number of event records

To avoid an undefined WOE, an adjustment factor, \( x \), is used:

\[
\text{WOE}_{\text{attribute}} = \ln \frac{(N_{\text{attr}(\text{nonevent})} + x)/N_{\text{tot}(\text{nonevent})}}{(N_{\text{attr}(\text{event})} + x)/N_{\text{tot}(\text{event})}}
\]
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:

\[
IV = \sum_{i=1}^{m} \left( \frac{N_{\text{attr}(\text{nonevent})}}{N_{\text{tot}(\text{nonevent})}} - \frac{N_{\text{attr}(\text{event})}}{N_{\text{tot}(\text{event})}} \right) \times \text{WOE}_i
\]

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, 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.

### ODS Table Names

Each table created by the BINNING 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 13.2.
Table 13.2  ODS Tables Produced by PROC BINNING

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Statement</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>BinDetails</td>
<td>Basic binning information and parameters</td>
<td>PROC BINNING</td>
<td>Default</td>
</tr>
<tr>
<td>InfoValue</td>
<td>Information value for each variable</td>
<td>PROC BINNING</td>
<td>WOE</td>
</tr>
<tr>
<td>Trim</td>
<td>Trimmed statistics for the given variables</td>
<td>PROC BINNING</td>
<td>METHOD=WINSOR</td>
</tr>
<tr>
<td>Winsor</td>
<td>Winsor statistics for the given variables</td>
<td>PROC BINNING</td>
<td>METHOD=WINSOR</td>
</tr>
</tbody>
</table>

Examples: BINNING 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 8 and “Loading a SAS Data Set onto a CAS Server” on page 9 in Chapter 2, “Shared Concepts.”

Example 13.1: Quantile Binning

The following DATA step generates 1,000,000 observations of an ID variable (id) and two continuous variables (x1–x2). The mycas CAS library reference enables your client machine to communicate with the CAS session.

```sas
data mycas.ex1;
  length id 8;
  do id=1 to 1000000;
    x1 = ranuni(101);
    x2 = 10*ranuni(201);
    output;
  end;
run;
```

The following statements demonstrate how to use PROC BINNING to perform the quantile binning:

```sas
proc binning data=mycas.ex1 numbin=10 method=quantile;
  input x1-x2;
  output out=mycas.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 “Binning Details” table in Output 13.1 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 13.1.1 Binning Details

<table>
<thead>
<tr>
<th>Variable</th>
<th>Bin ID</th>
<th>Lower Bound</th>
<th>Upper Bound</th>
<th>Bin Width</th>
<th>Number of Observations</th>
<th>Mean</th>
<th>Standard Deviation</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>Missing</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>x2</td>
<td>Missing</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

The BINNING Procedure
Example 13.2: Winsorized Binning

The following DATA step generates 10,000 observations of an ID variable (id) and three continuous variables (x1–x3). The mycas CAS library reference enables your client machine to communicate with the CAS session.

```sas
data mycas.ex2;
  length id 8;
  do id=1 to 10000;
    x1 = ranuni(101);
    x2 = 10*ranuni(201);
    x3 = 100*ranuni(301);
    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:

```sas
proc binning data=mycas.ex2 numbin=10 method=winsor(rate=0.05);
  input x1-x3;
  output out=mycas.out2;
run;
```

The preceding statements generate Output 13.2.1 through Output 13.2.3.
### Output 13.2.1 Binning Details

#### The BINNING Procedure

<table>
<thead>
<tr>
<th>Variable</th>
<th>Bin ID</th>
<th>Lower Bound</th>
<th>Upper Bound</th>
<th>Bin Width</th>
<th>Number of Observations</th>
<th>Mean</th>
<th>Standard Deviation</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>Missing</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>-Infy</td>
<td>0.1377</td>
<td></td>
<td></td>
<td>1405</td>
<td>0.0767</td>
<td>0.0304</td>
<td>0.0475</td>
<td>0.1376</td>
</tr>
<tr>
<td>2</td>
<td>0.1377</td>
<td>0.2278</td>
<td>0.0902</td>
<td></td>
<td>849</td>
<td>0.1832</td>
<td>0.0259</td>
<td>0.1380</td>
<td>0.2276</td>
</tr>
<tr>
<td>3</td>
<td>0.2278</td>
<td>0.3180</td>
<td>0.0902</td>
<td></td>
<td>897</td>
<td>0.2726</td>
<td>0.0263</td>
<td>0.2279</td>
<td>0.3180</td>
</tr>
<tr>
<td>4</td>
<td>0.3180</td>
<td>0.4082</td>
<td>0.0902</td>
<td></td>
<td>864</td>
<td>0.3625</td>
<td>0.0258</td>
<td>0.3181</td>
<td>0.4081</td>
</tr>
<tr>
<td>5</td>
<td>0.4082</td>
<td>0.4983</td>
<td>0.0902</td>
<td></td>
<td>906</td>
<td>0.4519</td>
<td>0.0258</td>
<td>0.4082</td>
<td>0.4982</td>
</tr>
<tr>
<td>6</td>
<td>0.4983</td>
<td>0.5885</td>
<td>0.0902</td>
<td></td>
<td>899</td>
<td>0.5430</td>
<td>0.0269</td>
<td>0.4984</td>
<td>0.5884</td>
</tr>
<tr>
<td>7</td>
<td>0.5885</td>
<td>0.6786</td>
<td>0.0902</td>
<td></td>
<td>936</td>
<td>0.6341</td>
<td>0.0268</td>
<td>0.5886</td>
<td>0.6786</td>
</tr>
<tr>
<td>8</td>
<td>0.6786</td>
<td>0.7688</td>
<td>0.0902</td>
<td></td>
<td>900</td>
<td>0.7248</td>
<td>0.0258</td>
<td>0.6786</td>
<td>0.7686</td>
</tr>
<tr>
<td>9</td>
<td>0.7688</td>
<td>0.8589</td>
<td>0.0902</td>
<td></td>
<td>950</td>
<td>0.8124</td>
<td>0.0263</td>
<td>0.7690</td>
<td>0.8589</td>
</tr>
<tr>
<td>10</td>
<td>0.8589</td>
<td>Infy</td>
<td></td>
<td></td>
<td>1394</td>
<td>0.9192</td>
<td>0.0307</td>
<td>0.8589</td>
<td>0.9491</td>
</tr>
</tbody>
</table>

| x2       | Missing |             | 0           |           |                        |      |                   |         |         |
| 1        | -Infy  | 1.3978      |             |           | 1385                   | 0.7827 | 0.2989            | 0.4948  | 1.3958  |
| 2        | 1.3978 | 2.3008      | 0.9030      |           | 941                    | 1.8522 | 0.2597            | 1.3997  | 2.2997  |
| 3        | 2.3008 | 3.2038      | 0.9030      |           | 888                    | 2.7546 | 0.2623            | 2.3027  | 3.2037  |
| 4        | 3.2038 | 4.1069      | 0.9030      |           | 855                    | 3.6590 | 0.2597            | 3.2060  | 4.1065  |
| 5        | 4.1069 | 5.0099      | 0.9030      |           | 905                    | 4.5591 | 0.2567            | 4.1072  | 5.0091  |
| 6        | 5.0099 | 5.9129      | 0.9030      |           | 864                    | 5.4573 | 0.2571            | 5.0105  | 5.9106  |
| 7        | 5.9129 | 6.8159      | 0.9030      |           | 908                    | 6.3697 | 0.2596            | 5.9137  | 6.8156  |
| 8        | 6.8159 | 7.7189      | 0.9030      |           | 909                    | 7.2777 | 0.2550            | 6.8161  | 7.7163  |
| 9        | 7.7189 | 8.6220      | 0.9030      |           | 935                    | 8.1767 | 0.2593            | 7.7201  | 8.6218  |
| 10       | 8.6220 | Infy        |             |           | 1410                   | 9.2308 | 0.3049            | 8.6222  | 9.5250  |

| x3       | Missing |             | 0           |           |                        |      |                   |         |         |
| 1        | -Infy  | 14.323      |             |           | 1443                   | 8.2442 | 2.9877            | 5.3640  | 14.310  |
| 3        | 23.282 | 32.241      | 8.9589      |           | 902                    | 27.757 | 2.6725            | 23.305  | 32.227  |
| 4        | 32.241 | 41.200      | 8.9589      |           | 898                    | 36.908 | 2.5926            | 32.242  | 41.184  |
| 5        | 41.200 | 50.158      | 8.9589      |           | 906                    | 45.582 | 2.5960            | 41.246  | 50.135  |
| 6        | 50.158 | 59.117      | 8.9589      |           | 901                    | 54.656 | 2.5158            | 50.165  | 59.108  |
| 7        | 59.117 | 68.076      | 8.9589      |           | 898                    | 63.534 | 2.5844            | 59.121  | 68.065  |
| 8        | 68.076 | 77.035      | 8.9589      |           | 877                    | 72.557 | 2.5229            | 68.090  | 77.026  |
| 9        | 77.035 | 85.994      | 8.9589      |           | 870                    | 81.437 | 2.5739            | 77.039  | 85.980  |
| 10       | 85.994 | Infy        |             |           | 1398                   | 92.050 | 3.0000            | 86.001  | 94.953  |

### Output 13.2.2 Winsorized Statistics

#### Winsorized Statistics

<table>
<thead>
<tr>
<th>Variable</th>
<th>Minimum</th>
<th>Maximum</th>
<th>Mean</th>
<th>Standard Deviation</th>
<th>N Percent Left Tail</th>
<th>N Percent Right Tail</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>0.0475</td>
<td>0.9491</td>
<td>0.5018</td>
<td>0.2848</td>
<td>499</td>
<td>500</td>
</tr>
<tr>
<td>x2</td>
<td>0.4948</td>
<td>9.5250</td>
<td>5.0302</td>
<td>2.8663</td>
<td>499</td>
<td>500</td>
</tr>
<tr>
<td>x3</td>
<td>5.3640</td>
<td>94.953</td>
<td>49.793</td>
<td>28.373</td>
<td>499</td>
<td>500</td>
</tr>
</tbody>
</table>
Example 13.3: Bucket Binning and Weight-of-Evidence Computation

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 mycas CAS library reference enables your client machine to communicate with the CAS session.

data mycas.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=mycas.ex3 numbin=5 woe;
      input x1/numbin=4;
      input x2;
      target y/event="y";
      output out=mycas.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 13.3.1 through Output 13.3.2.
Chapter 13: The BINNING Procedure

Output 13.3.1 Binning Details

The BINNING Procedure

<table>
<thead>
<tr>
<th>Variable</th>
<th>Bin ID</th>
<th>Lower Bound</th>
<th>Upper Bound</th>
<th>Bin Width</th>
<th>Number of Observations</th>
<th>Mean</th>
<th>Standard Deviation</th>
<th>Minimum</th>
<th>Maximum</th>
<th>Event Count</th>
<th>Weight of Evidence</th>
<th>Information Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>Missing</td>
<td>-Inf</td>
<td>1.5000</td>
<td></td>
<td>2</td>
<td>0</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.8109</td>
<td>0.2027</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>-Inf</td>
<td>1.5000</td>
<td></td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.8109</td>
<td>0.2027</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1.5000</td>
<td>3 1.5000</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0.8109</td>
<td>0.2027</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>3 4.5000</td>
<td>1.5000</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>0.5774</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>-0.981</td>
<td>0.4087</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>4.5000</td>
<td>Infy</td>
<td></td>
<td>3</td>
<td>3</td>
<td>0.5774</td>
<td>5</td>
<td>6</td>
<td>1</td>
<td>0.4055</td>
<td>0.0676</td>
</tr>
<tr>
<td>x2</td>
<td>Missing</td>
<td>-Inf</td>
<td>2.2000</td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.9163</td>
<td>0.3054</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>-Inf</td>
<td>2.2000</td>
<td></td>
<td>2</td>
<td>2</td>
<td>0.7071</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0.9163</td>
<td>0.3054</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2.2000</td>
<td>3.4000</td>
<td>1.2000</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.9163</td>
<td>0.3054</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>3.4000</td>
<td>4.6000</td>
<td>1.2000</td>
<td>3</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>0.9163</td>
<td>0.3054</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>4.6000</td>
<td>5.8000</td>
<td>1.2000</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.9163</td>
<td>0.3054</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>5.8000</td>
<td>Infy</td>
<td></td>
<td>5</td>
<td>5</td>
<td>0.5477</td>
<td>6</td>
<td>7</td>
<td>2</td>
<td>-0.288</td>
<td>0.0479</td>
</tr>
</tbody>
</table>

Output 13.3.2 Variable Information Value

<table>
<thead>
<tr>
<th>Variable</th>
<th>Information Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>0.8817</td>
</tr>
<tr>
<td>x2</td>
<td>0.3534</td>
</tr>
</tbody>
</table>

Example 13.4: Cutpoint Binning

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 mycas.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=mycas.ex4 numbin=4 method=cutpts(2, 2.3, 4.5, 3.1, 5);
  input x2;
  input x1/numbin=3;
run;
```

The DATA= option specifies the input data table. The first INPUT statement names one continuous variable \( x_2 \) as the first input variable for binning with 4 bins specified by the NUMBIN= global option. The second INPUT statement names another continuous variable \( x_1 \) as the second input variable for binning with 3 bins. The METHOD= option specifies that cutpoint binning method will be used. For the first input variable \( x_2 \), 2, 2.3, 4.5 (and infinity) will be used as the upper bounds for its 4 bins. For the second input variable \( x_1 \), 3.1, 5 (and infinity) will be used as the upper bounds for its 3 bins.

Output 13.4.1 shows the “Binning Details” table.

### Output 13.4.1 Binning Details

<table>
<thead>
<tr>
<th></th>
<th>Bin Details</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Variable</td>
</tr>
<tr>
<td>x2</td>
<td>Missing</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>4</td>
</tr>
<tr>
<td>x1</td>
<td>Missing</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>3</td>
</tr>
</tbody>
</table>
Chapter 14
The CARDINALITY Procedure

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 mycas cas sessref=mysess;

The CAS statement creates the CAS session named mysess, and the LIBNAME statement creates the mycas 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 8 in Chapter 2, “Shared Concepts.”

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 8 and “Loading a SAS Data Set onto a CAS Server” on page 9 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 mycas.iris;
   set sashelp.iris;
   run;

These statements assume that your CAS engine libref is named mycas, but you can substitute any appropriately defined CAS engine libref.

The following statements show the basic usage:

   proc cardinality data=mycas.iris outcard=mycas.card
      outdetails=mycas.details maxlevels=10;
   run;

The DATA= option names mycas.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 mycas.card data table, and the OUTDETAILS= option requests that the levels found for each variable be stored in the mycas.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 14.1. 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.

```plaintext
proc print data=mycas.card;
  var _varname_ _type_ _cardinality_ _more_;
run;
```

**Figure 14.1** Cardinality Output Data Table

<table>
<thead>
<tr>
<th>Obs</th>
<th><em>VARNAME</em></th>
<th><em>TYPE</em></th>
<th><em>CARDINALITY</em></th>
<th><em>MORE</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Species</td>
<td>C</td>
<td>3 N</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>SepalLength</td>
<td>N</td>
<td>10 Y</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>SepalWidth</td>
<td>N</td>
<td>10 Y</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>PetalLength</td>
<td>N</td>
<td>10 Y</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>PetalWidth</td>
<td>N</td>
<td>10 Y</td>
<td></td>
</tr>
</tbody>
</table>

Table 14.1 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 14.1** Variables from the Cardinality Data

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>VARNAME</em></td>
<td>Variable name</td>
</tr>
<tr>
<td><em>TYPE</em></td>
<td>Variable type (N for numeric or C for character)</td>
</tr>
<tr>
<td><em>CARDINALITY</em></td>
<td>Number of levels extracted (less than or equal to the value of the MAXLEVELS= option)</td>
</tr>
<tr>
<td><em>MORE</em></td>
<td>Indication of more unreported levels (Y to indicate more levels or N to indicate no more levels)</td>
</tr>
</tbody>
</table>

Figure 14.1 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 14.1 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 14.2:

```plaintext
data details;
  set mycas.details;
  where _varname_ in ('Species', 'SepalLength');
run;

proc print data=details;
  var _VARNAME_ _INDEX_ _FREQ_ _RAWNUM_ _RAWCHAR_;
run;
```
Table 14.2 explains the columns in the details output data table.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>VARNAME</em></td>
<td>Variable name</td>
</tr>
<tr>
<td><em>INDEX</em></td>
<td>Index of the level</td>
</tr>
<tr>
<td><em>FREQ</em></td>
<td>Frequency of the level</td>
</tr>
<tr>
<td><em>RAWNUM</em></td>
<td>Raw level of the variable if numeric</td>
</tr>
<tr>
<td><em>RAWCHAR</em></td>
<td>Raw level of the variable if character</td>
</tr>
</tbody>
</table>

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=CAS-libref.data-table < options > ;
    FREQ variable ;
    VAR variables </options> ;
```

The PROC CARDINALITY statement is required.

**PROC CARDINALITY Statement**

```
PROC CARDINALITY OUTCARD=CAS-libref.data-table < options > ;
```

The PROC CARDINALITY statement invokes the procedure. Table 14.3 summarizes the options available in the PROC CARDINALITY statement.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input and Output Data Sets</strong></td>
<td></td>
</tr>
<tr>
<td>DATA=</td>
<td>Specifies the input data table</td>
</tr>
<tr>
<td>OUTCARD=</td>
<td>Names the cardinality data table to contain the summary information for each variable</td>
</tr>
<tr>
<td>OUTDETAILS=</td>
<td>Names an optional data table to contain the detailed levels of all the variables</td>
</tr>
<tr>
<td><strong>Cardinality Options</strong></td>
<td></td>
</tr>
<tr>
<td>MAXLEVELS=</td>
<td>Specifies the maximum number of levels to consider</td>
</tr>
<tr>
<td>ORDER=</td>
<td>Specifies the order to be applied to all variables that have an unspecified order</td>
</tr>
</tbody>
</table>

You must specify the following option:

**OUTCARD=** *CAS-libref.data-table*

specifies the output data table to contain the cardinality and summary information for each variable. *CAS-libref.data-table* is a two-level name, where *CAS-libref* refers to the caslib and session identifier, 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 504.

You can also specify the following **options**:

**DATA=** *CAS-libref.data-table*

names the input data table for PROC CARDINALITY to use. *CAS-libref.data-table* is a two-level name, where

*CAS-libref* refers to a collection of information that is defined in the LIBNAME statement and includes the caslib, 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 CAS-libref, see the section “Using CAS Sessions and CAS Engine Librefs” on page 504.

*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.

**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=CAS-libref.data-table**

specifies the output data table to contain the levels that are found for each variable.

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 data set 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 the FREQ statement is not specified, each observation is assigned a frequency of 1.

Except for the minimum and the maximum values, the statistics for each variable are affected by the value of *variable*. 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.
Chapter 14: The CARDINALITY Procedure

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 unformatted.

---

Details: CARDINALITY Procedure

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:

```plaintext
proc cardinality data=mycas.iris outcard=mycas.card
   outdetails=mycas.details maxlevels=10;
run;

data card;
   set mycas.card;
run;

proc contents data=card order=varnum;
run;
```
### Figure 14.3 Contents of the Cardinality Output Data Table

**The CONTENTS Procedure**

<table>
<thead>
<tr>
<th>#</th>
<th>Variable</th>
<th>Type</th>
<th>Len</th>
<th>Format</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><em>VARNAME</em></td>
<td>Char</td>
<td>32</td>
<td>$</td>
<td>Variable name</td>
</tr>
<tr>
<td>2</td>
<td><em>FMTWIDTH</em></td>
<td>Num</td>
<td>8</td>
<td>BEST.</td>
<td>Width of the variable formatted value</td>
</tr>
<tr>
<td>3</td>
<td><em>TYPE</em></td>
<td>Char</td>
<td>1</td>
<td>$</td>
<td>Type of the raw values</td>
</tr>
<tr>
<td>4</td>
<td><em>RLEVEL</em></td>
<td>Char</td>
<td>10</td>
<td>$</td>
<td>Recommended level for analytics</td>
</tr>
<tr>
<td>5</td>
<td><em>ORDER</em></td>
<td>Char</td>
<td>8</td>
<td>$</td>
<td>Variable sort order</td>
</tr>
<tr>
<td>6</td>
<td><em>MORE</em></td>
<td>Char</td>
<td>1</td>
<td>$</td>
<td>Have more unreported levels</td>
</tr>
<tr>
<td>7</td>
<td><em>CARDINALITY</em></td>
<td>Num</td>
<td>8</td>
<td>BEST.</td>
<td>Number of levels</td>
</tr>
<tr>
<td>8</td>
<td><em>NOBS</em></td>
<td>Num</td>
<td>8</td>
<td>BEST.</td>
<td>Number of observations</td>
</tr>
<tr>
<td>9</td>
<td><em>SUMFREQS</em></td>
<td>Num</td>
<td>8</td>
<td>BEST.</td>
<td>Total summation of frequencies</td>
</tr>
<tr>
<td>10</td>
<td><em>NMISS</em></td>
<td>Num</td>
<td>8</td>
<td>BEST.</td>
<td>Number of missing values</td>
</tr>
<tr>
<td>11</td>
<td><em>MISSFMT</em></td>
<td>Char</td>
<td>32</td>
<td>$</td>
<td>Format of the missing value</td>
</tr>
<tr>
<td>12</td>
<td><em>VISIBLE</em></td>
<td>Num</td>
<td>8</td>
<td>BEST.</td>
<td>Percentage of the visible part of the report</td>
</tr>
<tr>
<td>13</td>
<td><em>MIN</em></td>
<td>Num</td>
<td>8</td>
<td>BEST.</td>
<td>Minimum numeric value</td>
</tr>
<tr>
<td>14</td>
<td><em>MAX</em></td>
<td>Num</td>
<td>8</td>
<td>BEST.</td>
<td>Maximum numeric value</td>
</tr>
<tr>
<td>15</td>
<td><em>MEAN</em></td>
<td>Num</td>
<td>8</td>
<td>BEST.</td>
<td>Mean</td>
</tr>
<tr>
<td>16</td>
<td><em>STDDEV</em></td>
<td>Num</td>
<td>8</td>
<td>BEST.</td>
<td>Standard deviation</td>
</tr>
<tr>
<td>17</td>
<td><em>SKEWNESS</em></td>
<td>Num</td>
<td>8</td>
<td>BEST.</td>
<td>Skewness</td>
</tr>
<tr>
<td>18</td>
<td><em>KURTOSIS</em></td>
<td>Num</td>
<td>8</td>
<td>BEST.</td>
<td>Kurtosis</td>
</tr>
<tr>
<td>19</td>
<td><em>MREQ</em></td>
<td>Num</td>
<td>8</td>
<td>BEST.</td>
<td>Maximum frequency</td>
</tr>
<tr>
<td>20</td>
<td><em>MFREQFOUNDLEVEL</em></td>
<td>Char</td>
<td>1</td>
<td>$</td>
<td>Found maximum frequency in the visible part of the report</td>
</tr>
<tr>
<td>21</td>
<td><em>MFREQNUM</em></td>
<td>Num</td>
<td>8</td>
<td>BEST.</td>
<td>Numeric level with the maximum frequency</td>
</tr>
<tr>
<td>22</td>
<td><em>MFREQCHR</em></td>
<td>Char</td>
<td>32</td>
<td>$</td>
<td>Character level with the maximum frequency</td>
</tr>
<tr>
<td>23</td>
<td><em>MFREQCFMT</em></td>
<td>Char</td>
<td>32</td>
<td>$</td>
<td>Formatted level with the maximum frequency</td>
</tr>
<tr>
<td>24</td>
<td><em>LASTNUM</em></td>
<td>Num</td>
<td>8</td>
<td>BEST.</td>
<td>Last raw numeric value (cutoff)</td>
</tr>
<tr>
<td>25</td>
<td><em>LASTCHR</em></td>
<td>Char</td>
<td>32</td>
<td>$</td>
<td>Last raw character value (cutoff)</td>
</tr>
<tr>
<td>26</td>
<td><em>LASTCFMT</em></td>
<td>Char</td>
<td>32</td>
<td>$</td>
<td>Last formatted value (cutoff)</td>
</tr>
</tbody>
</table>

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 14.4 Contents of the Details Output Data Table

**The CONTENTS Procedure**

<table>
<thead>
<tr>
<th>#</th>
<th>Variable</th>
<th>Type</th>
<th>Len</th>
<th>Format</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><em>VARNAME</em></td>
<td>Char</td>
<td>32</td>
<td>$</td>
<td>Variable name</td>
</tr>
<tr>
<td>2</td>
<td><em>INDEX</em></td>
<td>Num</td>
<td>8</td>
<td>BEST.</td>
<td>Level index</td>
</tr>
<tr>
<td>3</td>
<td><em>FREQ</em></td>
<td>Num</td>
<td>8</td>
<td>BEST.</td>
<td>Level frequency</td>
</tr>
<tr>
<td>4</td>
<td>FREQPERCENT</td>
<td>Num</td>
<td>8</td>
<td>BEST.</td>
<td>Percentage of the frequency of current level</td>
</tr>
<tr>
<td>5</td>
<td>NMISPERCENT</td>
<td>Num</td>
<td>8</td>
<td>BEST.</td>
<td>Percentage of non missing values</td>
</tr>
<tr>
<td>6</td>
<td><em>RAWNUM</em></td>
<td>Num</td>
<td>8</td>
<td>BEST.</td>
<td>Raw numeric value</td>
</tr>
<tr>
<td>7</td>
<td><em>RAWCHAR</em></td>
<td>Char</td>
<td>32</td>
<td>$</td>
<td>Raw character value</td>
</tr>
<tr>
<td>8</td>
<td><em>CFMT</em></td>
<td>Char</td>
<td>32</td>
<td>$</td>
<td>Formatted value of a variable</td>
</tr>
</tbody>
</table>
Examples: CARDINALITY 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 8 and “Loading a SAS Data Set onto a CAS Server” on page 9 in Chapter 2, “Shared Concepts.”

Example 14.1: Limited Cardinality of the Species Variable

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 505.

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:

```sas
data mycas.iris;
  set sashelp.iris;
run;
```

The following statements run PROC CARDINALITY and plot the levels from the details data set for the variable Species:

```sas
proc cardinality data=mycas.iris outcard=mycas.card
  outdetails=mycas.details maxlevels=5;
run;

data sp;
  set mycas.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 14.1.1 shows that the variable Species has only three levels in the details data.
Whether MAXLEVELS=100 or 5, the Species variable remains fully visible with a _CARDINALITY_ value of 3.

**Example 14.2: Limited Cardinality of the Sepal Length Variable**

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 505.

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:

```sas
data mycas.iris;
  set sashelp.iris;
run;
```

The following statements run PROC CARDINALITY and display some of the contents of the cardinality data set:

```sas
proc cardinality data=mycas.iris outcard=mycas.card outdetails=mycas.details maxlevels=100;
run;
```

```sas
proc print data=mycas.card;
  var _varname_ _type_ _cardinality_ _more_ _visible_ _min_ _max_;
run;
```

Output 14.2.1 shows that all the variables have a _VISIBILITY_ value of 100%.
Chapter 14: The CARDINALITY Procedure

Output 14.2.1 Visibility of the Variables

<table>
<thead>
<tr>
<th>Obs</th>
<th>VARNAME</th>
<th>TYPE</th>
<th>CARDINALITY</th>
<th>MORE</th>
<th>VISIBLE</th>
<th>MIN</th>
<th>MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Species</td>
<td>C</td>
<td>3</td>
<td>N</td>
<td>100</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>2</td>
<td>SepalLength</td>
<td>N</td>
<td>35</td>
<td>N</td>
<td>100</td>
<td>43</td>
<td>79</td>
</tr>
<tr>
<td>3</td>
<td>SepalWidth</td>
<td>N</td>
<td>23</td>
<td>N</td>
<td>100</td>
<td>20</td>
<td>44</td>
</tr>
<tr>
<td>4</td>
<td>PetalLength</td>
<td>N</td>
<td>43</td>
<td>N</td>
<td>100</td>
<td>10</td>
<td>69</td>
</tr>
<tr>
<td>5</td>
<td>PetalWidth</td>
<td>N</td>
<td>22</td>
<td>N</td>
<td>100</td>
<td>1</td>
<td>25</td>
</tr>
</tbody>
</table>

The following statements extract and display the levels of the variable SepalLength:

```plaintext
data sp;
  label _cfmt_='Formatted Value of the Variable SepalLength';
  set mycas.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 14.2.2 shows the histogram.

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.

```plaintext
proc cardinality data=mycas.iris outcard=mycas.card
  outdetails=mycas.details maxlevels=5;
run;

data sp;
```
Example 14.3: More Levels of the Sepal Length Variable

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:

```plaintext
data mycas.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 14.2 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:

```plaintext
proc cardinality data=mycas.iris (where=(SepalLength > 47))
   outcard=mycas.card outdetails=mycas.details maxlevels=5;
   var SepalLength;
   run;

data sp;
   set mycas.details;
```

Output 14.2.3 shows the full histogram of the variable SepalLength. It shows all 35 levels of the variable.

Output 14.2.3 Limited Histogram of the SepalLength Variable

![Histogram of SepalLength](image)
Chapter 14: The CARDINALITY Procedure

```sas
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 14.3.1 Histogram for SepalLength Levels Greater Than 47

The union of the detailed levels from Example 14.2 and this example cover 10 levels of the SepalLength variable.

Table 14.4 lists the variables that are related to the maximum frequency in the histogram.

Table 14.4 More Variables from the Cardinality Data

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>MFREQ</em></td>
<td>Maximum frequency</td>
</tr>
<tr>
<td><em>MFREQFOUNDLEVEL</em></td>
<td>Maximum frequency found in the visible part of the report</td>
</tr>
<tr>
<td><em>MFREQNUM</em></td>
<td>Numeric level that has the maximum frequency</td>
</tr>
<tr>
<td><em>MFREQCHR</em></td>
<td>Character level that has the maximum frequency</td>
</tr>
<tr>
<td><em>MFREQFMT</em></td>
<td>Formatted level that has the maximum frequency</td>
</tr>
</tbody>
</table>

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.
**Example 14.4: A Variable with a User-Defined Format**

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:

```plaintext
data mycas.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:

```plaintext
proc format casfmtlib='myfmtlib';
  value engsize
    low  - <3 = 'toys'
    3  - <6 = 'granny'
    6  - high = 'usable';
run;

data mycas.cars;
  format engineSize engsize.;
```

This code is equivalent to:

```plaintext
proc cardinality data=mycas.iris outcard=mycas.card
   outdetails=mycas.details maxlevels=5;
run;
```

```plaintext
proc print data=mycas.card;
  var _varname_ _mf;
run;
```

**Output 14.3.2 Maximum Frequency Information**

<table>
<thead>
<tr>
<th>Obs</th>
<th><em>VARNAME</em></th>
<th><em>MFREQ</em></th>
<th><em>MFREQFOUNDLEVEL</em></th>
<th><em>MFREQNUM</em></th>
<th><em>MFREQCHR</em></th>
<th><em>MFREQCFMT</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Species</td>
<td>50</td>
<td>Y</td>
<td></td>
<td>. Setosa</td>
<td>Setosa</td>
</tr>
<tr>
<td>2</td>
<td>SepalLength</td>
<td>139</td>
<td>N</td>
<td></td>
<td>.</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>SepalWidth</td>
<td>131</td>
<td>N</td>
<td></td>
<td>.</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>PetalLength</td>
<td>126</td>
<td>N</td>
<td></td>
<td>.</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>PetalWidth</td>
<td>101</td>
<td>N</td>
<td></td>
<td>.</td>
<td></td>
</tr>
</tbody>
</table>

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).
set mycas.cars;
run;

proc cardinality data=mycas.cars outcard=mycas.card
   outdetails=mycas.details maxlevels=5;
   var engineSize;
run;

title 'Cars data with a user-defined format';
proc print data=mycas.card;
   var _varname_ _order_ _more_ _cardinality_; 
run;

Output 14.4.1 Summary of the Variable engineSize

Cars data with a user-defined format

<table>
<thead>
<tr>
<th>Obs</th>
<th><em>VARNAME</em></th>
<th><em>ORDER</em></th>
<th><em>MORE</em></th>
<th><em>CARDINALITY</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>engineSize</td>
<td>ASCFMT</td>
<td>N</td>
<td>3</td>
</tr>
</tbody>
</table>

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 14.4.2:

proc print data=mycas.details;
   var _varname_ _index_ _freq_ _cfmt_; 
run;

Output 14.4.2 Details Data with a User-Defined Format for the Variable engineSize

Cars data with a user-defined format

<table>
<thead>
<tr>
<th>Obs</th>
<th><em>VARNAME</em></th>
<th><em>INDEX</em></th>
<th><em>FREQ</em></th>
<th><em>CFMT</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>engineSize</td>
<td>1</td>
<td>242</td>
<td>granny</td>
</tr>
<tr>
<td>2</td>
<td>engineSize</td>
<td>2</td>
<td>178</td>
<td>toys</td>
</tr>
<tr>
<td>3</td>
<td>engineSize</td>
<td>3</td>
<td>8</td>
<td>usable</td>
</tr>
</tbody>
</table>

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 14.5: Forcing Another Order on the engineSize Variable

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 mycas.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=mycas.cars outcard=mycas.card
  outdetails=mycas.details maxlevels=5;
  var engineSize /order=asc;
run;
```

```
proc print data=mycas.card;
  var _varname_ _order_ _more_ _cardinality_;
run;
```

Output 14.5.1 Summary of the engineSize Variable without a Format

**Cars data with a user-defined format with an ASC order**

<table>
<thead>
<tr>
<th>Obs</th>
<th><em>VARNAME</em></th>
<th><em>ORDER</em></th>
<th><em>MORE</em></th>
<th><em>CARDINALITY</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>EngineSize</td>
<td>ASC</td>
<td>Y</td>
<td>5</td>
</tr>
</tbody>
</table>

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 mycas.details;
  if _index_ = . then do;
    _cfmt_ = cats(">", put(_rawnum_,best12.));
    _rawnum_ = .;
  end;
  _cfmt_ = left(_cfmt_);
run;
```

```
proc print data=details;
run;
```
Output 14.5.2 Details of the engineSize Variable without a Format

Cars data with a user-defined format with an ASC order

<table>
<thead>
<tr>
<th>Obs</th>
<th><em>VARNAME</em></th>
<th><em>INDEX</em></th>
<th><em>FREQ</em></th>
<th>FREQPERCENT</th>
<th>NMISSPERCENT</th>
<th><em>RAWNUM</em></th>
<th><em>CFMT</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>EngineSize</td>
<td>1</td>
<td>2</td>
<td>0.4672897196</td>
<td>0.4672897196</td>
<td>1.3</td>
<td>1.3</td>
</tr>
<tr>
<td>2</td>
<td>EngineSize</td>
<td>2</td>
<td>1</td>
<td>0.2336448598</td>
<td>0.2336448598</td>
<td>1.4</td>
<td>1.4</td>
</tr>
<tr>
<td>3</td>
<td>EngineSize</td>
<td>3</td>
<td>6</td>
<td>1.4018691589</td>
<td>1.4018691589</td>
<td>1.5</td>
<td>1.5</td>
</tr>
<tr>
<td>4</td>
<td>EngineSize</td>
<td>4</td>
<td>10</td>
<td>2.3364485981</td>
<td>2.3364485981</td>
<td>1.6</td>
<td>1.6</td>
</tr>
<tr>
<td>5</td>
<td>EngineSize</td>
<td>5</td>
<td>4</td>
<td>0.9345794393</td>
<td>0.9345794393</td>
<td>1.7</td>
<td>1.7</td>
</tr>
<tr>
<td>6</td>
<td>EngineSize</td>
<td>.</td>
<td>405</td>
<td>94.626168224</td>
<td>94.626168224</td>
<td>.&gt;1.7</td>
<td></td>
</tr>
</tbody>
</table>

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

Chapter 15
The PARTITION Procedure

Overview: PARTITION Procedure

The PARTITION procedure performs sampling in SAS Viya. It performs simple random sampling, stratified sampling, and oversampling 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 data table
- one frequency table, which contains the frequency information for the population and sample
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 mycas cas sessref=mysess;
```

The CAS statement creates the CAS session named mysess, and the LIBNAME statement creates the mycas 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 8 in Chapter 2, “Shared Concepts.”

Getting Started: PARTITION 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 8 and “Loading a SAS Data Set onto a CAS Server” on page 9 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 mycas.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:

```plaintext
data mycas.hmeq;
    set sampsio.hmeq;
run;
```

The following statements perform the partitioning:

```plaintext
proc partition data=mycas.hmeq samppct=10 samppct2=20 seed=1234 partind;
    by bad;
    output out=mycas.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, mycas.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 15.1) that shows the frequency of observations in each level of bad.

![Figure 15.1 Frequency Information Table](image_url)

---

**The PARTITION Procedure**

<table>
<thead>
<tr>
<th>Stratified Sampling Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
</tr>
<tr>
<td>Index</td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td>0 0</td>
</tr>
<tr>
<td>1 1</td>
</tr>
</tbody>
</table>
Syntax: PARTITION Procedure

The following statements are available in the PARTITION procedure:

\[
\text{PROC PARTITION } < \text{options} > ; \\
\quad \text{BY} \quad \text{variable} < \text{variable} \ldots \text{variable} > ; \\
\quad \text{OUTPUT} \quad \text{OUT=} \! \! \text{CAS-libref.data-table} < \text{options} > ; \\
\quad \text{DISPLAY} < \text{table-list} > < /\text{options} > ; \\
\quad \text{DISPLAYOUT} \quad \text{table-spec-list} < /\text{options} > ; \\
\]

The BY statement is required for stratified sampling and oversampling. The OUTPUT statement is required for all sampling methods.

PROC PARTITION statement

\[
\text{PROC PARTITION } < \text{options} > ; \\
\]

The PROC PARTITION statement invokes the procedure.

You can specify the following options for all types of sampling:

\[
\text{DATA=} \! \! \text{CAS-libref.data-table} \\
\quad \text{names the input data table for PROC PARTITION to use. CAS-libref.data-table is a two-level name,} \\
\quad \text{where} \\
\quad \text{CAS-libref} \quad \text{refers to a collection of information that is defined in the LIBNAME} \\
\quad \text{statement and includes the caslib, which includes a path to the data, and a session identifier, which} \\
\quad \text{defaults to the active session but which can be explicitly defined in the LIBNAME} \\
\quad \text{statement. For more information about CAS-libref, see the section “Using CAS} \\
\quad \text{Sessions and CAS Engine Librefs” on page 522.} \\
\quad \text{data-table} \quad \text{specifies the name of the input data table.} \\
\]

\[
\text{NTHREADS=} \! \! \text{number-of-threads} \\
\quad \text{specifies the number of threads to be used. The default is the CPU count on each node.} \\
\]

\[
\text{PARTIND} \\
\quad \text{adds to the output data table a partition indicator, _PartInd_, which indicates whether an observation is} \\
\quad \text{selected to a partition (1 or 2) or not (0).} \\
\]

\[
\text{SEED=} \! \! \text{random-seed} \\
\quad \text{specifies an integer that is used to start the pseudorandom number generator. If you do not specify a} \\
\quad \text{seed or you specify a value less than or equal to 0, the seed is generated from reading the time of day} \\
\quad \text{from the computer’s clock. The SEED=} \! \! \text{option enables you to reproduce the same sample output.} \\
\]

You can specify the following options only for simple random sampling and stratified sampling:
SAMPPCT=sample-percentage
specifies what percentage of the data is to be sampled, where \textit{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 \textit{sample-percentage}, and the percentage of the sample whose _PartInd_ is 0 is 100 minus \textit{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 \textit{sample-percentages} specified 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:

\[
\begin{align*}
    n &= \text{round}((\text{sample-percentage} + \text{sample-percentage-2}) \times N) \\
    n_1 &= \text{round}((\text{sample-percentage}/(\text{sample-percentage} + \text{sample-percentage-2})) \times n) \\
    n_2 &= n - n_1
\end{align*}
\]

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 \textit{options} only for oversampling:

\textbf{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.

\textbf{EVENTPROP=event-proportion}
specifies the proportion of rare events that you want in the sample, where \textit{event-proportion} is a positive number less than or equal to 1.

\textbf{SAMPPCTEVT=sample-event-percentage}
specifies the sample percentage from the rare event level, where \textit{sample-event-percentage} is a positive number less than or equal to 100.

\section*{BY Statement}

\textbf{BY variable < variable ... variable> ;}

The \textit{BY} statement specifies classification variables to be used for stratification. There is no limit on the \textit{BY} variables for stratified sampling, but only one \textit{BY} variable is allowed for oversampling.
DISPLAY Statement

```plaintext
DISPLAY < table-list > < / options > ;
```

The DISPLAY statement enables you to specify a list of ODS 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, all ODS tables are sent to the client and then the client creates a subset. If both DISPLAY and ODS statements are used together, the DISPLAY statement takes precedence over the ODS statements. For more information about ODS, see SAS Output Delivery System: Procedures Guide.

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

- **CASESENSITIVE**
  - performs a case-sensitive comparison of table names in the `table-list` to ODS 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 ODS tables except those specified in the `table-list`.

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

- **TRACE**
  - displays the ODS table names, labels, and paths.

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

A path is a table name that is prefixed with dot-separated grouping information. For example, a SelectionSummary table that is produced by a procedure during a selection routine might have the path `Bygroup1.Summary.SelectionSummary`. A partial pathname does not include all groups; for example, `Selection-` and `Summary.SelectionSummary` are partial pathnames for `Bygroup1.Summary.SelectionSummary`.

When you specify a table name or partial pathname, all ODS 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 starts with a “/” or a “!”. 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.

DISPLAYOUT Statement

```plaintext
DISPLAYOUT table-spec-list < / options > ;
```
The DISPLAYOUT statement enables you to create CAS 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 CAS output tables to create. Each entry in the list has either a key or a key=value format:

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

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 (/):

- **NOREPLACE**
  
  does not replace an existing CAS output table of the same name.

- **REPEATED**
  
  replicates the CAS output tables on all nodes.

---

**OUTPUT Statement**

```
OUTPUT OUT=CAS-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=**CAS-libref.data-table

  names the output data table for PROC PARTITION to use. You must specify this option before any other options. *CAS-libref.data-table* is a two-level name, where

  - *CAS-libref* refers to a collection of information that is defined in the LIBNAME statement and includes the caslib, 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 *CAS-libref*, see the section “Using CAS Sessions and CAS Engine Librefs” on page 522.

  - *data-table* specifies the name of the output data table.

This table includes variables that are specified in the COPYVARS= option. 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. _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 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.
FREQNAME=freq-name
        renames the output data table’s _Freq_ column (which is generated in the oversampling case) 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.

Details: PARTITION Procedure

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 observa-
tions 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.

**Output CAS Tables**

When you specify the OUTPUT statement, the PARTITION procedure produces the output data table along with a table that lists the CAS library, the data table name, and the number of rows and columns in that data table.

---

### 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 15.1.

**Table 15.1** ODS Tables Produced by PROC PARTITION

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Statement</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>OutCASTblFull</td>
<td>Contains the name of the CAS library, name of the output data table, and number of rows and columns in the output data table</td>
<td>OUTPUT OUT=</td>
<td></td>
</tr>
<tr>
<td>OVERFreq</td>
<td>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)</td>
<td>PROC PARTITION</td>
<td>Default</td>
</tr>
<tr>
<td>SRSFreq</td>
<td>Frequency table of input data size and sample size (this table is produced for simple random sampling)</td>
<td>PROC PARTITION</td>
<td>Default</td>
</tr>
<tr>
<td>STRAFreq</td>
<td>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)</td>
<td>PROC PARTITION</td>
<td>Default</td>
</tr>
</tbody>
</table>
Examples: PARTITION 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 8 and “Loading a SAS Data Set onto a CAS Server” on page 9 in Chapter 2, “Shared Concepts.”

### Example 15.1: Simple Random Sampling

This example demonstrates how to use PROC PARTITION to perform simple random sampling on the mycas.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:

```sas
data mycas.hmeq;
    set sampsio.hmeq;
run;
```

The following statements perform the sampling:

```sas
proc partition data=mycas.hmeq samppct=10 seed=10 nthreads=1;
    output out=mycas.out2 copyvars=(job reason loan value delinq derog);
    display 'SRSFreq';
run;
```

```sas
proc print data=mycas.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 mycas.out2, and the COPYVARS= option lists the variables to be copied from mycas.hmeq to mycas.out2. The DISPLAY statement requests that the SRSFreq ODS table be displayed.

Output 15.1.1 shows the number of observations in the mycas.hmeq data table and the number of samples.

**Output 15.1.1** Frequency Information Table

<table>
<thead>
<tr>
<th>The PARTITION Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple Random Sampling</td>
</tr>
<tr>
<td>Frequency</td>
</tr>
<tr>
<td>Number of Obs</td>
</tr>
<tr>
<td>5960</td>
</tr>
</tbody>
</table>

Output 15.1.2 shows the sample data, which are stored in the mycas.out2 data table.
Example 15.2: Stratified Sampling

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 15.1.

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:

```plaintext
data mycas.hmeq;
  set sampsio.hmeq;
run;
```

The following statements perform the partitioning:

```plaintext
proc partition data=mycas.hmeq samppct=10 samppct2=20 seed=10 partind nthreads=3;
  by bad;
  output out=mycas.out3 copyvars=(job reason loan value delinq derog);
run;
```

```plaintext
proc print data=mycas.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, mycas.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 mycas.out3, and the COPYVARS= option lists the variables to be copied from mycas.hmeq to mycas.out3.

Output 15.2.1 shows the frequency information for each level of BY variable bad in the mycas.hmeq data table.

**Output 15.2.1** Frequency Information Table

<table>
<thead>
<tr>
<th>Index</th>
<th>BAD</th>
<th>Number of Obs</th>
<th>Sample Size 1</th>
<th>Sample Size 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>4771</td>
<td>477</td>
<td>954</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1189</td>
<td>119</td>
<td>238</td>
</tr>
</tbody>
</table>

Output 15.2.2 shows the first 20 output sample observations in mycas.out3; the _PartInd_ column shows which partition the observation is selected for (1 for training or 2 for testing) or none (0).

**Output 15.2.2** Sample Output with Partition Indicator

<table>
<thead>
<tr>
<th>Obs</th>
<th>JOB</th>
<th>REASON</th>
<th>LOAN</th>
<th>VALUE</th>
<th>DELINQ</th>
<th>DEROG</th>
<th><em>PartInd</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Other</td>
<td>HomeImp</td>
<td>1100</td>
<td>39025</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td>1500</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>Other</td>
<td>HomeImp</td>
<td>1800</td>
<td>57037</td>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>Sales</td>
<td>HomeImp</td>
<td>2000</td>
<td>62250</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>Other</td>
<td>HomeImp</td>
<td>2000</td>
<td>55000</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>Other</td>
<td>HomeImp</td>
<td>2200</td>
<td>34687</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>Other</td>
<td>HomeImp</td>
<td>2300</td>
<td>40150</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>ProfEx</td>
<td>HomeImp</td>
<td>2400</td>
<td>73395</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>Other</td>
<td>HomeImp</td>
<td>2400</td>
<td>17180</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>HomeImp</td>
<td>2500</td>
<td>20200</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>ProfEx</td>
<td>HomeImp</td>
<td>2500</td>
<td>78600</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>ProfEx</td>
<td>DebtCon</td>
<td>2900</td>
<td>113000</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>13</td>
<td>Other</td>
<td>HomeImp</td>
<td>2900</td>
<td>67996</td>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>Other</td>
<td>HomeImp</td>
<td>3000</td>
<td>20300</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>15</td>
<td>Other</td>
<td>HomeImp</td>
<td>3000</td>
<td>193500</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>Other</td>
<td>HomeImp</td>
<td>3000</td>
<td>14100</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>Mgr</td>
<td>HomeImp</td>
<td>3000</td>
<td>71500</td>
<td>2</td>
<td>.</td>
<td>2</td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td>3100</td>
<td>70400</td>
<td>.</td>
<td>.</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>Other</td>
<td>HomeImp</td>
<td>3200</td>
<td>40834</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>Mgr</td>
<td>HomeImp</td>
<td>3200</td>
<td>.</td>
<td>2</td>
<td>.</td>
<td>0</td>
</tr>
</tbody>
</table>
Example 15.3: Oversampling

This example demonstrates how to use PROC PARTITION to perform oversampling; it uses the same data table as in Example 15.1.

You can load the sampio.hmeq data set into your CAS session by naming your CAS engine libref in the first statement of the following DATA step:

```plaintext
data mycas.hmeq;
  set sampio.hmeq;
run;
```

The following statements perform oversampling:

```plaintext
proc partition data=mycas.hmeq samppctevt=90 eventprop=0.5 event="1" seed=10 nthreads=1;
  by bad;
  ods output OVERFREQ=outFreq;
  output out=mycas.out4 copyvars=(job loan value delinq derog) freqname=_Freq2_;
run;
```

```plaintext
proc print data=mycas.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 mycas.out4, specifies the variables to be transferred from the input data table, requests that the _Freq_ column be renamed to _Freq2_.

Output 15.3.1 shows the number of observations in the sample and in each level of the BY variable bad in the mycas.hmeq data table.

Output 15.3.1 Frequency Information Table

```
<table>
<thead>
<tr>
<th>Index</th>
<th>BAD</th>
<th>Number of Obs</th>
<th>Number of Samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>4771</td>
<td>1070</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1189</td>
<td>1070</td>
</tr>
</tbody>
</table>
```

Output 15.3.2 shows the first 20 output sample observations in the mycas.out4 data table; the _Freq2_ column shows the ratio of target level’s proportion in the population to its proportion in the sample.
### Output 15.3.2 Output Data Table

<table>
<thead>
<tr>
<th>Obs</th>
<th>JOB</th>
<th>LOAN</th>
<th>VALUE</th>
<th>DELINQ</th>
<th>DEROG</th>
<th>Freq2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Other</td>
<td>1100</td>
<td>39025</td>
<td>0</td>
<td>0</td>
<td>0.39899</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>1500</td>
<td></td>
<td></td>
<td>0</td>
<td>0.39899</td>
</tr>
<tr>
<td>3</td>
<td>Other</td>
<td>1800</td>
<td>57037</td>
<td>2</td>
<td>3</td>
<td>0.39899</td>
</tr>
<tr>
<td>4</td>
<td>Sales</td>
<td>2000</td>
<td>62250</td>
<td>0</td>
<td>0</td>
<td>0.39899</td>
</tr>
<tr>
<td>5</td>
<td>Other</td>
<td>2000</td>
<td>55000</td>
<td>0</td>
<td>0</td>
<td>0.39899</td>
</tr>
<tr>
<td>6</td>
<td>Other</td>
<td>2200</td>
<td>34687</td>
<td>1</td>
<td>0</td>
<td>0.39899</td>
</tr>
<tr>
<td>7</td>
<td>Other</td>
<td>2300</td>
<td>40150</td>
<td>0</td>
<td>0</td>
<td>0.39899</td>
</tr>
<tr>
<td>8</td>
<td>ProfExe</td>
<td>2400</td>
<td>73395</td>
<td>0</td>
<td>1</td>
<td>0.39899</td>
</tr>
<tr>
<td>9</td>
<td>Other</td>
<td>2400</td>
<td>17180</td>
<td>0</td>
<td>0</td>
<td>0.39899</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>2500</td>
<td>20200</td>
<td>0</td>
<td>0</td>
<td>0.39899</td>
</tr>
<tr>
<td>11</td>
<td>ProfExe</td>
<td>2900</td>
<td>113000</td>
<td>0</td>
<td>1</td>
<td>0.39899</td>
</tr>
<tr>
<td>12</td>
<td>Other</td>
<td>2900</td>
<td>67996</td>
<td>0</td>
<td>3</td>
<td>0.39899</td>
</tr>
<tr>
<td>13</td>
<td>Other</td>
<td>3000</td>
<td>20300</td>
<td>0</td>
<td>0</td>
<td>0.39899</td>
</tr>
<tr>
<td>14</td>
<td>Other</td>
<td>3000</td>
<td>193500</td>
<td>0</td>
<td>0</td>
<td>0.39899</td>
</tr>
<tr>
<td>15</td>
<td>Other</td>
<td>3000</td>
<td>14100</td>
<td>0</td>
<td>0</td>
<td>0.39899</td>
</tr>
<tr>
<td>16</td>
<td>Mgr</td>
<td>3000</td>
<td>71500</td>
<td>2</td>
<td></td>
<td>1.60101</td>
</tr>
<tr>
<td>17</td>
<td>Other</td>
<td>3200</td>
<td>40834</td>
<td>0</td>
<td>0</td>
<td>0.39899</td>
</tr>
<tr>
<td>18</td>
<td>Mgr</td>
<td>3200</td>
<td></td>
<td>2</td>
<td></td>
<td>0.39899</td>
</tr>
<tr>
<td>19</td>
<td>Office</td>
<td>3400</td>
<td>52000</td>
<td>1</td>
<td>0</td>
<td>0.39899</td>
</tr>
<tr>
<td>20</td>
<td>Other</td>
<td>3900</td>
<td>45960</td>
<td>0</td>
<td>0</td>
<td>0.39899</td>
</tr>
</tbody>
</table>
Overview: VARIMPUTE Procedure

The VARIMPUTE procedure performs numeric variable imputation in SAS Viya. Imputation is a common step in data preparation. The VARIMPUTE procedure accepts only numeric variables.

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.

When PROC VARIMPUTE calculates the mean, median, 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 mycas cas sessref=mysess;
```

The CAS statement creates the CAS session named `mysess`, and the LIBNAME statement creates the `mycas` 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 8 in Chapter 2, “Shared Concepts.”

**Getting Started: VARIMPUTE 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 8 and “Loading a SAS Data Set onto a CAS Server” on page 9 in Chapter 2, “Shared Concepts.”

The VARIMPUTE procedure can use any or all of four methods to impute numeric missing values. This example uses all four imputation methods to manipulate a data table. The input data table `mycas.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 table into your CAS session by naming your CAS engine libref in the first statement of the following DATA step:

```
data mycas.hmeq;
  set sampsio.hmeq;
run;
```

These statements assume that your CAS engine libref is named `mycas`, but you can substitute any appropriately defined CAS engine libref.
The following statements perform the imputation:

```plaintext
proc varimpute data=mycas.hmeq seed=18000;
   input clage /ctech=mean;
   input delinq/ctech=median;
   input ninq/ctech=random;
   input debtinc yoj/ctech=value cvalues=50,100;
   output out=mycas.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 CTECH=VALUE option) requests that 50 be imputed as the value of `debtinc` variable and 100 be imputed as the value of `yoj` variable. The OUTPUT statement creates a new data table to contain the imputed values; it is shown in Output 16.1. The Variable column shows the original variable names from the input data table. The Imputation Method column shows the types of imputation methods: value, median, random (between the minimum value and the maximum value of the nonmissing values), and mean. 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 16.1 Imputation Information

<table>
<thead>
<tr>
<th>Variable</th>
<th>Imputation Method</th>
<th>Result Variable</th>
<th>N</th>
<th>Number of Missing</th>
<th>Imputed Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLAGE</td>
<td>Mean</td>
<td>IM_CLAGE</td>
<td>5652</td>
<td>308</td>
<td>179.766275</td>
</tr>
<tr>
<td>DEBTINC</td>
<td>Value</td>
<td>IM_DEBTINC</td>
<td>4693</td>
<td>1267</td>
<td>50</td>
</tr>
<tr>
<td>DELINQ</td>
<td>Median</td>
<td>IM_DELINQ</td>
<td>5380</td>
<td>580</td>
<td>0</td>
</tr>
<tr>
<td>NINQ</td>
<td>Random</td>
<td>IM_NINQ</td>
<td>5450</td>
<td>510</td>
<td>0</td>
</tr>
<tr>
<td>YOJ</td>
<td>Value</td>
<td>IM_YOJ</td>
<td>5445</td>
<td>515</td>
<td>100</td>
</tr>
</tbody>
</table>

**Syntax: VARIMPUTE Procedure**

The following statements are available in the VARIMPUTE procedure:

```plaintext
PROC VARIMPUTE DATA=CAS-libref.data-table < options >;
   CODE FILE=filename ;
   FREQ variable ;
   INPUT variables < /option > ;
   OUTPUT OUT=CAS-libref.data-table < option > ;
```

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=CASE-libref.data-table < options > ;

The PROC VARIMPUTE statement invokes the procedure.

You can specify the following options:

DATA=CASE-libref.data-table

names the input data table for PROC VARIMPUTE to use. The default is the most recently created data table. CASE-libref.data-table is a two-level name, where

  CASE-libref refers to a collection of information that is defined in the LIBNAME statement and includes the caslib, 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 CASE-libref, see the section “Using CAS Sessions and CAS Engine Librefs” on page 535.

  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 FILE=filename ;

The CODE statement generates SAS DATA step code that mimics the computations that are performed. Only one CODE statement is processed. If you specify multiple CODE statements, only the first one is used.

You can specify the following option:

FILE=filename

specifies the name 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 data set 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 the FREQ statement is not specified, each observation is assigned a frequency of 1.

**INPUT Statement**

```
INPUT variables < /option> ;
```

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.

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=double-list** specifies that missing values are replaced by the values in double-list, where double-list indicates a list of numbers where the data type is double.

By default, CONTINUOUSTECH=MEAN.

**OUTPUT Statement**

```
OUTPUT OUT=CAS-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=CAS-libref.data-table** names the output data table for PROC VARIMPUTE to use. You must specify this option before any other options. CAS-libref.data-table is a two-level name, where
**CAS-libref** refers to a collection of information that is defined in the LIBNAME statement and includes the caslib, 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 **CAS-libref**, see the section “Using CAS Sessions and CAS Engine Librefs” on page 535.

**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.

---

**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, 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

\[
  y = \begin{cases} 
    x_j & \text{if } g = 0 \\
    x_{j+1} & \text{if } g > 0 
  \end{cases}
\]

### 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\('uniform')}$, where rand\('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 “Imputation Requests” table lists the imputation method, the number of variables that are imputed, and the seed value for the random imputation.

**Imputation Information**

The “Imputation Information” table lists the names of variables, the imputation method, the numbers of nonmissing and missing values for each variable, and the imputation values that replace missing values.

---

**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 16.1.

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Statement</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ImputeInfo</td>
<td>Imputation method, number of variables that are imputed, and seed for the pseudorandom number generator</td>
<td>PROC VARIMPUTE</td>
<td>Default</td>
</tr>
<tr>
<td>VarImputeInfo</td>
<td>Variables, imputation method, number of nonmissing values, number of missing values, and imputation values</td>
<td>PROC VARIMPUTE</td>
<td>Default</td>
</tr>
</tbody>
</table>

---

**Examples: VARIMPUTE 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 8 and “Loading a SAS Data Set onto a CAS Server” on page 9 in Chapter 2, “Shared Concepts.”

---

**Example 16.1: Imputation of Missing Values for HMEQ Data Table**

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 536.
You can load the sampsio.hmeq data table into your CAS session by naming your CAS engine libref in the first statement of the following DATA step:

```plaintext
data mycas.hmeq;
   set sampsio.hmeq;
run;
```

The following statements perform the imputation:

```plaintext
proc varimpute data=mycas.hmeq seed=12345;
   input derog clno/ctech=value cvalues=5,20;
   input value /ctech=mean;
   input mortdue /ctech=median;
   input ninq /ctech=random;
   output out=mycas.out1;
run;
```

```plaintext
proc print data=mycas.out1(firstobs=110 obs=124);
run;
```

Output 16.1.1 shows the number of variables for which the missing observations are imputed and the random seed value for the random imputation method.

**Output 16.1.1 Imputation Requests**

<table>
<thead>
<tr>
<th>Imputation Requests</th>
<th>Number of Variables</th>
<th>Seed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Random</td>
<td>1 12345</td>
<td></td>
</tr>
<tr>
<td>Median</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Value</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

Output 16.1.2 shows the imputation results.

**Output 16.1.2 Imputation Information**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Imputation Method</th>
<th>Result Variable</th>
<th>Number of Missing</th>
<th>Imputed Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLNO</td>
<td>Value</td>
<td>IM_CLNO</td>
<td>5738</td>
<td>222 20</td>
</tr>
<tr>
<td>DEROG</td>
<td>Value</td>
<td>IM_DEROG</td>
<td>5252</td>
<td>708 5</td>
</tr>
<tr>
<td>MORTDUE</td>
<td>Median</td>
<td>IM_MORTDUE</td>
<td>5442</td>
<td>518 65017</td>
</tr>
<tr>
<td>NINQ</td>
<td>Random</td>
<td>IM_NINQ</td>
<td>5450</td>
<td>510</td>
</tr>
<tr>
<td>VALUE</td>
<td>Mean</td>
<td>IM_VALUE</td>
<td>5848</td>
<td>112 101776.049</td>
</tr>
</tbody>
</table>

Output 16.1.3 shows the 15 output observations of the variables that had missing values and their new imputed values.
### Output 16.1.3 Output data table

<table>
<thead>
<tr>
<th>Obs</th>
<th>IM_CLNO</th>
<th>IM_DEROG</th>
<th>IM_MORTDUE</th>
<th>IM_NINQ</th>
<th>IM_VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>110</td>
<td>22</td>
<td>1</td>
<td>65017</td>
<td>1</td>
<td>95500</td>
</tr>
<tr>
<td>111</td>
<td>20</td>
<td>5</td>
<td>65017 4.73249850515276</td>
<td>43683</td>
<td></td>
</tr>
<tr>
<td>112</td>
<td>20</td>
<td>5</td>
<td>65017 15.8702587487641</td>
<td>47400</td>
<td></td>
</tr>
<tr>
<td>113</td>
<td>12</td>
<td>5</td>
<td>65017 3.18566470243968</td>
<td>35250</td>
<td></td>
</tr>
<tr>
<td>114</td>
<td>22</td>
<td>1</td>
<td>68743</td>
<td>0</td>
<td>65321</td>
</tr>
<tr>
<td>115</td>
<td>20</td>
<td>5</td>
<td>78753 6.26889186142943</td>
<td>95433</td>
<td></td>
</tr>
<tr>
<td>116</td>
<td>22</td>
<td>0</td>
<td>171582</td>
<td>0</td>
<td>245730</td>
</tr>
<tr>
<td>117</td>
<td>23</td>
<td>0</td>
<td>60354</td>
<td>2</td>
<td>87596</td>
</tr>
<tr>
<td>118</td>
<td>17</td>
<td>5</td>
<td>65017 2 101776.04874145</td>
<td>96462</td>
<td></td>
</tr>
<tr>
<td>119</td>
<td>23</td>
<td>2</td>
<td>70793</td>
<td>0</td>
<td>71552</td>
</tr>
<tr>
<td>120</td>
<td>20</td>
<td>5</td>
<td>75121 1.71818908327259</td>
<td>96462</td>
<td></td>
</tr>
<tr>
<td>121</td>
<td>47</td>
<td>0</td>
<td>75623</td>
<td>0</td>
<td>89634</td>
</tr>
<tr>
<td>122</td>
<td>20</td>
<td>0</td>
<td>55000</td>
<td>0</td>
<td>69000</td>
</tr>
<tr>
<td>123</td>
<td>12</td>
<td>5</td>
<td>65017 12.6002731549087</td>
<td>39050</td>
<td></td>
</tr>
<tr>
<td>124</td>
<td>12</td>
<td>2</td>
<td>65875</td>
<td>1</td>
<td>76000</td>
</tr>
</tbody>
</table>
Chapter 17
The VARREDUCE Procedure

Contents

Overview: VARREDUCE Procedure .................................................. 546
 PROC VARREDUCE Features ......................................................... 546
 PROC VARREDUCE Compared to Other SAS Procedures ...................... 547
 Using CAS Sessions and CAS Engine Librefs ................................. 547
 Getting Started: VARREDUCE Procedure .................................... 548
 Unsupervised Variable Selection ................................................. 550
 Supervised Variable Selection ................................................... 552
 Syntax: VARREDUCE Procedure .................................................. 553
 PROC VARREDUCE Statement ..................................................... 553
 CLASS Statement ................................................................. 555
 DISPLAY Statement .............................................................. 556
 DISPLAYOUT Statement .......................................................... 557
 FREQ Statement ................................................................. 557
 REDUCE Statement ............................................................... 557
 Details: VARREDUCE Procedure .................................................. 559
 Missing Values ................................................................. 559
 Unsupervised Variable Selection ................................................. 559
 Supervised Variable Selection ................................................... 560
  Variable Selection for Regression ............................................. 560
  Variable Selection for Classification ........................................ 561
 Criteria Used in Model Selection ............................................... 561
 Computational Method .......................................................... 562
 GLM Parameterization of Classification Variables and Effects .......... 563
  Intercept ............................................................................. 564
  Regression Effects ................................................................ 564
  Main Effects ........................................................................ 564
  Interaction Effects ............................................................. 565
  Nested Effects ..................................................................... 565
  Continuous-Nesting-Class Effects .......................................... 566
  Continuous-by-Class Effects .................................................. 566
  General Effects .................................................................... 567
 Displayed Output ..................................................................... 567
  Number of Observations ....................................................... 568
  Selection Summary ............................................................. 568
  Selected Variables ............................................................. 568
 ODS Table Names .................................................................... 568
Overview: VARREDUCE Procedure

The VARREDUCE procedure performs both supervised and unsupervised variable selection in SAS Viya. You can use the VARREDUCE 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), the VARREDUCE procedure reduces dimensionality by selecting a subset of the original variables (variable selection). Thus, this technique preserves model interpretation.

The VARREDUCE procedure performs supervised variable selection by identifying a set of variables that jointly explain the maximum amount of variance contained in the response variables. The VARREDUCE procedure 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 the VARREDUCE 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 the statistical procedures of SAS Visual Data Mining and Machine Learning 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:
Chapter 17: The VARREDUCE Procedure

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 mycas cas sessref=mysess;

The CAS statement creates the CAS session named mysess, and the LIBNAME statement creates the mycas 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 8 in Chapter 2, “Shared Concepts.”

Getting Started: VARREDUCE 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 8 and “Loading a SAS Data Set onto a CAS Server” on page 9 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 mycas.getStarted data table and loads the table into your CAS session:

    data mycas.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
<p>| | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>0</td>
<td>4</td>
<td>6</td>
<td>0.9</td>
<td>4</td>
<td>36</td>
<td>2.1</td>
<td>30</td>
<td>0.8</td>
<td>9</td>
<td>4.6</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>6.2</td>
<td>2</td>
<td>1.8</td>
<td>14</td>
<td>79</td>
<td>1.1</td>
<td>70</td>
<td>0.2</td>
<td>9</td>
<td>5.1</td>
</tr>
<tr>
<td>H</td>
<td>0</td>
<td>3.7</td>
<td>3</td>
<td>0.8</td>
<td>12</td>
<td>66</td>
<td>1.3</td>
<td>40</td>
<td>0.4</td>
<td>0.5</td>
<td>3.3</td>
</tr>
<tr>
<td>A</td>
<td>1</td>
<td>9.2</td>
<td>3</td>
<td>2.3</td>
<td>48</td>
<td>51</td>
<td>2.3</td>
<td>50</td>
<td>0</td>
<td>6</td>
<td>5.4</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>14</td>
<td>3</td>
<td>2</td>
<td>18</td>
<td>12</td>
<td>2.2</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>3.4</td>
</tr>
<tr>
<td>E</td>
<td>1</td>
<td>19.5</td>
<td>6</td>
<td>3.7</td>
<td>26</td>
<td>81</td>
<td>0.1</td>
<td>30</td>
<td>0.6</td>
<td>5</td>
<td>4.8</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>11</td>
<td>3</td>
<td>2.8</td>
<td>38</td>
<td>9</td>
<td>1.7</td>
<td>50</td>
<td>0.8</td>
<td>6.5</td>
<td>0.9</td>
</tr>
<tr>
<td>I</td>
<td>0</td>
<td>15.3</td>
<td>7</td>
<td>2.2</td>
<td>20</td>
<td>98</td>
<td>2.7</td>
<td>100</td>
<td>0.4</td>
<td>7</td>
<td>0.8</td>
</tr>
<tr>
<td>H</td>
<td>1</td>
<td>7.4</td>
<td>4</td>
<td>0.5</td>
<td>28</td>
<td>65</td>
<td>1.3</td>
<td>60</td>
<td>0.2</td>
<td>9.5</td>
<td>5.4</td>
</tr>
<tr>
<td>F</td>
<td>0</td>
<td>11.4</td>
<td>2</td>
<td>1.4</td>
<td>42</td>
<td>12</td>
<td>2.4</td>
<td>10</td>
<td>0.4</td>
<td>1</td>
<td>4.5</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>19.4</td>
<td>1</td>
<td>0.4</td>
<td>42</td>
<td>4</td>
<td>2.4</td>
<td>10</td>
<td>0</td>
<td>6.5</td>
<td>0.1</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>5.9</td>
<td>4</td>
<td>2.6</td>
<td>12</td>
<td>57</td>
<td>0.8</td>
<td>50</td>
<td>0.4</td>
<td>2</td>
<td>5.8</td>
</tr>
<tr>
<td>G</td>
<td>1</td>
<td>15.8</td>
<td>6</td>
<td>3.7</td>
<td>34</td>
<td>8</td>
<td>1.3</td>
<td>90</td>
<td>0.6</td>
<td>2.5</td>
<td>5.7</td>
</tr>
<tr>
<td>I</td>
<td>0</td>
<td>10</td>
<td>3</td>
<td>1.9</td>
<td>16</td>
<td>80</td>
<td>3</td>
<td>90</td>
<td>0.4</td>
<td>9.5</td>
<td>1.9</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>15.7</td>
<td>1</td>
<td>2.7</td>
<td>32</td>
<td>25</td>
<td>1.7</td>
<td>20</td>
<td>0.2</td>
<td>8.5</td>
<td>6</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>11</td>
<td>5</td>
<td>2.9</td>
<td>48</td>
<td>53</td>
<td>0.1</td>
<td>50</td>
<td>1</td>
<td>3.5</td>
<td>1.2</td>
</tr>
<tr>
<td>J</td>
<td>1</td>
<td>16.8</td>
<td>0</td>
<td>0.9</td>
<td>14</td>
<td>86</td>
<td>1.4</td>
<td>40</td>
<td>0.8</td>
<td>9</td>
<td>5</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>11</td>
<td>4</td>
<td>3.2</td>
<td>48</td>
<td>63</td>
<td>2.8</td>
<td>90</td>
<td>0.6</td>
<td>0</td>
<td>2.2</td>
</tr>
<tr>
<td>J</td>
<td>1</td>
<td>4.8</td>
<td>7</td>
<td>3.6</td>
<td>24</td>
<td>1</td>
<td>2.2</td>
<td>20</td>
<td>1</td>
<td>8.5</td>
<td>0.5</td>
</tr>
<tr>
<td>J</td>
<td>1</td>
<td>10.4</td>
<td>5</td>
<td>2</td>
<td>42</td>
<td>56</td>
<td>1</td>
<td>20</td>
<td>0</td>
<td>3.5</td>
<td>4.2</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>12.7</td>
<td>7</td>
<td>3.6</td>
<td>8</td>
<td>56</td>
<td>2.1</td>
<td>70</td>
<td>1</td>
<td>4.5</td>
<td>1.5</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>6.8</td>
<td>1</td>
<td>3.2</td>
<td>30</td>
<td>27</td>
<td>0.6</td>
<td>0</td>
<td>0.8</td>
<td>2</td>
<td>5.6</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>8.8</td>
<td>0</td>
<td>3.2</td>
<td>2</td>
<td>67</td>
<td>0.7</td>
<td>10</td>
<td>0.4</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>I</td>
<td>1</td>
<td>0.2</td>
<td>0</td>
<td>2.9</td>
<td>10</td>
<td>41</td>
<td>2.3</td>
<td>60</td>
<td>0.2</td>
<td>9</td>
<td>0.3</td>
</tr>
<tr>
<td>J</td>
<td>1</td>
<td>4.6</td>
<td>7</td>
<td>3.9</td>
<td>50</td>
<td>61</td>
<td>2.1</td>
<td>50</td>
<td>0.4</td>
<td>3</td>
<td>4.9</td>
</tr>
<tr>
<td>J</td>
<td>1</td>
<td>2.3</td>
<td>2</td>
<td>3.2</td>
<td>36</td>
<td>98</td>
<td>0.1</td>
<td>40</td>
<td>0.6</td>
<td>4.5</td>
<td>4.3</td>
</tr>
<tr>
<td>I</td>
<td>0</td>
<td>10.8</td>
<td>3</td>
<td>2.7</td>
<td>28</td>
<td>58</td>
<td>0.8</td>
<td>80</td>
<td>0.8</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>9.3</td>
<td>2</td>
<td>3.3</td>
<td>44</td>
<td>44</td>
<td>0.3</td>
<td>50</td>
<td>0.8</td>
<td>5.5</td>
<td>0.4</td>
</tr>
<tr>
<td>F</td>
<td>0</td>
<td>9.2</td>
<td>6</td>
<td>0.6</td>
<td>4</td>
<td>64</td>
<td>0.1</td>
<td>0</td>
<td>0.6</td>
<td>4.5</td>
<td>3.9</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>7.4</td>
<td>0</td>
<td>2.9</td>
<td>14</td>
<td>0</td>
<td>0.2</td>
<td>30</td>
<td>0.8</td>
<td>7.5</td>
<td>4.5</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>18.3</td>
<td>3</td>
<td>3.1</td>
<td>8</td>
<td>60</td>
<td>0.3</td>
<td>60</td>
<td>0.2</td>
<td>7</td>
<td>1.9</td>
</tr>
<tr>
<td>F</td>
<td>0</td>
<td>5.3</td>
<td>4</td>
<td>0.2</td>
<td>48</td>
<td>63</td>
<td>2.3</td>
<td>80</td>
<td>0.2</td>
<td>8</td>
<td>5.2</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>2.6</td>
<td>5</td>
<td>2.2</td>
<td>24</td>
<td>4</td>
<td>1.3</td>
<td>20</td>
<td>0</td>
<td>2</td>
<td>1.4</td>
</tr>
<tr>
<td>F</td>
<td>0</td>
<td>13.8</td>
<td>4</td>
<td>3.6</td>
<td>4</td>
<td>7</td>
<td>1.1</td>
<td>10</td>
<td>0.4</td>
<td>3.5</td>
<td>1.9</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>12.4</td>
<td>6</td>
<td>1.7</td>
<td>30</td>
<td>44</td>
<td>1.1</td>
<td>60</td>
<td>0.2</td>
<td>6</td>
<td>1.5</td>
</tr>
<tr>
<td>I</td>
<td>0</td>
<td>1.3</td>
<td>1</td>
<td>1.3</td>
<td>8</td>
<td>53</td>
<td>1.1</td>
<td>70</td>
<td>0.6</td>
<td>7</td>
<td>0.8</td>
</tr>
<tr>
<td>F</td>
<td>0</td>
<td>18.2</td>
<td>7</td>
<td>1.7</td>
<td>26</td>
<td>92</td>
<td>2.2</td>
<td>30</td>
<td>1</td>
<td>8.5</td>
<td>4.8</td>
</tr>
<tr>
<td>J</td>
<td>0</td>
<td>5.2</td>
<td>2</td>
<td>2.2</td>
<td>18</td>
<td>12</td>
<td>1.4</td>
<td>90</td>
<td>0.8</td>
<td>4</td>
<td>4.9</td>
</tr>
<tr>
<td>G</td>
<td>1</td>
<td>9.4</td>
<td>2</td>
<td>0.8</td>
<td>22</td>
<td>86</td>
<td>0.4</td>
<td>30</td>
<td>0.4</td>
<td>1</td>
<td>5.9</td>
</tr>
<tr>
<td>J</td>
<td>1</td>
<td>10.4</td>
<td>2</td>
<td>1.7</td>
<td>26</td>
<td>31</td>
<td>2.4</td>
<td>10</td>
<td>0.2</td>
<td>7</td>
<td>1.6</td>
</tr>
<tr>
<td>J</td>
<td>0</td>
<td>13</td>
<td>1</td>
<td>1.8</td>
<td>14</td>
<td>11</td>
<td>2.3</td>
<td>50</td>
<td>0.6</td>
<td>5.5</td>
<td>2.6</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>17.9</td>
<td>4</td>
<td>3.1</td>
<td>46</td>
<td>58</td>
<td>2.6</td>
<td>90</td>
<td>0.6</td>
<td>1.5</td>
<td>3.2</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>19.4</td>
<td>6</td>
<td>3</td>
<td>20</td>
<td>50</td>
<td>2.8</td>
<td>100</td>
<td>0.2</td>
<td>9</td>
<td>1.2</td>
</tr>
<tr>
<td>I</td>
<td>0</td>
<td>19.6</td>
<td>3</td>
<td>3.6</td>
<td>22</td>
<td>19</td>
<td>1.2</td>
<td>0</td>
<td>0.6</td>
<td>5</td>
<td>4.1</td>
</tr>
<tr>
<td>I</td>
<td>1</td>
<td>6</td>
<td>2</td>
<td>1.5</td>
<td>30</td>
<td>30</td>
<td>2.2</td>
<td>20</td>
<td>0.4</td>
<td>8.5</td>
<td>5.3</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>13.8</td>
<td>1</td>
<td>2.7</td>
<td>0</td>
<td>52</td>
<td>2.4</td>
<td>20</td>
<td>0.8</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>14.3</td>
<td>4</td>
<td>2.9</td>
<td>30</td>
<td>11</td>
<td>0.6</td>
<td>90</td>
<td>0.6</td>
<td>0.5</td>
<td>4.9</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>15.6</td>
<td>0</td>
<td>0.4</td>
<td>38</td>
<td>79</td>
<td>0.4</td>
<td>80</td>
<td>0.4</td>
<td>1</td>
<td>3.3</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>14</td>
<td>2</td>
<td>1</td>
<td>22</td>
<td>61</td>
<td>3</td>
<td>90</td>
<td>0.6</td>
<td>2</td>
<td>0.1</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>9.4</td>
<td>5</td>
<td>0.4</td>
<td>12</td>
<td>53</td>
<td>1.7</td>
<td>40</td>
<td>0</td>
<td>3</td>
<td>1.1</td>
</tr>
<tr>
<td>H</td>
<td>0</td>
<td>13.2</td>
<td>1</td>
<td>1.6</td>
<td>40</td>
<td>15</td>
<td>0.7</td>
<td>40</td>
<td>0.2</td>
<td>9</td>
<td>5.5</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>13.5</td>
<td>5</td>
<td>2.4</td>
<td>18</td>
<td>89</td>
<td>1.6</td>
<td>20</td>
<td>0.4</td>
<td>9.5</td>
<td>4.7</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>2.6</td>
<td>4</td>
<td>2.3</td>
<td>38</td>
<td>6</td>
<td>0.8</td>
<td>20</td>
<td>0.4</td>
<td>5</td>
<td>5.3</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>12.4</td>
<td>3</td>
<td>1.3</td>
<td>26</td>
<td>8</td>
<td>2.8</td>
<td>10</td>
<td>0.8</td>
<td>6</td>
<td>5.8</td>
</tr>
</tbody>
</table>
### Unsupervised Variable Selection

The following statements use PROC VARREDUCE for unsupervised variable selection:

```sas
proc varreduce data=mycas.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.

These statements assume that your CAS engine libref is named mycas, but you can substitute any appropriately defined CAS engine libref.
The output from this analysis is presented in Figure 17.1 through Figure 17.3.

Figure 17.1 displays the “Number of Observations” tables. This table shows that all 100 observations in the data table are used in the analysis.

**Figure 17.1** Number of Observations

<table>
<thead>
<tr>
<th>The VARREDUCE Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Observations Read</td>
</tr>
<tr>
<td>Number of Observations Used</td>
</tr>
</tbody>
</table>

Figure 17.2 and Figure 17.3 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 17.2** Selection Summary

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Parameter</th>
<th>Proportion of Variance Explained</th>
<th>SSE</th>
<th>MSE</th>
<th>AIC</th>
<th>AICC</th>
<th>BIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>x3</td>
<td>0.068102</td>
<td>18.637953</td>
<td>0.188262</td>
<td>7.105200</td>
<td>27.216339</td>
<td>2.971252</td>
</tr>
<tr>
<td>2</td>
<td>x7</td>
<td>0.132579</td>
<td>17.348424</td>
<td>0.177025</td>
<td>6.993502</td>
<td>26.094008</td>
<td>2.945605</td>
</tr>
<tr>
<td>3</td>
<td>x10</td>
<td>0.193293</td>
<td>16.134131</td>
<td>0.166331</td>
<td>6.860937</td>
<td>24.945494</td>
<td>2.919092</td>
</tr>
<tr>
<td>4</td>
<td>C1</td>
<td>0.252883</td>
<td>14.942348</td>
<td>0.155649</td>
<td>6.704199</td>
<td>23.767490</td>
<td>2.888406</td>
</tr>
<tr>
<td>5</td>
<td>C F</td>
<td>0.311964</td>
<td>13.760728</td>
<td>0.144850</td>
<td>6.521819</td>
<td>22.558528</td>
<td>2.852077</td>
</tr>
<tr>
<td>6</td>
<td>C J</td>
<td>0.368951</td>
<td>12.620982</td>
<td>0.134266</td>
<td>6.315361</td>
<td>21.320171</td>
<td>2.811671</td>
</tr>
<tr>
<td>7</td>
<td>C B</td>
<td>0.425720</td>
<td>11.485609</td>
<td>0.123501</td>
<td>6.081095</td>
<td>20.048690</td>
<td>2.763457</td>
</tr>
<tr>
<td>8</td>
<td>C G</td>
<td>0.481257</td>
<td>10.374861</td>
<td>0.112770</td>
<td>5.819386</td>
<td>18.744449</td>
<td>2.707799</td>
</tr>
<tr>
<td>9</td>
<td>C D</td>
<td>0.535098</td>
<td>9.298037</td>
<td>0.102176</td>
<td>5.529803</td>
<td>17.407018</td>
<td>2.644269</td>
</tr>
<tr>
<td>10</td>
<td>C H</td>
<td>0.588910</td>
<td>8.221792</td>
<td>0.091353</td>
<td>5.206788</td>
<td>16.030839</td>
<td>2.567305</td>
</tr>
<tr>
<td>11</td>
<td>C A</td>
<td>0.640665</td>
<td>7.186698</td>
<td>0.080749</td>
<td>4.852232</td>
<td>14.617801</td>
<td>2.478801</td>
</tr>
<tr>
<td>12</td>
<td>x5</td>
<td>0.691609</td>
<td>6.167815</td>
<td>0.070089</td>
<td>4.459345</td>
<td>13.161117</td>
<td>2.371965</td>
</tr>
</tbody>
</table>

**Figure 17.3** Selected Effects

<table>
<thead>
<tr>
<th>Number</th>
<th>Selected Variable</th>
<th>Variable Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>x3</td>
<td>INTERVAL</td>
</tr>
<tr>
<td>2</td>
<td>x7</td>
<td>INTERVAL</td>
</tr>
<tr>
<td>3</td>
<td>x10</td>
<td>INTERVAL</td>
</tr>
<tr>
<td>4</td>
<td>C</td>
<td>CLASS</td>
</tr>
<tr>
<td>5</td>
<td>x5</td>
<td>INTERVAL</td>
</tr>
</tbody>
</table>
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=mycas.getStarted technique=DiscriminantAnalysis;
  class C y;
  reduce supervised y = C x1-x10 / maxeffects=5;
run;
```

The output from this analysis is presented in Figure 17.4 through Figure 17.6.

Figure 17.4 shows that all 100 observations in the data table are used in the analysis.

![Figure 17.4 Number of Observations](image_url)

**The VARREDUCE Procedure**

<table>
<thead>
<tr>
<th>Number of Observations Read</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Observations Used</td>
<td>100</td>
</tr>
</tbody>
</table>

Figure 17.5 and Figure 17.6 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 17.5 Selection Summary](image_url)

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Parameter</th>
<th>Proportion of Variance Explained</th>
<th>SSE</th>
<th>MSE</th>
<th>AIC</th>
<th>AICC</th>
<th>BIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>C J</td>
<td>0.081099 0.918901 0.009282 0.015424 2.019590 -0.038525</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>x8</td>
<td>0.132303 0.867697 0.008854 -0.001913 2.005456 -0.049809</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>x2</td>
<td>0.168747 0.831253 0.008570 -0.004821 2.006669 -0.046665</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>C C</td>
<td>0.199152 0.800848 0.008342 -0.002084 2.014475 -0.037877</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>x4</td>
<td>0.218366 0.781634 0.008228 0.013631 2.036240 -0.016110</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>x9</td>
<td>0.236868 0.763132 0.008118 0.029676 2.059346 0.005986</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

![Figure 17.6 Selected Effects](image_url)

<table>
<thead>
<tr>
<th>Selected Effects</th>
<th>Selected Variable</th>
<th>Variable Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>C</td>
<td>CLASS</td>
</tr>
<tr>
<td>2</td>
<td>x8</td>
<td>INTERVAL</td>
</tr>
<tr>
<td>3</td>
<td>x2</td>
<td>INTERVAL</td>
</tr>
<tr>
<td>4</td>
<td>x4</td>
<td>INTERVAL</td>
</tr>
<tr>
<td>5</td>
<td>x9</td>
<td>INTERVAL</td>
</tr>
</tbody>
</table>
**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> ;
   REDUCE UNSUPERVISED effects </ reduce-options > ;
   REDUCE SUPERVISED response . . . < response > = effects </ reduce-options > ;
   FREQ 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 17.1 summarizes the important options in the PROC VARREDUCE statement by function. The options are then described fully in alphabetical order.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic Options</td>
<td>Specifies the input data table</td>
</tr>
<tr>
<td>DATA=</td>
<td></td>
</tr>
<tr>
<td>Options Related to Variable Selection</td>
<td></td>
</tr>
<tr>
<td>NOPRINT</td>
<td>Suppresses ODS output</td>
</tr>
<tr>
<td>MATRIX=</td>
<td>Specifies the matrix to use to select variables</td>
</tr>
<tr>
<td>TECHNIQUE=</td>
<td>Selects the variable selection technique</td>
</tr>
<tr>
<td>OUTCP=</td>
<td>Outputs the CORR, COV, or SSCP matrix, which is specified in the MATRIX= option</td>
</tr>
</tbody>
</table>

**Table 17.1 PROC VARREDUCE Statement Options**

You can specify the following options:

- `DATA=CAS-libref.data-table`
  names the input data table for PROC VARREDUCE to use. The default is the most recently created data table. `CAS-libref.data-table` is a two-level name, where

  - `CAS-libref` refers to a collection of information that is defined in the LIBNAME statement and includes the caslib, which includes a path to the data, and a session identifier, which
Chapter 17: The VARREDUCE Procedure

defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about CAS-libref, see the section “Using CAS Sessions and CAS Engine Librefs” on page 547.

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

\[
\text{Corr}(X, Y) = \frac{E((X - E(X))(Y - E(Y)))}{\sqrt{E(X - E(X))^2 E(Y - E(Y))^2}}
\]

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

\[
\text{Cov}(X, Y) = E((X - E(X))(Y - E(Y)))
\]

SSCP selects variables based on the sums of squares and crossproducts matrix. Assuming that X and Y are two variables and that x and y are their corresponding variable vectors, the SSCP between X and Y is computed by

\[
\text{SSCP}(X, Y) = x^T y
\]

By default, MATRIX=CORR.

NOPRINT suppresses the generation of ODS output.

OUTCP=CAS-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. CAS-libref.data-table is a two-level name, where CAS-libref refers to the caslib and session identifier, 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 547. 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.
**CLASS Statement**

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 563.

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 17.2** 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 17.2**.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DESCENDING</td>
<td>Reverses the sort order</td>
</tr>
<tr>
<td>MISSING</td>
<td>Treats missing values as valid levels</td>
</tr>
</tbody>
</table>

**DESCENDING**

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 ODS 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, all ODS tables are sent to the client and then the client creates a subset. If both DISPLAY and ODS statements are used together, the DISPLAY statement takes precedence over the ODS statements. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

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

**CASESENSITIVE**

performs a case-sensitive comparison of table names in the *table-list* to ODS 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 ODS tables except those specified in the *table-list*.

**EXCLUDEALL**

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

**TRACE**

displays the ODS table names, labels, and paths.

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

A path is a table name that is prefixed with dot-separated grouping information. For example, a SelectionSummary table that is produced by a procedure 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 ODS 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 starts with a “/” or a “!”. 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.
**DISPLAYOUT Statement**

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

The DISPLAYOUT statement enables you to create CAS 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 CAS output tables to create. Each entry in the list has either a `key` or a `key=value` format:

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

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 (/):

- **NOREPLACE**
  does not replace an existing CAS output table of the same name.

- **REPEATED**
  replicates the CAS output tables on all nodes.

**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 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 the FREQ statement is not specified, 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 \textit{effects} and one or more \textit{response-variables}. A \textit{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 \textit{response-variable}, and in a regression context, in which you can specify more than one \textit{response-variable}.

Table 17.3 summarizes the \textit{reduce-options}, which control the number of variables to be selected.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIC</td>
<td>Uses Akaike’s information criterion to perform model selection</td>
</tr>
<tr>
<td>AICC</td>
<td>Uses the corrected Akaike’s information criterion to perform model selection</td>
</tr>
<tr>
<td>BIC</td>
<td>Uses Schwarz Bayesian information criterion to perform model selection</td>
</tr>
<tr>
<td>MAXITER</td>
<td>MAXSTEPS=</td>
</tr>
<tr>
<td>MAXEFFECTS=</td>
<td>Specifies the number of effects to select; the number must be greater than or equal to 1.</td>
</tr>
<tr>
<td>VARIANCEEXPLAINED</td>
<td>VAREXP=</td>
</tr>
<tr>
<td>MINVARIANCEINCREMENT</td>
<td>VARINC=</td>
</tr>
</tbody>
</table>

The \textit{reduce-options} determine the number of variables to be selected. You can specify the following \textit{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 \textit{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 \textit{reduce-options} for both supervised and unsupervised variable selection:
MAXITER=n
MAXSTEPS=n
stops PROC VARREDUCE after it runs n steps.

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 MAXITER= option.

VARIANCEEXPLAINED=fraction
VAREXP=fraction
stops PROC VARREDUCE when the fraction of the total variance can be explained by the selected variables.

MINVARIANCEINCREMENT=fraction
VARIANT=fraction
stops PROC VARREDUCE when the minimum increment of the explained variance is less than fraction of the total variance.

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 \( X \in \mathbb{R}^{n \times m} \) be a data table that contains n samples and m variables; let \( X = (X_1, X_2) \), where \( X_1 \in \mathbb{R}^{n \times k} \) contains the k selected variables and \( X_2 \in \mathbb{R}^{n \times (m-k)} \) contains the remaining \( m-k \) variables. PROC VARREDUCE selects the variables by minimizing the following equation:

\[
\min \operatorname{Trace}\left( X_2^\top \left( I - X_1 (X_1^\top X_1)^{-1} X_1^\top \right) X_2 \right)
\]
\left( I - X_1 \left( X_1^T X_1 \right)^{-1} X_1^T \right)^\frac{1}{2} X_2 \text{ projects } X_2 \text{ to the null space of } X_1. \text{ Therefore, the preceding equation measures the data variance that resides in the null space of } X_1, \text{ which is the data variance that cannot be explained by the variables in } X_1. \text{ Minimizing this equation leads to the selection of the variables that jointly explain the maximum amount of the variance in the original data.}

Let \( C_{11} = X_1^T X_1, \ C_{12} = X_1^T X_2, \) and \( C_{21} = X_2^T X_1. \) The following equations hold:
\[
C = X^T X = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}
\]
\[
X_2^T \left( I - X_1 \left( X_1^T X_1 \right)^{-1} X_1^T \right) X_2 = C_{22} - C_{21} C_{11}^{-1} C_{21}
\]

When all the variables are centralized to have a zero mean, \( C \) is the covariance matrix. This corresponds to specifying \texttt{MATRIX=COV} in the \texttt{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 \texttt{MATRIX=CORR} in the \texttt{PROC VARREDUCE} statement, which requests that the correlation matrix be used for variable selection. If neither centralization nor normalization should be applied, specify \texttt{MATRIX=SSCP} in the \texttt{PROC VARREDUCE} statement.

Principal component analysis (PCA) (Jolliffe 2002) also reduces dimensionality by preserving data variance. The key difference between PCA and \texttt{PROC VARREDUCE} is that PCA generates a small set of new variables (variable extraction) by linearly combining the original variables, whereas \texttt{PROC VARREDUCE} selects a small set of the original variables (variable selection). The variables returned by \texttt{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).

\section*{Supervised Variable Selection}

When response variables are specified in a \texttt{REDUCE} statement, \texttt{PROC VARREDUCE} conducts supervised variable selection, either in a regression context or in a classification (categorization) context.

\section*{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 \texttt{CLASS} statement.

Let \( Y \in \mathbb{R}^{n \times t} \) be the response data that contain \( t \) response variables. Assume that \( k \) variables need to be selected. Let \( X \in \mathbb{R}^{n \times m} \) be a data table that contains \( n \) samples and \( m \) variables; let \( X = (X_1, X_2), \) where \( X_1 \in \mathbb{R}^{n \times k} \) contains the \( k \) selected variables and \( X_2 \in \mathbb{R}^{n \times (m-k)} \) contains the remaining \( m - k \) variables. \texttt{PROC VARREDUCE} selects the variables by minimizing the following equation:
\[
\min \text{Trace} \left( Y^T \left( I - X_1 \left( X_1^T X_1 \right)^{-1} X_1^T \right) Y \right)
\]
\[
\left( I - X_1 \left( X_1^T X_1 \right)^{-1} X_1^T \right)^\frac{1}{2} Y \text{ projects } Y \text{ onto the null space of } X_1. \text{ Therefore, the equation measures the response variance that resides in the null space of } X_1, \text{ which is the variance of the response variables that cannot be explained by the variables in } X_1. \text{ 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 $y$ with $c$ levels $\{1, \ldots, c\}$. Then $y$ can be levelized in a special way to generate response data $Y \in \mathbb{R}^{n \times c}$ as:

$$Y_{i,j} = \begin{cases} \frac{1}{\sqrt{n}} \left( \frac{n}{n_j} - \frac{n_j}{n} \right), & y_i = j \\ -\frac{1}{\sqrt{n}} \frac{n_j}{n}, & y_i \neq j \end{cases}$$

By using this $Y$ 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 17.4 provides formulas and definitions for these fit statistics.

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Definition or Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td>Number of observations</td>
</tr>
<tr>
<td>$p$</td>
<td>Number of parameters</td>
</tr>
<tr>
<td>$t$</td>
<td>Number of response variables</td>
</tr>
<tr>
<td>SSE</td>
<td>Error sum of squares</td>
</tr>
<tr>
<td>MSE</td>
<td>$\frac{SSE}{n-p}$</td>
</tr>
<tr>
<td>AIC</td>
<td>$\ln \left( \frac{SSE}{n} \right) + \frac{2pt + t (t + 1)}{n}$</td>
</tr>
<tr>
<td>AICC</td>
<td>$\ln \left( \frac{SSE}{n} \right) + \frac{(n + p)t}{n - p - t - 1}$</td>
</tr>
<tr>
<td>BIC</td>
<td>$\ln \left( \frac{SSE}{n} \right) + \frac{p \ln(n)}{n}$</td>
</tr>
</tbody>
</table>
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 V ARREDUCE applies \( k \) steps of a greedy search to generate a suboptimal solution for the problem.

Assume that \( q \) features have been selected, that \( X_1 \) contains the \( q \) selected variables, and that \( X_2 \) contains the remaining variables. PROC V ARREDUCE selects the \( q \) variable, \( F \), by minimizing the equation

\[
\arg \min_F \text{Trace} \left( \hat{X}_2^T \left( I - \hat{X}_1 \left( \hat{X}_1^T \hat{X}_1 \right)^{-1} \hat{X}_1^T \right) \hat{X}_2 \right)
\]

where \( \hat{X}_1 \) is the data table that contains the feature \( F \) and the \( q \) selected variables, and \( \hat{X}_2 \) is the data table that contains the remaining variables. Minimizing the preceding problem is equivalent to maximizing the following problem:

\[
\frac{\| X_2^T \left( I - X_1 \left( X_1^T X_1 \right)^{-1} X_1^T \right) f \|_2^2}{\left\| \left( I - X_1 \left( X_1^T X_1 \right)^{-1} X_1^T \right)^{1/2} f \right\|_2^2}
\]

In the preceding equation, \( \| X_2^T \left( I - X_1 \left( X_1^T X_1 \right)^{-1} X_1^T \right) f \|_2^2 \) is the summation of the squares of the covariance between the variable \( f \) and all the unselected variables in the null space of \( X_1 \). And \( \left\| \left( I - X_1 \left( X_1^T X_1 \right)^{-1} X_1^T \right)^{1/2} f \right\|_2^2 \) is the square of the variance of \( f \) in the null space of \( X_1 \), 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 V ARREDUCE. The following problem is maximized for supervised variable selection:

\[
\frac{\| Y^T \left( I - X_1 \left( X_1^T X_1 \right)^{-1} X_1^T \right) f \|_2^2}{\left\| \left( I - X_1 \left( X_1^T X_1 \right)^{-1} X_1^T \right)^{1/2} f \right\|_2^2}
\]

Here, \( Y \) is the response data table. Let \( c \) be the number of columns in \( Y \). The time complexity for selecting \( k \) variables by solving the preceding problem is

\[ O \left( k^2 (c + k) m + m^2 n \right) \]

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

\[
\text{CPU} \left( \frac{m^2(n + k^2)}{p} + m^2 \log p \right) + \text{NET} \left( m^2 \log p \right)
\]

and the time complexity for supervised variable selection is

\[
\text{CPU} \left( \frac{k^2(c + k)m + m^2n}{p} + m^2 \log p \right) + \text{NET} \left( m^2 \log p \right)
\]

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 17.5 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 17.5 Available Types of Effects**

<table>
<thead>
<tr>
<th>Effect</th>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intercept</td>
<td>Default</td>
<td>Intercept (unless the NOINT option is specified)</td>
</tr>
<tr>
<td>Regression</td>
<td>( X Z )</td>
<td>Continuous variables</td>
</tr>
<tr>
<td>Polynomial</td>
<td>( X^2Z )</td>
<td>Interaction of continuous variables</td>
</tr>
<tr>
<td>Main</td>
<td>( A B )</td>
<td>CLASS variables</td>
</tr>
<tr>
<td>Interaction</td>
<td>( A^2B )</td>
<td>Crossing of CLASS variables</td>
</tr>
<tr>
<td>Nested</td>
<td>( A(B) )</td>
<td>Main effect ( A ) nested within CLASS effect ( B )</td>
</tr>
<tr>
<td>Continuous-by-class</td>
<td>( X^2A )</td>
<td>Crossing of continuous and CLASS variables</td>
</tr>
<tr>
<td>Continuous-nesting-class</td>
<td>( X(A) )</td>
<td>Continuous variable ( X1 ) nested within CLASS variable ( A )</td>
</tr>
<tr>
<td>General</td>
<td>( X^2Z^2A(B) )</td>
<td>Combinations of different types of effects</td>
</tr>
</tbody>
</table>

Table 17.6 shows some examples of MODEL statements that use various types of effects.

**Table 17.6 Model Statement Effect Examples**

<table>
<thead>
<tr>
<th>Specification</th>
<th>Type of Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>model ( Y=X );</td>
<td>Simple regression</td>
</tr>
<tr>
<td>model ( Y=X \ Z; )</td>
<td>Multiple regression</td>
</tr>
<tr>
<td>model ( Y=X \ X^2X; )</td>
<td>Polynomial regression</td>
</tr>
<tr>
<td>model ( Y=A );</td>
<td>One-way analysis of variance (ANOVA)</td>
</tr>
<tr>
<td>model ( Y=A \ B \ C; )</td>
<td>Main-effects ANOVA</td>
</tr>
<tr>
<td>model ( Y=A \ B \ A^2B; )</td>
<td>Factorial ANOVA with interaction</td>
</tr>
<tr>
<td>model ( y=A \ B(A) \ C(B \ A); )</td>
<td>Nested ANOVA</td>
</tr>
</tbody>
</table>
Table 17.6 continued

<table>
<thead>
<tr>
<th>Specification</th>
<th>Type of Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>model Y=A X;</td>
<td>Analysis of covariance (ANCOVA)</td>
</tr>
<tr>
<td>model Y=A X(A);</td>
<td>Separate-slopes regression</td>
</tr>
<tr>
<td>model Y=A X X*A;</td>
<td>Homogeneity-of-slopes regression</td>
</tr>
</tbody>
</table>

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 $X X X X X$, 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 17.7 is an example where $\beta_0$ denotes the intercept and $A$ and $B$ are classification variables that have two and three levels, respectively.

Table 17.7 Example of Main Effects

<table>
<thead>
<tr>
<th>Data</th>
<th>I</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>$\beta_0$</td>
<td>A1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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 17.8).

Table 17.8  Example of Interaction Effects

<table>
<thead>
<tr>
<th>Data</th>
<th>I</th>
<th>A</th>
<th>B</th>
<th>A*B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>β₀</td>
<td>A₁</td>
<td>A₂</td>
<td>B₁</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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):

```plaintext
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 17.9).

Table 17.9  Example of Nested Effects

<table>
<thead>
<tr>
<th>Data</th>
<th>I</th>
<th>A</th>
<th>B(A)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>β₀</td>
<td>A₁</td>
<td>A₂</td>
</tr>
<tr>
<td>A</td>
<td>1</td>
<td>B₁</td>
<td>B₂A₁</td>
</tr>
<tr>
<td>A</td>
<td>2</td>
<td>B₂</td>
<td>B₂A₂</td>
</tr>
<tr>
<td>A</td>
<td>3</td>
<td>B₃</td>
<td>B₂A₃</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>B₁</td>
<td>B₁A₂</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>B₂</td>
<td>B₂A₂</td>
</tr>
<tr>
<td>B</td>
<td>3</td>
<td>B₃</td>
<td>B₂A₃</td>
</tr>
</tbody>
</table>
Table 17.9  continued

<table>
<thead>
<tr>
<th>Data</th>
<th>I</th>
<th>A</th>
<th>B(A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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 17.10).

Table 17.10  Example of Continuous-Nesting-Class Effects

<table>
<thead>
<tr>
<th>Data</th>
<th>I</th>
<th>A</th>
<th>β₀</th>
<th>A₁</th>
<th>A₂</th>
<th>X(A₁)</th>
<th>X(A₂)</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>A</td>
<td>β₀</td>
<td>A₁</td>
<td>A₂</td>
<td>X(A₁)</td>
<td>X(A₂)</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>21</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>24</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>22</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>23</td>
<td></td>
</tr>
</tbody>
</table>

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 17.11 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 17.10.

Table 17.11  Example of Continuous-by-Class Effects

<table>
<thead>
<tr>
<th>Data</th>
<th>I</th>
<th>X</th>
<th>β₀</th>
<th>A</th>
<th>X*A</th>
<th>A₁</th>
<th>A₂</th>
<th>X*A₁</th>
<th>X*A₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>A</td>
<td>X</td>
<td>β₀</td>
<td>A</td>
<td>X*A</td>
<td>A₁</td>
<td>A₂</td>
<td>X*A₁</td>
<td>X*A₂</td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>21</td>
<td>1</td>
<td>0</td>
<td>21</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>1</td>
<td>24</td>
<td>1</td>
<td>0</td>
<td>24</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>1</td>
<td>22</td>
<td>1</td>
<td>0</td>
<td>22</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>2</td>
<td>28</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>28</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>2</td>
<td>19</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>19</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>2</td>
<td>23</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>23</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 17.10).

Table 17.10  Example of Continuous-Nesting-Class Effects

<table>
<thead>
<tr>
<th>Data</th>
<th>I</th>
<th>A</th>
<th>β₀</th>
<th>A₁</th>
<th>A₂</th>
<th>X(A₁)</th>
<th>X(A₂)</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>A</td>
<td>β₀</td>
<td>A₁</td>
<td>A₂</td>
<td>X(A₁)</td>
<td>X(A₂)</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>21</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>24</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>22</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>23</td>
<td></td>
</tr>
</tbody>
</table>

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

```plaintext
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:

```
A1B1C1D1  A1B2C1D1  A2B1C1D1  A2B2C1D1
A1B1C2D1  A1B2C2D1  A2B1C2D1  A2B2C2D1
```

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 17.12.

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Statement</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>NObs</td>
<td>Number of observations read and used; number of events and trials, if applicable</td>
<td>PROC VARREDUCE</td>
<td>Default</td>
</tr>
<tr>
<td>SelectedEffects</td>
<td>Summary of selected variables</td>
<td>PROC VARREDUCE</td>
<td>Default</td>
</tr>
<tr>
<td>SelectionSummary</td>
<td>Selection summary</td>
<td>PROC VARREDUCE</td>
<td>Default</td>
</tr>
</tbody>
</table>

Examples: VARREDUCE 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 8 and “Loading a SAS Data Set onto a CAS Server” on page 9 in Chapter 2, “Shared Concepts.”

Example 17.1: Analyzing Framingham Heart Study Data

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 naming your CAS engine libref in the first statement of the following DATA step:

```plaintext
data mycas.heart;
    set sashelp.heart;
run;
```

These statements assume that your CAS engine libref is named mycas, but you can substitute any appropriately defined CAS engine libref.

The following statements invoke the VARREDUCE procedure:

```plaintext
data mycas.heart;
    set sashelp.heart;
run;
```

```plaintext
proc varreduce data=mycas.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;
```

```plaintext
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 17.1.1 and Output 17.1.2 display the selection summary from each iteration and the selected effects.

**Output 17.1.1** Selection Summary

**The VARREDUCE Procedure**

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Parameter</th>
<th>Selection Summary</th>
<th>Proportion of Variance Explained</th>
<th>SSE</th>
<th>MSE</th>
<th>AIC</th>
<th>AICC</th>
<th>BIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>AgeAtStart</td>
<td></td>
<td>0.191310</td>
<td>0.806690</td>
<td>0.00016052</td>
<td>-0.210355</td>
<td>1.789647</td>
<td>-0.210648</td>
</tr>
<tr>
<td>2</td>
<td>Smoking</td>
<td></td>
<td>0.220923</td>
<td>0.779077</td>
<td>0.00015467</td>
<td>-0.246867</td>
<td>1.753136</td>
<td>-0.246262</td>
</tr>
<tr>
<td>3</td>
<td>Systolic</td>
<td></td>
<td>0.245405</td>
<td>0.754595</td>
<td>0.00014984</td>
<td>-0.278002</td>
<td>1.722002</td>
<td>-0.276499</td>
</tr>
<tr>
<td>4</td>
<td>Sex Female</td>
<td></td>
<td>0.255479</td>
<td>0.744521</td>
<td>0.00014787</td>
<td>-0.290648</td>
<td>1.709358</td>
<td>-0.288247</td>
</tr>
<tr>
<td>5</td>
<td>Weight_Status Underweight</td>
<td></td>
<td>0.256772</td>
<td>0.743228</td>
<td>0.00014764</td>
<td>-0.291593</td>
<td>1.708415</td>
<td>-0.288294</td>
</tr>
<tr>
<td>6</td>
<td>BP_Status Normal</td>
<td></td>
<td>0.257880</td>
<td>0.742120</td>
<td>0.00014745</td>
<td>-0.292291</td>
<td>1.707720</td>
<td>-0.288093</td>
</tr>
<tr>
<td>7</td>
<td>Smoking_Status Non-smoker</td>
<td></td>
<td>0.258962</td>
<td>0.741038</td>
<td>0.00014727</td>
<td>-0.292957</td>
<td>1.707057</td>
<td>-0.287861</td>
</tr>
<tr>
<td>8</td>
<td>Chol_Status Borderline</td>
<td></td>
<td>0.259547</td>
<td>0.740453</td>
<td>0.00014718</td>
<td>-0.292952</td>
<td>1.707065</td>
<td>-0.286958</td>
</tr>
</tbody>
</table>
### Output 17.1.2 Selected Effects

<table>
<thead>
<tr>
<th>Selected Effects</th>
<th>Selected Number</th>
<th>Variable Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>AgeAtStart</td>
<td>1</td>
<td>INTERVAL</td>
</tr>
<tr>
<td>Smoking</td>
<td>2</td>
<td>INTERVAL</td>
</tr>
<tr>
<td>Systolic</td>
<td>3</td>
<td>INTERVAL</td>
</tr>
<tr>
<td>Sex</td>
<td>4</td>
<td>CLASS</td>
</tr>
<tr>
<td>Weight_Status</td>
<td>5</td>
<td>CLASS</td>
</tr>
<tr>
<td>BP_Status</td>
<td>6</td>
<td>CLASS</td>
</tr>
<tr>
<td>Smoking_Status</td>
<td>7</td>
<td>CLASS</td>
</tr>
<tr>
<td>Chol_Status</td>
<td>8</td>
<td>CLASS</td>
</tr>
</tbody>
</table>

Output 17.1.3 shows the BIC curve change throughout iterations.

### Output 17.1.3 BIC curve

---

**Example 17.2: Output a Correlation Matrix to a SAS Data File**

This example shows how to output a correlation matrix to a SAS data file. The `OUTCP=` option creates an output data table named `mycas.corr`.

The following DATA step generates a data table that has 2,000 observations and contains both interval variables ($x_1$–$x_2$) and CLASS variables ($a$, $c_1$, and $c_2$):

```sas
data mycas.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;
```

---
These statements assume that your CAS engine libref is named mycas, but you can substitute any appropriately defined CAS engine libref.

The following statements invoke the VARREDUCE procedure:

```plaintext
   title "Output the Correlation Matrix";

   proc varreduce data=mycas.data1 matrix=corr outcp=mycas.corr;
     class a;
     reduce unsupervised a x1-x2 /maxsteps=4;
   run;

   proc print data=mycas.corr;
   run;
```

Output 17.2.1 shows the content of the data file that PROC VARREDUCE generates.

**Output 17.2.1 Output the Correlation Matrix**

<table>
<thead>
<tr>
<th>Obs</th>
<th>ID</th>
<th>TYPE</th>
<th><em>VAR</em></th>
<th><em>vID</em></th>
<th>v1</th>
<th>v2</th>
<th>v3</th>
<th>v4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>MEAN/FREQ</td>
<td></td>
<td></td>
<td>1.00</td>
<td>-1.00</td>
<td>-0.00</td>
<td>-0.02</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>N</td>
<td></td>
<td></td>
<td>2000.00</td>
<td>2000.00</td>
<td>2000.00</td>
<td>2000.00</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>CORR</td>
<td>a0</td>
<td>v1</td>
<td>-1.00</td>
<td>1.00</td>
<td>0.00</td>
<td>0.02</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>CORR</td>
<td>a1</td>
<td>v2</td>
<td>-1.00</td>
<td>1.00</td>
<td>0.00</td>
<td>0.02</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>CORR</td>
<td>x1</td>
<td>v3</td>
<td>-0.00</td>
<td>0.00</td>
<td>1.00</td>
<td>-0.01</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>CORR</td>
<td>x2</td>
<td>v4</td>
<td>-0.02</td>
<td>0.02</td>
<td>-0.01</td>
<td>1.00</td>
</tr>
</tbody>
</table>

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, \( v_1 \) 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 \( 4 \times 4 \), and it resides in the table in rows 3–6 and columns 7–10.

---

**Example 17.3: Output the Correlation Matrix in LIL Format**

This example shows how to output a correlation matrix in list-of-list (LIL) format. The OUTCP= option creates an output data table named mycas.corr_lil.

The following DATA step generates a data table that has 2,000 observations and contains both interval variables (\( x_1-x_2 \)) and a CLASS variable (a):

```plaintext
   output;
   end;
run;
```
data mycas.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 mycas, 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=mycas.data2 matrix=corr outcp=mycas.corr_lil/list eps=0.01;
      class a;
      reduce unsupervised a x1-x2 /maxsteps=4;
    run;

    proc print data=mycas.corr_lil;
    run;

Output 17.3.1 shows the correlation matrix in LIL format.

Output 17.3.1 Output the Correlation Matrix in LIL Format

Output the Correlation Matrix in LIL Format

<table>
<thead>
<tr>
<th>Obs</th>
<th>TYPE</th>
<th>ID</th>
<th>NAME1</th>
<th>NAME2</th>
<th>VAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>S</td>
<td>1</td>
<td>samples</td>
<td></td>
<td>2000.00</td>
</tr>
<tr>
<td>2</td>
<td>S</td>
<td>2</td>
<td>nVar</td>
<td></td>
<td>3.00</td>
</tr>
<tr>
<td>3</td>
<td>S</td>
<td>3</td>
<td>nEff</td>
<td></td>
<td>4.00</td>
</tr>
<tr>
<td>4</td>
<td>F</td>
<td>1</td>
<td>a</td>
<td>0</td>
<td>979.00</td>
</tr>
<tr>
<td>5</td>
<td>F</td>
<td>2</td>
<td>a</td>
<td>1</td>
<td>1021.00</td>
</tr>
<tr>
<td>6</td>
<td>M</td>
<td>3</td>
<td>x1</td>
<td></td>
<td>0.49</td>
</tr>
<tr>
<td>7</td>
<td>M</td>
<td>4</td>
<td>x2</td>
<td></td>
<td>0.50</td>
</tr>
<tr>
<td>8</td>
<td>R</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1.00</td>
</tr>
<tr>
<td>9</td>
<td>R</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>-1.00</td>
</tr>
<tr>
<td>10</td>
<td>R</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>1.00</td>
</tr>
<tr>
<td>11</td>
<td>R</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>0.03</td>
</tr>
<tr>
<td>12</td>
<td>R</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>-0.03</td>
</tr>
<tr>
<td>13</td>
<td>R</td>
<td>6</td>
<td>3</td>
<td>3</td>
<td>1.00</td>
</tr>
<tr>
<td>14</td>
<td>R</td>
<td>10</td>
<td>4</td>
<td>4</td>
<td>1.00</td>
</tr>
</tbody>
</table>

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**


Subject Index

ABC parameters
  KCLUS procedure, 155
ABC statistics
  KCLUS procedure, 156
adaptive LASSO selection
  Shared Concepts, 63
ANCOVA effects
  Shared Concepts, 49
  VARREDUCE procedure, 564
ANOVA effects
  Shared Concepts, 48
  VARREDUCE procedure, 563
ANOVA table
  REGSELECT procedure, 401
ASSESS procedure, 467
  displayed output, 476
  fit statistics, 475, 476
  input data tables, 471
  input variables, 473
  lift information, 474, 476
  lift regression information, 476
  ODS table names, 477
  ROC information, 475, 476
  target variables, 474
at sign (@) operator
  Shared Concepts, 47, 50, 565
B-spline
  spline basis (Shared Concepts), 29
B-spline basis
  GENSELECT procedure, 29
  KCLUS procedure, 29
  LOGSELECT procedure, 29
  NLMOD procedure, 29
  PCA procedure, 29
  REGSELECT procedure, 29
  TREESPLIT procedure, 29
backward elimination
  Shared Concepts, 58
bar (|) operator
  Shared Concepts, 46, 49, 50, 565
  VARREDUCE procedure, 564
Bernoulli distribution
  NLMOD procedure, 237
bin details
  BINNING procedure, 494
binary distribution
  NLMOD procedure, 237
BINNING procedure
  bin details, 494
  computation, 491
  displayed output, 494
  formula, 491
  information value, 493
  input data tables, 489
  ODS table names, 494
  output data tables, 491
  trimmed statistics, 494
  variable information value (IV) table, 494
  Winsorized statistics, 494
binomial distribution
  NLMOD procedure, 237
boundary constraints, 234
bounds
  NLMOD procedure, 234
C4.5 pruning
  TREESPLIT procedure, 435, 444
candidates for addition or removal
  GENSELECT procedure, 125
  LOGSELECT procedure, 208
  QTRSELECT procedure, 361
  REGSELECT procedure, 400
CARDINALITY procedure, 503
  displayed output, 510
  input data tables, 508
centering and scaling information
  PCA procedure, 277
  PLSMOD procedure, 318
class level
  GENSELECT procedure, 88, 124
  LOGSELECT procedure, 180, 207
  PLSMOD procedure, 303, 318
  QTRSELECT procedure, 340, 360
  REGSELECT procedure, 382, 400
CLASS statement
  Shared Concepts, 43
  syntax (Shared Concepts), 10, 43
classification level
  PARTITION procedure, 528
classification trees, see decision trees
classification variables
  Shared Concepts, 43
cluster summary
  KCLUS procedure, 153
cluster summary for nominal variables
<table>
<thead>
<tr>
<th>Subject Index</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>KCLUS procedure</strong>, 154</td>
</tr>
<tr>
<td><strong>CODE statement</strong></td>
</tr>
<tr>
<td>syntax (Shared Concepts), 14</td>
</tr>
<tr>
<td><strong>colon (:) operator</strong></td>
</tr>
<tr>
<td>Shared Concepts, 47</td>
</tr>
<tr>
<td><strong>complete separation</strong></td>
</tr>
<tr>
<td>GENSELECT procedure, 116</td>
</tr>
<tr>
<td>LOGSELECT procedure, 198</td>
</tr>
<tr>
<td><strong>components</strong></td>
</tr>
<tr>
<td>PLSMOD procedure, 288</td>
</tr>
<tr>
<td><strong>computation</strong></td>
</tr>
<tr>
<td>BINNING procedure, 491</td>
</tr>
<tr>
<td><strong>computational method</strong></td>
</tr>
<tr>
<td>GENSELECT procedure, 68</td>
</tr>
<tr>
<td>KCLUS procedure, 68</td>
</tr>
<tr>
<td>LOGSELECT procedure, 68</td>
</tr>
<tr>
<td>NLMOD procedure, 68</td>
</tr>
<tr>
<td>PCA procedure, 68</td>
</tr>
<tr>
<td>QTRSELECT procedure, 360</td>
</tr>
<tr>
<td>REGSELECT procedure, 68, 398</td>
</tr>
<tr>
<td>TREESPLIT procedure, 68</td>
</tr>
<tr>
<td>VARREDUCE procedure, 562</td>
</tr>
<tr>
<td><strong>confidence limits</strong></td>
</tr>
<tr>
<td>model parameters (GENSELECT), 95, 96</td>
</tr>
<tr>
<td>model parameters (LOGSELECT), 187</td>
</tr>
<tr>
<td>model parameters (QTRSELECT), 345</td>
</tr>
<tr>
<td>model parameters (REGSELECT), 388</td>
</tr>
<tr>
<td><strong>continuous-by-class effects</strong></td>
</tr>
<tr>
<td>Shared Concepts, 51</td>
</tr>
<tr>
<td>VARREDUCE procedure, 566</td>
</tr>
<tr>
<td><strong>continuous-nesting-class effects</strong></td>
</tr>
<tr>
<td>Shared Concepts, 51</td>
</tr>
<tr>
<td>VARREDUCE procedure, 566</td>
</tr>
<tr>
<td><strong>convergence criterion</strong></td>
</tr>
<tr>
<td>GENSELECT procedure, 38–42, 87</td>
</tr>
<tr>
<td>KCLUS procedure, 38–42</td>
</tr>
<tr>
<td>LOGSELECT procedure, 38–42, 178</td>
</tr>
<tr>
<td>NLMOD procedure, 38–42</td>
</tr>
<tr>
<td>PCA procedure, 38–42</td>
</tr>
<tr>
<td>REGSELECT procedure, 38–42</td>
</tr>
<tr>
<td>Shared Concepts, 38–42</td>
</tr>
<tr>
<td>TREESPLIT procedure, 38–42</td>
</tr>
<tr>
<td><strong>convergence status</strong></td>
</tr>
<tr>
<td>GENSELECT procedure, 124</td>
</tr>
<tr>
<td>LOGSELECT procedure, 207</td>
</tr>
<tr>
<td><strong>correlation</strong></td>
</tr>
<tr>
<td>PCA procedure, 278</td>
</tr>
<tr>
<td>principal components, 276, 278</td>
</tr>
<tr>
<td><strong>correlation matrix</strong></td>
</tr>
<tr>
<td>GENSELECT procedure, 87, 126</td>
</tr>
<tr>
<td>LOGSELECT procedure, 179, 209</td>
</tr>
<tr>
<td><strong>cost-complexity pruning</strong></td>
</tr>
<tr>
<td>TREESPLIT procedure, 435, 443</td>
</tr>
<tr>
<td><strong>covariance</strong></td>
</tr>
<tr>
<td>PCA procedure, 278</td>
</tr>
<tr>
<td>principal components, 276, 278</td>
</tr>
<tr>
<td><strong>covariance matrix</strong></td>
</tr>
<tr>
<td>GENSELECT procedure, 87, 126</td>
</tr>
<tr>
<td>LOGSELECT procedure, 179, 209</td>
</tr>
<tr>
<td><strong>crossed effects</strong></td>
</tr>
<tr>
<td>Shared Concepts, 50</td>
</tr>
<tr>
<td>VARREDUCE procedure, 565</td>
</tr>
<tr>
<td><strong>dash (-) operator</strong></td>
</tr>
<tr>
<td>Shared Concepts, 47</td>
</tr>
<tr>
<td><strong>decision trees</strong></td>
</tr>
<tr>
<td>TREESPLIT procedure, 414, 436</td>
</tr>
<tr>
<td><strong>descriptive statistics</strong></td>
</tr>
<tr>
<td>KCLUS procedure, 154</td>
</tr>
<tr>
<td><strong>diagnostic statistics</strong></td>
</tr>
<tr>
<td>QTRSELECT procedure, 355</td>
</tr>
<tr>
<td>REGSELECT procedure, 395</td>
</tr>
<tr>
<td><strong>dimensions</strong></td>
</tr>
<tr>
<td>GENSELECT procedure, 125</td>
</tr>
<tr>
<td>LOGSELECT procedure, 208</td>
</tr>
<tr>
<td>PLSMOD procedure, 317</td>
</tr>
<tr>
<td>QTRSELECT procedure, 361</td>
</tr>
<tr>
<td>REGSELECT procedure, 400</td>
</tr>
<tr>
<td><strong>DISPLAY statement</strong></td>
</tr>
<tr>
<td>syntax (Shared Concepts), 15</td>
</tr>
<tr>
<td><strong>displayed output</strong></td>
</tr>
<tr>
<td>ASSESS procedure, 476</td>
</tr>
<tr>
<td>BINNING procedure, 494</td>
</tr>
<tr>
<td>CARDINALITY procedure, 510</td>
</tr>
<tr>
<td>GENSELECT procedure, 123</td>
</tr>
<tr>
<td>KCLUS procedure, 153</td>
</tr>
<tr>
<td>LOGSELECT procedure, 206</td>
</tr>
<tr>
<td>NLMOD procedure, 246</td>
</tr>
<tr>
<td>PARTITION procedure, 528</td>
</tr>
<tr>
<td>PCA procedure, 277</td>
</tr>
<tr>
<td>PLSMOD procedure, 317</td>
</tr>
<tr>
<td>QTRSELECT procedure, 360</td>
</tr>
<tr>
<td>REGSELECT procedure, 399</td>
</tr>
<tr>
<td>VARIMPUTE procedure, 540</td>
</tr>
<tr>
<td>VARREDUCE procedure, 567</td>
</tr>
<tr>
<td><strong>DISPLAYOUT statement</strong></td>
</tr>
<tr>
<td>syntax (Shared Concepts), 16</td>
</tr>
<tr>
<td><strong>distribution function</strong></td>
</tr>
<tr>
<td>GENSELECT procedure, 95</td>
</tr>
<tr>
<td><strong>distribution functions</strong></td>
</tr>
<tr>
<td>NLMOD procedure, 244</td>
</tr>
<tr>
<td><strong>double dash (- -) operator</strong></td>
</tr>
<tr>
<td>Shared Concepts, 48</td>
</tr>
<tr>
<td><strong>dummy parameterization</strong></td>
</tr>
<tr>
<td>Shared Concepts, 48</td>
</tr>
<tr>
<td>VARREDUCE procedure, 563</td>
</tr>
<tr>
<td><strong>effect parameterization</strong></td>
</tr>
</tbody>
</table>
Subject Index

Shared Concepts, 53

**EFFECT statement**
- collection effect (Shared Concepts), 19
- multimember effect (Shared Concepts), 19
- polynomial effect (Shared Concepts), 21
- spline effect (Shared Concepts), 24
- syntax (Shared Concepts), 17

**eigenvalue decomposition method**
- PCA procedure, 257, 274

**eigenvalues**
- PCA procedure, 278

**eigenvalues and eigenvectors**
- PCA procedure, 274, 276, 278

**eigenvectors**
- PCA procedure, 278

**empirical Bayes estimate**
- NLMOD procedure, 240

**estimated number of clusters**
- KCLUS procedure, 156

**examples, GENSELECT**
- multimember effect, 20

**examples, KCLUS**
- multimember effect, 20

**examples, LOGSELECT**
- multimember effect, 20

**examples, NLMOD**
- multimember effect, 20

**examples, NLMOD procedure**
- boundary specification, 234
- conditional model expression, 251
- enzyme data, 228
- gamma distribution, 245
- join point, 251
- plateau model, 251
- predicted values, 251
- segmented model, 251
- starting values, data table, 239
- starting values, grid, 238

**examples, PCA**
- multimember effect, 20

**examples, REGSELECT**
- multimember effect, 20

**examples, TREESPLIT**
- multimember effect, 20

**explained variation of variables**
- PCA procedure, 277

**factors**
- PLSMOD procedure, 288

**finding the number of clusters**
- KCLUS procedure, 150

**fit criteria**
- QTRSELECT procedure, 352
- REGSELECT procedure, 393

**VARREDUCE procedure, 561**

**fit statistics**
- ASSESS procedure, 475, 476
- GENSELECT procedure, 126
- LOGSELECT procedure, 209
- QTRSELECT procedure, 362
- REGSELECT procedure, 402

**formula**
- BINNING procedure, 491

**forward selection**
- Shared Concepts, 56

**forward swap selection**
- Shared Concepts, 61

**frequencies and descriptive statistics for nominal variables**
- KCLUS procedure, 155

**frequency information table**
- PARTITION procedure, 528

**function-based convergence criteria**
- GENSELECT procedure, 38, 39
- KCLUS procedure, 38, 39
- LOGSELECT procedure, 38, 39
- NLMOD procedure, 38, 39
- PCA procedure, 38, 39
- REGSELECT procedure, 38, 39
- TREESPLIT procedure, 38, 39

**gamma distribution**
- NLMOD procedure, 237

**Gaussian distribution**
- NLMOD procedure, 236

**general distribution**
- NLMOD procedure, 237

**general effects**
- Shared Concepts, 52
- VARREDUCE procedure, 567

**GENSELECT procedure, 78, 96**
- B-spline basis, 29
- candidates for addition or removal, 125
- class level, 88, 124
- complete separation, 116
- computational method, 68
- confidence limits, 95
- convergence criterion, 38–42, 87
- convergence status, 124
- correlation matrix, 87, 126
- covariance matrix, 87, 126
- dimensions, 125
- displayed output, 123
- distribution function, 95
- existence of MLEs, 116
- fit statistics, 126
- function-based convergence criteria, 38, 39
gradient-based convergence criteria, 38, 40, 41
infinite parameter estimates, 116
input data tables, 87
iteration history, 124
LASSO method, 117
LASSO selection, 117
link function, 96
maximum likelihood estimates, 116
model information, 123
model options summary, 93
multithreading, 68, 122
natural cubic spline basis, 31
number of observations, 123
ODS table names, 31, 127
optimization algorithms, 123
optimization techniques, 42, 68
output CAS tables, 126
output data tables, 98
parameter estimates, 126
parameter-based convergence criteria, 39, 42
quasi-complete separation, 116
response level ordering, 93
response profile, 124
response variable options, 93
selected effects, 125
selection information, 124
selection reason, 125
selection summary, 125
separation, 116
splines and spline bases, 27
stop reason, 125
test data, 66
truncated power function (TPF) basis, 28
validation, 66
GLM parameterization
Shared Concepts, 48, 563
global test
LOGSELECT procedure, 208
gradient-based convergence criteria
GENSELECT procedure, 38, 40, 41
KCLUS procedure, 38, 40, 41
LOGSELECT procedure, 38, 40, 41
NLMOD procedure, 38, 40, 41
PCA procedure, 38, 40, 41
REGSELECT procedure, 38, 40, 41
Shared Concepts, 38, 40, 41
TREESPLIT procedure, 38, 40, 41
group LASSO selection
Shared Concepts, 63
imputation information table
VARIMPUTE procedure, 541
imputation requests table
VARIMPUTE procedure, 541
infinite parameter estimates
GENSELECT procedure, 116
LOGSELECT procedure, 198
information criteria
QTRSELECT procedure, 353
information value
BINNING procedure, 493
initial values
NLMOD procedure, 237
input variables
ASSESS procedure, 473
interaction effects
Shared Concepts, 50
VARREDUCE procedure, 565
intercept
Shared Concepts, 49
VARREDUCE procedure, 564
iteration history
GENSELECT procedure, 124
KCLUS procedure, 154
LOGSELECT procedure, 207
ITERGS method
PCA procedure, 257, 274
KCLUS procedure, 136
ABC parameters, 155
ABC statistics, 156
B-spline basis, 29
cluster centroid output data tables, 145
cluster summary, 153
cluster summary for nominal variables, 154
computational method, 68
convergence criterion, 38–42
descriptive statistics, 154
displayed output, 153
estimated number of clusters, 156
finding the number of clusters, 150
frequencies and descriptive statistics for nominal
variables, 155
function-based convergence criteria, 38, 39
gradient-based convergence criteria, 38, 40, 41
input data tables, 142
iteration history, 154
missing values, 149
model information, 153
multithreading, 68
natural cubic spline basis, 31
number of observations, 153
ODS table names, 31, 156
optimization techniques, 42, 68
output CAS tables, 156
output data tables, 149
parameter-based convergence criteria, 39, 42
splines and spline bases, 27
standardization, 155
test data, 66
truncated power function (TPF) basis, 28
validation, 66
within-cluster statistics, 154

lag functionality
NLMOD procedure, 242
LAR selection
Shared Concepts, 61
LASSO method
GENSELECT procedure, 117
LASSO selection
GENSELECT procedure, 117
LOGSELECT procedure, 199
Shared Concepts, 62
latent variables
PLSMOD procedure, 288
latent vectors
PLSMOD procedure, 288
least squares
NLMOD procedure, 243
least squares distribution
NLMOD procedure, 236
levelization
Shared Concepts, 43
lift information
ASSESS procedure, 474, 476
lift regression information
ASSESS procedure, 476
link function
GENSELECT procedure, 96
LOGSELECT procedure, 187
loadings
PCA procedure, 274, 278
log-likelihood functions
NLMOD procedure, 244
LOGSELECT procedure, 168, 187
B-spline basis, 29
candidates for addition or removal, 208
class level, 180, 207
complete separation, 198
computational method, 68
confidence limits, 187
convergence criterion, 38–42, 178
convergence status, 207
correlation matrix, 179, 209
covariance matrix, 179, 209
dimensions, 208
displayed output, 206
existence of MLEs, 198
fit statistics, 209
function-based convergence criteria, 38, 39
global test, 208
gradient-based convergence criteria, 38, 40, 41
infinite parameter estimates, 198
input data tables, 179
iteration history, 207
LASSO selection, 199
link function, 187
maximum likelihood estimates, 198
model information, 206
model options summary, 185
multithreading, 68, 205
natural cubic spline basis, 31
number of observations, 206
ODS table names, 31, 210
optimization algorithms, 205
optimization techniques, 42, 68
output CAS tables, 210
output data tables, 189, 389
parameter estimates, 209
parameter-based convergence criteria, 39, 42
quasi-complete separation, 198
response level ordering, 185
response profile, 206
response variable options, 185
selected effects, 208
selection information, 207
selection reason, 208
selection summary, 208
separation, 198
splines and spline bases, 27
stop reason, 208
test data, 66
truncated power function (TPF) basis, 28
Type III tables, 209
validation, 66

main effects
Shared Concepts, 49
VARREDUCE procedure, 564
maximum likelihood estimates
GENSELECT procedure, 116
LOGSELECT procedure, 198
missing values
KCLUS procedure, 149
PCA procedure, 275
PLSMOD procedure, 317
VARREDUCE procedure, 559
model
information (GENSELECT), 123
model details
PLSMOD procedure, 318
model information
KCLUS procedure, 153
LOGSELECT procedure, 206
PCA procedure, 277
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLSMOD procedure</td>
<td>317</td>
</tr>
<tr>
<td>model specification</td>
<td></td>
</tr>
<tr>
<td>NLMOD procedure</td>
<td>236</td>
</tr>
<tr>
<td>multithreading</td>
<td></td>
</tr>
<tr>
<td>GENSELECT procedure</td>
<td>68, 122</td>
</tr>
<tr>
<td>KCLUS procedure</td>
<td>68</td>
</tr>
<tr>
<td>LOGSELECT procedure</td>
<td>68, 205</td>
</tr>
<tr>
<td>NLMOD procedure</td>
<td>68, 246</td>
</tr>
<tr>
<td>PCA procedure</td>
<td>68</td>
</tr>
<tr>
<td>QTRSELECT procedure</td>
<td>360</td>
</tr>
<tr>
<td>REGSELECT procedure</td>
<td>68, 398</td>
</tr>
<tr>
<td>TREESPLIT procedure</td>
<td>68</td>
</tr>
<tr>
<td>natural cubic spline basis</td>
<td></td>
</tr>
<tr>
<td>GENSELECT procedure</td>
<td>31</td>
</tr>
<tr>
<td>KCLUS procedure</td>
<td>31</td>
</tr>
<tr>
<td>LOGSELECT procedure</td>
<td>31</td>
</tr>
<tr>
<td>NLMOD procedure</td>
<td>31</td>
</tr>
<tr>
<td>PCA procedure</td>
<td>31</td>
</tr>
<tr>
<td>REGSELECT procedure</td>
<td>31</td>
</tr>
<tr>
<td>Shared Concepts</td>
<td>31</td>
</tr>
<tr>
<td>TREESPLIT procedure</td>
<td>31</td>
</tr>
<tr>
<td>negative binomial distribution</td>
<td>237</td>
</tr>
<tr>
<td>nested effects</td>
<td></td>
</tr>
<tr>
<td>Shared Concepts</td>
<td>50</td>
</tr>
<tr>
<td>VARREDUCE procedure</td>
<td>565</td>
</tr>
<tr>
<td>nested versus crossed effects</td>
<td></td>
</tr>
<tr>
<td>Shared Concepts</td>
<td>50</td>
</tr>
<tr>
<td>VARREDUCE procedure</td>
<td>565</td>
</tr>
<tr>
<td>NIPALS method</td>
<td></td>
</tr>
<tr>
<td>PCA procedure</td>
<td>257, 274</td>
</tr>
<tr>
<td>NLMOD procedure</td>
<td>225</td>
</tr>
<tr>
<td>additional estimates</td>
<td>235, 248</td>
</tr>
<tr>
<td>additional estimates correlation</td>
<td>249</td>
</tr>
<tr>
<td>additional estimates covariance</td>
<td>249</td>
</tr>
<tr>
<td>ANOVA</td>
<td>248</td>
</tr>
<tr>
<td>B-spline basis</td>
<td>29</td>
</tr>
<tr>
<td>Bernoulli distribution</td>
<td>237</td>
</tr>
<tr>
<td>binary distribution</td>
<td>237</td>
</tr>
<tr>
<td>binomial distribution</td>
<td>237</td>
</tr>
<tr>
<td>bounds</td>
<td>234</td>
</tr>
<tr>
<td>computational method</td>
<td>68</td>
</tr>
<tr>
<td>convergence criterion</td>
<td>38–42</td>
</tr>
<tr>
<td>convergence status</td>
<td>247</td>
</tr>
<tr>
<td>correlation</td>
<td>248</td>
</tr>
<tr>
<td>covariance</td>
<td>248</td>
</tr>
<tr>
<td>dimensions</td>
<td>247</td>
</tr>
<tr>
<td>displayed output</td>
<td>246</td>
</tr>
<tr>
<td>distribution functions</td>
<td>244</td>
</tr>
<tr>
<td>empirical Bayes estimate</td>
<td>240</td>
</tr>
<tr>
<td>fit statistics</td>
<td>248</td>
</tr>
<tr>
<td>function-based convergence criteria</td>
<td>38, 39</td>
</tr>
<tr>
<td>gamma distribution</td>
<td>237</td>
</tr>
<tr>
<td>Gaussian distribution</td>
<td>236</td>
</tr>
<tr>
<td>general distribution</td>
<td>237</td>
</tr>
<tr>
<td>gradient-based convergence criteria</td>
<td>38, 40, 41</td>
</tr>
<tr>
<td>initial values</td>
<td>237</td>
</tr>
<tr>
<td>input data tables</td>
<td>233</td>
</tr>
<tr>
<td>iteration history</td>
<td>247</td>
</tr>
<tr>
<td>lag functionality</td>
<td>242</td>
</tr>
<tr>
<td>least squares</td>
<td>243</td>
</tr>
<tr>
<td>least squares distribution</td>
<td>236</td>
</tr>
<tr>
<td>linear constraints</td>
<td>247</td>
</tr>
<tr>
<td>log-likelihood functions</td>
<td>244</td>
</tr>
<tr>
<td>multithreading</td>
<td>68, 246</td>
</tr>
<tr>
<td>natural cubic spline basis</td>
<td>31</td>
</tr>
<tr>
<td>negative binomial distribution</td>
<td>237</td>
</tr>
<tr>
<td>normal distribution</td>
<td>236</td>
</tr>
<tr>
<td>number of observations</td>
<td></td>
</tr>
<tr>
<td>GENSELECT procedure</td>
<td>123</td>
</tr>
<tr>
<td>KCLUS procedure</td>
<td>153</td>
</tr>
<tr>
<td>LOGSELECT procedure</td>
<td>206</td>
</tr>
<tr>
<td>PCA procedure</td>
<td>277</td>
</tr>
<tr>
<td>PLSMOD procedure</td>
<td>317</td>
</tr>
<tr>
<td>QTRSELECT procedure</td>
<td>360</td>
</tr>
<tr>
<td>REGSELECT procedure</td>
<td>399</td>
</tr>
<tr>
<td>VARREDUCE procedure</td>
<td>568</td>
</tr>
<tr>
<td>number of variables</td>
<td></td>
</tr>
<tr>
<td>PCA procedure</td>
<td>277</td>
</tr>
<tr>
<td>ODS graph names</td>
<td></td>
</tr>
<tr>
<td>PCA procedure</td>
<td>279</td>
</tr>
</tbody>
</table>
**Subject Index**

TREESPLIT procedure, 451

ODS table names
- GENSELECT procedure, 31
- KCLUS procedure, 31
- LOGSELECT procedure, 31
- NLMOD procedure, 31
- PCA procedure, 31, 279
- PLSMOD procedure, 319
- REGSELECT procedure, 31
- TREESPLIT procedure, 31, 450

optimization algorithms
- GENSELECT procedure, 123
- LOGSELECT procedure, 205
- NLMOD procedure, 246

optimization techniques
- GENSELECT procedure, 42, 68
- KCLUS procedure, 42, 68
- LOGSELECT procedure, 42, 68
- NLMOD procedure, 42, 68
- PCA procedure, 42, 68
- REGSELECT procedure, 42, 68
- Shared Concepts, 42
- TREESPLIT procedure, 42, 68

options summary
- CLASS statement, 11, 555
- EFFECT statement, 18, 91, 183, 305, 343, 385
- PROC BINNING statement, 488
- PROC CARDINALITY statement, 508
- PROC GENSELECT statement, 86
- PROC KCLUS statement, 141
- PROC LOGSELECT statement, 177
- PROC NLMOD statement, 232
- PROC QTRSELECT statement, 339
- PROC REGSELECT statement, 381
- PROC TREESPLIT statement, 422
- PROC VARREDUCE statement, 553

ordering
- of class levels (Shared Concepts), 44

ordinal parameterization
- Shared Concepts, 53
- ORTHEFFECT parameterization
- Shared Concepts, 54
- ORTHORDINAL parameterization
- Shared Concepts, 54
- ORTHOTHERM parameterization
- Shared Concepts, 54
- ORTHPOLY parameterization
- Shared Concepts, 54
- ORTHREF parameterization
- Shared Concepts, 54

output CAS tables
- GENSELECT procedure, 126
- KCLUS procedure, 156
- LOGSELECT procedure, 210
- PARTITION procedure, 529
- QTRSELECT procedure, 363
- REGSELECT procedure, 403

output data tables
- PCA procedure, 271
- PLSMOD procedure, 307
- OUTSTAT= data table
- PCA procedure, 275

parameter estimates
- GENSELECT procedure, 126
- LOGSELECT procedure, 209
- PLSMOD procedure, 319
- QTRSELECT procedure, 362
- REGSELECT procedure, 402

parameter tuning
- TREESPLIT procedure, 449

parameter-based convergence criteria
- GENSELECT procedure, 39, 42
- KCLUS procedure, 39, 42
- LOGSELECT procedure, 39, 42
- NLMOD procedure, 39, 42
- PCA procedure, 39, 42
- REGSELECT procedure, 39, 42
- Shared Concepts, 39, 42
- TREESPLIT procedure, 39, 42

parameterization
- dummy (Shared Concepts), 48
- dummy (VARREDUCE procedure), 563
- effect (Shared Concepts), 53
- GLM (Shared Concepts), 48, 563
- nonsingular (Shared Concepts), 52
- ordinal (Shared Concepts), 53
- ORTHEFFECT (Shared Concepts), 54
- ORTHORDINAL (Shared Concepts), 54
- ORTHOTHERM (Shared Concepts), 54
- ORTHPOLY (Shared Concepts), 54
- ORTHREF (Shared Concepts), 55
- polynomial (Shared Concepts), 54
- reference (Shared Concepts), 53
- Shared Concepts, 45
- thermometer (Shared Concepts), 53

partial correlation
- PCA procedure, 278
- principal components, 278

partial covariance
- PCA procedure, 278
- principal components, 278

partial least squares
- PLSMOD procedure, 288, 289, 311

PARTITION procedure, 521
- classification level, 528
- displayed output, 528
- frequency information table, 528
Subject Index

input data tables, 524
ODS table names, 529
output CAS tables, 529
output data tables, 527
training, validation, and test data, 528

PARTITION statement
syntax (Shared Concepts), 32

PCA procedure, 256
B-spline basis, 29
centering and scaling information, 277
computational method, 68, 265
correction for means, 266
correlation, 278
covariance, 278
crime rate data, example, 259
displayed output, 277
eigenvalue decomposition method, 257, 274
eigenvalues, 278
eigenvalues and eigenvectors, 274, 276, 278
eigenvectors, 278
examples, 280
explained variation of variables, 277
function-based convergence criteria, 38, 39
gradient-based convergence criteria, 38, 40, 41
input data tables, 265
ITERGS method, 257, 274
loadings, 274, 278
model information, 277
multithreading, 68
natural cubic spline basis, 31
NIPALS method, 257, 274, 281
number of observations, 277
number of variables, 277
ODS graph names, 279
ODS table names, 31, 279
optimization techniques, 42, 68
output data tables, 266, 271, 275, 276
OUTSTAT= data table, 275
parameter-based convergence criteria, 39, 42
partial correlation, 278
partial covariance, 278
regression coefficients, 278
regression statistics, 278
simple statistics, 277
splines and spline bases, 27
test data, 66
timing, 278
total variance, 278
truncated power function (TPF) basis, 28
validation, 66
percentage of variation accounted for by extracted factors
PLSMOD procedure, 318

PLSMOD procedure, 288
algorithms, 302
centering, 316
centering and scaling information, 318
class level, 303, 318
compared with other procedures, 288
components, 288
computational method, 302
dimensions, 317
displayed output, 317
eigenvalues and eigenvectors, 274, 276, 278
examples, 320
factors, 288
factors, selecting the number of, 295
input data tables, 301
introductory example, 292
latent variables, 288
latent vectors, 288
missing values, 317
model details, 318
model information, 317
number of observations, 317
ODS table names, 319
output data tables, 307
parameter estimates, 319
partial least squares regression, 288, 289, 311
percentage of variation accounted for by extracted factors, 318
predicting new observations, 299
principal component regression, 289, 312
reduced rank regression, 289, 312
scaling, 316
SIMPLS method, 311
test set validation, 315, 318, 320
timing, 319

Poisson distribution
NLMOD procedure, 237

polynomial effects
Shared Concepts, 49
VARREDUCE procedure, 564

polynomial parameterization
Shared Concepts, 54

prediction
NLMOD procedure, 240

principal component
regression (PLSMOD), 289, 312
principal components
interpreting eigenvalues, 262
partialing out variables, 273

PROC BINNING, 483

PROC BINNING binning methods
bucket, 484
cutpoint, 484
quantile, 484
Winsorized, 484
Subject Index

PROC BINNING features, 484
programming statements
NLINMOD procedure, 241

QTRSELECT procedure, 327, 345
candidates for addition or removal, 361
class level, 340, 360
computational method, 360
diagnostic statistics, 355
dimensions, 361
displayed output, 360
fit criteria, 352
fit statistics, 362
information criteria, 353
input data tables, 340
introductory example, 331
model options summary, 344
multithreading, 360
number of observations, 360
ODS table names, 363
output CAS tables, 363
output data tables, 346
parameter estimates, 362
quantile regression, 350
selected effects, 362
selection information, 360
selection reason, 362
selection summary, 361
significance level criteria, 354
stop reason, 361
test data, 357
timing, 363
validation, 357
quantile regression
QTRSELECT procedure, 350
quasi-complete separation
GENSELECT procedure, 116
LOGSELECT procedure, 198

reduce options
VARREDUCE procedure, 558
reduce options summary
VARREDUCE procedure, 558
reduced rank regression, 289
PLS MOD procedure, 312
reduced-error pruning
TREESPLIT procedure, 435, 445
reference parameterization
Shared Concepts, 53
regression
partial least squares (PROC PLS MOD), 289, 311
principal component (PROC PLS MOD), 289, 312
reduced rank (PROC PLS MOD), 289, 312
regression coefficients
PCA procedure, 278
regression effects
Shared Concepts, 48, 49
VARREDUCE procedure, 563, 564
regression statistics
PCA procedure, 278
regression trees, see decision trees
REGSELECT procedure, 371, 388
ANOVA table, 401
B-spline basis, 29
candidates for addition or removal, 400
class level, 382, 400
computational method, 68, 398
convergence criterion, 38–42
diagnostic statistics, 395
dimensions, 400
displayed output, 399
fit criteria, 393
fit statistics, 402
function-based convergence criteria, 38, 39
gradient-based convergence criteria, 38, 40, 41
input data tables, 382
introductory example, 375
model options summary, 387
multithreading, 68, 398
natural cubic spline basis, 31
number of observations, 399
ODS table names, 31, 403
optimization techniques, 42, 68
output CAS tables, 403
parameter estimates, 402
parameter-based convergence criteria, 39, 42
selected effects, 401
selection information, 399
selection reason, 401
selection summary, 400
splines and spline bases, 27
stop reason, 400
test data, 66, 397
timing, 403
truncated power function (TPF) basis, 28
Type III tables, 403
validation, 66, 397
residual distribution
NLINMOD procedure, 236
residuals
and partial correlation (PCA), 275
partial correlation (PCA), 273
response level ordering
GENSELECT procedure, 93
LOGSELECT procedure, 185
response profile
GENSELECT procedure, 124
LOGSELECT procedure, 206
584  •  Subject Index

response variable options
  GENSELECT procedure, 93
  LOGSELECT procedure, 185
restrictions
  NLMOD procedure, 240
reverse response level ordering
  GENSELECT procedure, 93
ROC information
  ASSESS procedure, 475, 476
selected effects
  GENSELECT procedure, 125
  LOGSELECT procedure, 208
  QTRSELECT procedure, 362
  REGSELECT procedure, 401
selected variable
  VARREDUCE procedure, 568
selection information
  GENSELECT procedure, 124
  LOGSELECT procedure, 207
  QTRSELECT procedure, 360
  REGSELECT procedure, 399
selection reason
  GENSELECT procedure, 125
  LOGSELECT procedure, 208
  QTRSELECT procedure, 362
  REGSELECT procedure, 401
SELECTION statement
  syntax (Shared Concepts), 32
selection summary
  GENSELECT procedure, 125
  LOGSELECT procedure, 208
  QTRSELECT procedure, 361
  REGSELECT procedure, 400
  VARREDUCE procedure, 568
separation
  GENSELECT procedure, 116
  LOGSELECT procedure, 198
Shared Concepts
  ABSCONV option, 38
  ABSFCNV option, 38
  ABSFTOL option, 38
  ABSSCNV option, 38
  ABSGTOL option, 38
  ABSTOL option, 38
  ABSSXCONV option, 39
  ABSTOL option, 38
  ABSXTOL option, 39
  adaptive LASSO selection, 63
  ANCOVA effects, 49
  ANOVA effects, 48
  at sign (@) operator, 47, 50, 565
  backward elimination, 58
  bar (|) operator, 46, 49, 50, 565
  CLASS statement, 10, 43
  classification variables, 43
  CODE statement, 14
  collection effect (EFFECT statement), 19
  colon (:) operator, 47
  continuous-by-class effects, 51
  continuous-nesting-class effects, 51
  convergence criterion, 38–42
  crossed effects, 50
  dash (-) operator, 47
  DISPLAY statement, 15
  DISPLAYOUT statement, 16
  double dash (--) operator, 48
  dummy parameterization, 48
  effect parameterization, 53
  EFFECT statement, 17
  FCONV option, 39
  FCONV2 option, 39
  forward selection, 56
  forward swap selection, 61
  FTOL option, 39
  FTOL2 option, 39
  function-based convergence criteria, 38, 39
  GCONV option, 40
  GCONV2 option, 41
  general effects, 52
  GLM parameterization, 48, 563
  gradient-based convergence criteria, 38, 40, 41
  group LASSO selection, 63
  GTOL option, 40
  GTOL2 option, 41
  interaction effects, 50
  intercept, 49
  LAR selection, 61
  LASSO selection, 62
  levelization, 43
  main effects, 49
  MAXFUNC= option, 41
  MAXITER= option, 41
  MAXTIME= option, 41
  MINTIME= option, 42
  ORDER= option, 44
  ordering of class levels, 44
  ordinal parameterization, 53
  ORTHEFFECT parameterization, 54
  ORTHORDINAL parameterization, 54
  ORTHOTHERM parameterization, 54
  ORTHPOLY parameterization, 54
<table>
<thead>
<tr>
<th>Subject Index</th>
<th>p. 585</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORTHREF parameterization</td>
<td>55</td>
</tr>
<tr>
<td>parameter-based convergence criteria</td>
<td>39, 42</td>
</tr>
<tr>
<td>parameterization</td>
<td>45</td>
</tr>
<tr>
<td>PARTITION statement</td>
<td>32</td>
</tr>
<tr>
<td>polynomial effect (EFFECT statement)</td>
<td>21</td>
</tr>
<tr>
<td>polynomial effects</td>
<td>49</td>
</tr>
<tr>
<td>polynomial parameterization</td>
<td>54</td>
</tr>
<tr>
<td>reference parameterization</td>
<td>53</td>
</tr>
<tr>
<td>regression effects</td>
<td>48, 49</td>
</tr>
<tr>
<td>SELECTION statement</td>
<td>32</td>
</tr>
<tr>
<td>singular parameterization</td>
<td>49</td>
</tr>
<tr>
<td>sort order of class levels</td>
<td>44</td>
</tr>
<tr>
<td>spline basis, B-spline</td>
<td>29</td>
</tr>
<tr>
<td>spline basis, natural cubic spline</td>
<td>31</td>
</tr>
<tr>
<td>spline basis, truncated power function</td>
<td>28</td>
</tr>
<tr>
<td>spline effect (EFFECT statement)</td>
<td>24</td>
</tr>
<tr>
<td>stepwise selection</td>
<td>59</td>
</tr>
<tr>
<td>TECHNIQUE= option</td>
<td>42</td>
</tr>
<tr>
<td>thermometer parameterization</td>
<td>53</td>
</tr>
<tr>
<td>XCONV option</td>
<td>42</td>
</tr>
<tr>
<td>XTOL option</td>
<td>42</td>
</tr>
<tr>
<td>significance level criteria</td>
<td></td>
</tr>
<tr>
<td>QTRSELECT procedure</td>
<td>354</td>
</tr>
<tr>
<td>simple statistics</td>
<td></td>
</tr>
<tr>
<td>PCA procedure</td>
<td>277</td>
</tr>
<tr>
<td>SIMPLS method</td>
<td></td>
</tr>
<tr>
<td>PLSMOD procedure</td>
<td>311</td>
</tr>
<tr>
<td>singular parameterization</td>
<td></td>
</tr>
<tr>
<td>Shared Concepts</td>
<td>49</td>
</tr>
<tr>
<td>VARREDUCE procedure</td>
<td>564</td>
</tr>
<tr>
<td>sort order</td>
<td></td>
</tr>
<tr>
<td>of class levels (Shared Concepts)</td>
<td>44</td>
</tr>
<tr>
<td>splines and spline bases</td>
<td></td>
</tr>
<tr>
<td>GENSELECT procedure</td>
<td>27</td>
</tr>
<tr>
<td>KCLUS procedure</td>
<td>27</td>
</tr>
<tr>
<td>LOGSELECT procedure</td>
<td>27</td>
</tr>
<tr>
<td>NLMod procedure</td>
<td>27</td>
</tr>
<tr>
<td>PCA procedure</td>
<td>27</td>
</tr>
<tr>
<td>REGSELECT procedure</td>
<td>27</td>
</tr>
<tr>
<td>TREESPLIT procedure</td>
<td>27</td>
</tr>
<tr>
<td>splitting criteria</td>
<td></td>
</tr>
<tr>
<td>TREESPLIT procedure</td>
<td>431, 439</td>
</tr>
<tr>
<td>standardization</td>
<td></td>
</tr>
<tr>
<td>KCLUS procedure</td>
<td>155</td>
</tr>
<tr>
<td>starting values</td>
<td></td>
</tr>
<tr>
<td>NLMod procedure</td>
<td>237</td>
</tr>
<tr>
<td>stepwise selection</td>
<td></td>
</tr>
<tr>
<td>Shared Concepts</td>
<td>59</td>
</tr>
<tr>
<td>stop reason</td>
<td></td>
</tr>
<tr>
<td>GENSELECT procedure</td>
<td>125</td>
</tr>
<tr>
<td>LOGSELECT procedure</td>
<td>208</td>
</tr>
<tr>
<td>QTRSELECT procedure</td>
<td>361</td>
</tr>
<tr>
<td>REGSELECT procedure</td>
<td>400</td>
</tr>
<tr>
<td>supervised variable selection</td>
<td></td>
</tr>
<tr>
<td>VARREDUCE procedure</td>
<td>560</td>
</tr>
<tr>
<td>surrogate rules</td>
<td></td>
</tr>
<tr>
<td>TREESPLIT procedure</td>
<td>425, 441</td>
</tr>
<tr>
<td>target variables</td>
<td></td>
</tr>
<tr>
<td>ASSESS procedure</td>
<td>474</td>
</tr>
<tr>
<td>test data</td>
<td></td>
</tr>
<tr>
<td>GENSELECT procedure</td>
<td>66</td>
</tr>
<tr>
<td>KCLUS procedure</td>
<td>66</td>
</tr>
<tr>
<td>LOGSELECT procedure</td>
<td>66</td>
</tr>
<tr>
<td>NLMod procedure</td>
<td>66</td>
</tr>
<tr>
<td>PCA procedure</td>
<td>66</td>
</tr>
<tr>
<td>QTRSELECT procedure</td>
<td>357</td>
</tr>
<tr>
<td>REGSELECT procedure</td>
<td>66, 397</td>
</tr>
<tr>
<td>TREESPLIT procedure</td>
<td>66</td>
</tr>
<tr>
<td>test set validation</td>
<td></td>
</tr>
<tr>
<td>PLSMOD procedure</td>
<td>315, 318</td>
</tr>
<tr>
<td>thermometer parameterization</td>
<td></td>
</tr>
<tr>
<td>Shared Concepts</td>
<td>53</td>
</tr>
<tr>
<td>timing</td>
<td></td>
</tr>
<tr>
<td>PCA procedure</td>
<td>278</td>
</tr>
<tr>
<td>PLSMOD procedure</td>
<td>319</td>
</tr>
<tr>
<td>QTRSELECT procedure</td>
<td>363</td>
</tr>
<tr>
<td>REGSELECT procedure</td>
<td>403</td>
</tr>
<tr>
<td>total variance</td>
<td></td>
</tr>
<tr>
<td>PCA procedure</td>
<td>278</td>
</tr>
<tr>
<td>TPF basis, see truncated power function (TPF) basis</td>
<td></td>
</tr>
<tr>
<td>training, validation, and test data</td>
<td></td>
</tr>
<tr>
<td>PARTITION procedure</td>
<td>528</td>
</tr>
<tr>
<td>TREESPLIT procedure</td>
<td>413</td>
</tr>
<tr>
<td>B-spline basis</td>
<td>29</td>
</tr>
<tr>
<td>C4.5 pruning</td>
<td>435, 444</td>
</tr>
<tr>
<td>computational method</td>
<td>68</td>
</tr>
<tr>
<td>convergence criterion</td>
<td>38–42</td>
</tr>
<tr>
<td>cost-complexity pruning</td>
<td>435, 443</td>
</tr>
<tr>
<td>decision trees</td>
<td>436</td>
</tr>
<tr>
<td>function-based convergence criteria</td>
<td>38, 39</td>
</tr>
<tr>
<td>gradient-based convergence criteria</td>
<td>38, 40, 41</td>
</tr>
<tr>
<td>input data tables</td>
<td>423</td>
</tr>
<tr>
<td>multithreading</td>
<td>68</td>
</tr>
<tr>
<td>natural cubic spline basis</td>
<td>31</td>
</tr>
<tr>
<td>ODS graph names</td>
<td>451</td>
</tr>
<tr>
<td>ODS table names</td>
<td>31, 450</td>
</tr>
<tr>
<td>optimization techniques</td>
<td>42, 68</td>
</tr>
<tr>
<td>output data tables</td>
<td>434</td>
</tr>
<tr>
<td>parameter tuning</td>
<td>449</td>
</tr>
<tr>
<td>parameter-based convergence criteria</td>
<td>39, 42</td>
</tr>
<tr>
<td>reduced-error pruning</td>
<td>435, 445</td>
</tr>
<tr>
<td>splines and spline bases</td>
<td>27</td>
</tr>
<tr>
<td>splitting criteria</td>
<td>431, 439</td>
</tr>
<tr>
<td>surrogate rules</td>
<td>425, 441</td>
</tr>
<tr>
<td>test data</td>
<td>66</td>
</tr>
<tr>
<td>truncated power function (TPF) basis</td>
<td>28</td>
</tr>
<tr>
<td>validation</td>
<td>66</td>
</tr>
</tbody>
</table>
trimmed statistics
  BINNING procedure, 494
truncated power function (TPF) basis
  GENSELECT procedure, 28
  KCLUS procedure, 28
  LOGSELECT procedure, 28
  NLMOD procedure, 28
  PCA procedure, 28
  REGSELECT procedure, 28
  Shared Concepts, 28
  TREESPLIT procedure, 28

Type III tables
  LOGSELECT procedure, 209
  REGSELECT procedure, 403

unsupervised variable selection
  VARREDUCE procedure, 559

validation
  GENSELECT procedure, 66
  KCLUS procedure, 66
  LOGSELECT procedure, 66
  NLMOD procedure, 66
  PCA procedure, 66
  QTRSELECT procedure, 357
  REGSELECT procedure, 66, 397
  TREESPLIT procedure, 66

variable information value (IV) table
  BINNING procedure, 494
  VARIMPUTE procedure, 535
  displayed output, 540
  imputation information table, 541
  imputation requests table, 541
  input data tables, 538
  ODS table names, 541
  output data tables, 539
  VARREDUCE procedure, 546

  ANCOVA effects, 564
  ANOVA effects, 563
  bar (|) operator, 564
  computational method, 562
  continuous-by-class effects, 566
  continuous-nesting-class effects, 566
  correlations, 554
  crossed effects, 565
  displayed output, 567
  dummy parameterization, 563
  fit criteria, 561
  general effects, 567
  input data tables, 553
  interaction effects, 565
  intercept, 564
  main effects, 564
  maximum effects, 559

maximum steps, 559
minimal explained variance increment, 559
missing values, 559
nested effects, 565
nested versus crossed effects, 565
number of observations, 568
ODS table names, 568
polynomial effects, 564
regression effects, 563, 564
selected variable, 568
selection summary, 568
singular parameterization, 564
supervised variable selection, 560
unsupervised variable selection, 559
variance explained, 559

Winsorized statistics
  BINNING procedure, 494

within-cluster statistics
  KCLUS procedure, 154
Syntax Index

ABSCONV option
  PROC GENSELECT statement, 38
  PROC KCLUS statement, 38
  PROC LOGSELECT statement, 38
  PROC NLMOD statement, 38
  PROC PCA statement, 38
  PROC REGSELECT statement, 38
  PROC TREESPLIT statement, 38

ABSFCONV option
  PROC GENSELECT statement, 38
  PROC KCLUS statement, 38
  PROC LOGSELECT statement, 38
  PROC NLMOD statement, 38
  PROC PCA statement, 38
  PROC REGSELECT statement, 38
  PROC TREESPLIT statement, 38

ABSFTOL option
  PROC GENSELECT statement, 38
  PROC KCLUS statement, 38
  PROC LOGSELECT statement, 38
  PROC NLMOD statement, 38
  PROC PCA statement, 38
  PROC REGSELECT statement, 38
  PROC TREESPLIT statement, 38

ABSGCONV option
  PROC GENSELECT statement, 38
  PROC KCLUS statement, 38
  PROC LOGSELECT statement, 38
  PROC NLMOD statement, 38
  PROC PCA statement, 38
  PROC REGSELECT statement, 38
  PROC TREESPLIT statement, 38

ABSGTOL option
  PROC GENSELECT statement, 38
  PROC KCLUS statement, 38
  PROC LOGSELECT statement, 38
  PROC NLMOD statement, 38
  PROC PCA statement, 38
  PROC REGSELECT statement, 38
  PROC TREESPLIT statement, 38

ABSTOL option
  PROC GENSELECT statement, 38
  PROC KCLUS statement, 38
  PROC LOGSELECT statement, 38
  PROC NLMOD statement, 38
  PROC PCA statement, 38
  PROC REGSELECT statement, 38
  PROC TREESPLIT statement, 38

ABXCONV option
  PROC GENSELECT statement, 39
  PROC KCLUS statement, 39
  PROC LOGSELECT statement, 39
  PROC NLMOD statement, 39
  PROC PCA statement, 39
  PROC REGSELECT statement, 39
  PROC TREESPLIT statement, 39

ABSXTOL option
  PROC GENSELECT statement, 39
  PROC KCLUS statement, 39
  PROC LOGSELECT statement, 39
  PROC NLMOD statement, 39
  PROC PCA statement, 39
  PROC REGSELECT statement, 39
  PROC TREESPLIT statement, 39

ADAPTIVE option
  SELECTION statement (GENSELECT), 34
  SELECTION statement (KCLUS), 34
  SELECTION statement (LOGSELECT), 34
  SELECTION statement (NLMOD), 34
  SELECTION statement (PCA), 34
  SELECTION statement (REGSELECT), 34
  SELECTION statement (TREESPLIT), 34

AIC option
  REDUCE statement, 558

AICC option
  REDUCE statement, 558

ALGORITHM= option
  PROC PLSMOD statement, METHOD=PLS option, 302

ALL option
  OUTPUT statement (GENSELECT), 99
  OUTPUT statement (LOGSELECT), 189

ALPHA= option
  ESTIMATE statement (NLMOD), 236
  GROW statement (TREESPLIT), 432
  OUTPUT statement (GENSELECT), 99
  OUTPUT statement (LOGSELECT), 189
  PREDICT statement (NLMOD), 240
  PROC GENSELECT statement, 87
  PROC LOGSELECT statement, 178
  PROC NLMOD statement, 233
  PROC QTRSELECT statement, 340
  PROC REGSELECT statement, 382
  PRUNE statement (TREESPLIT), 435

ASSESS procedure, BY statement, 472
ASSESS procedure, FITSTAT statement, 472
DELMITE= option, 473
DLM= option, 473
PEVENT= option, 473
PVAR= option, 473
ASSESS procedure, INPUT statement, 473
ASSESS procedure, PROC ASSESS statement, 471
   DATA= option, 471
   FITSTATOUT= option, 471
   LIFTOUT= option, 471
   MAXITER= option, 472
   NBINS= option, 472
   NCUTS= option, 472
   NTHREADS= option, 472
   ROCOUT= option, 472
ASSESS procedure, syntax, 471
ASSESS procedure, TARGET statement, 474
   EVENT= option, 474
   LEVEL= option, 474
ASSESS procedure, VAR statement, 473
ASSESS procedure, syntax, 471
ASSIGNMISSING= option
PROC TREESPLIT statement, 422
AUTOTUNE statement
   TREESPLIT procedure, 427
   BASIS option
      EFFECT statement, spline (GENSELECT), 24
      EFFECT statement, spline (KCLUS), 24
      EFFECT statement, spline (LOGSELECT), 24
      EFFECT statement, spline (NLMOD), 24
      EFFECT statement, spline (PCA), 24
      EFFECT statement, spline (REGSELECT), 24
      EFFECT statement, spline (TREESPLIT), 24
   BIC option
      REDUCE statement, 558
   BINNING procedure, CODE statement, 490
      FILE= option, 490
   BINNING procedure, INPUT statement, 490
      NUMBIN= option, 490
   BINNING procedure, OUTPUT statement, 490
      COPYVARS= option, 491
      OUT= option, 491
   BINNING procedure, PROC BINNING statement, 488
      DATA= option, 489
      METHOD= option, 489
      NUMBIN= option, 489
      WOE(WOEAJUST)= option, 489
   BINNING procedure, syntax, 488
   BINNING procedure, TARGET statement, 491
      EVENT= option, 491
   BONFERRONI option
      GROW statement (TREESPLIT), 432
   BOUNDS statement
      NLMOD procedure, 234
   BY statement
   ASSESS procedure, 472
   GENSELECT procedure, 88
   LOGSELECT procedure, 180
   NLMOD procedure, 234
   PARTITION procedure, 525
   PCA procedure, 269
   QTRSELECT procedure, 340
   REGSELECT procedure, 382
   C45 option
      PRUNE statement (TREESPLIT), 435
   CARDINALITY procedure
      VAR statement, 509
   CARDINALITY procedure, PROC CARDINALITY
      statement, 508
      DATA= option, 508
      MAXLEVELS= option, 509
      ORDER= option, 509
      OUTCARD= option, 508
      OUTDETAILS= option, 509
   CARDINALITY procedure, syntax, 508
   CARDINALITY procedure, VAR statement
      ORDER= option, 510
   CASESENSITIVE option
      DISPLAY statement (GENSELECT), 16, 89
      DISPLAY statement (KCLUS), 16, 146
      DISPLAY statement (LOGSELECT), 16, 182
      DISPLAY statement (NLMOD), 16, 234
      DISPLAY statement (PARTITION), 526
      DISPLAY statement (PCA), 16, 270
      DISPLAY statement (PLSMOD), 304
      DISPLAY statement (QTRSELECT), 341
      DISPLAY statement (REGSELECT), 16, 384
      DISPLAY statement (TREESPLIT), 16
      DISPLAY statement (VARREDUCE), 556
   CBAR= option
      OUTPUT statement (GENSELECT), 100
      OUTPUT statement (LOGSELECT), 190
   CENSCALE option
      PROC PLSMOD statement, 301
   CHAID option
      GROW statement (TREESPLIT), 432
   CHISQUARE option
      GROW statement (TREESPLIT), 432
   CHOOSE= option
      SELECTION statement (GENSELECT), 34
      SELECTION statement (KCLUS), 34
      SELECTION statement (LOGSELECT), 34
      SELECTION statement (NLMOD), 34
      SELECTION statement (PCA), 34
      SELECTION statement (REGSELECT), 34
      SELECTION statement (TREESPLIT), 34
   CLASS statement
      GENSELECT procedure, 10, 43, 88
KCLUS procedure, 10, 43
LOGSELECT procedure, 10, 43, 180
NLMOD procedure, 10, 43
PCA procedure, 10, 43
PLSMOD procedure, 303
QTRSELECT procedure, 340
REGSELECT procedure, 10, 43, 382
TREESPLIT procedure, 10, 43, 431
VARREDUCE procedure, 555

CLB option
MODEL statement (GENSELECT), 95
MODEL statement (LOGSELECT), 187
MODEL statement (QTRSELECT), 345
MODEL statement (REGSELECT), 387

CODE statement
BINNING procedure, 490
GENSELECT procedure, 14, 89
KCLUS procedure, 14, 146
LOGSELECT procedure, 14, 181
NLMOD procedure, 14
PCA procedure, 14, 269
REGSELECT procedure, 14, 383
TREESPLIT procedure, 14, 431
VARIMPUTE procedure, 538

COMMENT option
CODE statement (GENSELECT), 14
CODE statement (KCLUS), 14
CODE statement (LOGSELECT), 14
CODE statement (NLMOD), 14
CODE statement (PCA), 14
CODE statement (REGSELECT), 14
CODE statement (TREESPLIT), 14

COMPETITIVE option
SELECTION statement (GENSELECT), 34
SELECTION statement (KCLUS), 34
SELECTION statement (LOGSELECT), 34
SELECTION statement (NLMOD), 34
SELECTION statement (PCA), 34
SELECTION statement (REGSELECT), 34
SELECTION statement (TREESPLIT), 34

CONFIDENCE= option
PRUNE statement (TREESPLIT), 435

COPYVARS= option
OUTPUT statement (GENSELECT), 99
OUTPUT statement (LOGSELECT), 189
OUTPUT statement (QTRSELECT), 346
OUTPUT statement (REGSELECT), 389

COPYVARS= option
OUTPUT statement, 149, 434, 491, 528, 540
OUTPUT statement (PCA), 272
OUTPUT statement (PLSMOD), 308

CORR option
PROC NLMOD statement, 233

CORRB option
PROC GENSELECT statement, 87
PROC LOGSELECT statement, 179

COSTCOMPLEXITY option
PRUNE statement (TREESPLIT), 435

COV option
PROC NLMOD statement, 233
PROC PCA statement, 265

COV=SPARSITY option
PROC QTRSELECT statement, 340

COVARIANCE option
PROC PCA statement, 265

DATA= option
PROC ASSESS statement, 471
PROC BINNING statement, 489
PROC CARDINALITY statement, 508
PROC GENSELECT statement, 87
PROC KCLUS statement, 142
PROC LOGSELECT statement, 179
PROC NLMOD statement, 233
PROC PARTITION statement, 524
PROC PCA statement, 265
PROC PLSMOD statement, 301
PROC QTRSELECT statement, 340
PROC REGSELECT statement, 382
PROC TREESPLIT statement, 423
PROC VARIMPUTE statement, 538
PROC VARREDUCE statement, 553

DATABOUNDARY option
EFFECT statement, spline (GENSELECT), 24
EFFECT statement, spline (KCLUS), 24
EFFECT statement, spline (LOGSELECT), 24
EFFECT statement, spline (NLMOD), 24
EFFECT statement, spline (PCA), 24
EFFECT statement, spline (REGSELECT), 24
EFFECT statement, spline (TREESPLIT), 24

DEGREE option
EFFECT statement, polynomial (GENSELECT), 22
EFFECT statement, polynomial (KCLUS), 22
EFFECT statement, polynomial (LOGSELECT), 22
EFFECT statement, polynomial (NLMOD, 22
EFFECT statement, polynomial (PCA), 22
EFFECT statement, polynomial (REGSELECT), 22
EFFECT statement, polynomial (TREESPLIT), 22
EFFECT statement, spline (GENSELECT), 25
EFFECT statement, spline (KCLUS), 25
EFFECT statement, spline (LOGSELECT), 25
EFFECT statement, spline (NLMOD), 25
EFFECT statement, spline (PCA), 25
EFFECT statement, spline (REGSELECT), 25
EFFECT statement, spline (TREESPLIT), 25
DELIMITER= option
FITSTAT statement, 473
DESCENDING option
CLASS statement (GENSELECT), 11, 555
CLASS statement (KCLUS), 11, 555
CLASS statement (LOGSELECT), 11, 555
CLASS statement (NLMOD), 11, 555
CLASS statement (PCA), 11, 555
CLASS statement (REGSELECT), 11, 555
CLASS statement (TREESPLIT), 11, 555
MODEL statement (GENSELECT), 93
MODEL statement (LOGSELECT), 93
DETAILS option
EFFECT statement, multimember
(GENSELECT), 20
EFFECT statement, multimember (KCLUS), 20
EFFECT statement, multimember (LOGSELECT), 20
EFFECT statement, multimember (NLMOD), 20
EFFECT statement, multimember (PCA), 20
EFFECT statement, multimember (REGSELECT), 20
EFFECT statement, multimember (TREESPLIT), 20
EFFECT statement, polynomial (GENSELECT), 22
EFFECT statement, polynomial (KCLUS), 22
EFFECT statement, polynomial (LOGSELECT), 22
EFFECT statement, polynomial (NLMOD), 22
EFFECT statement, polynomial (PCA), 22
EFFECT statement, polynomial (REGSELECT), 22
EFFECT statement, polynomial (TREESPLIT), 22
EFFECT statement, spline (GENSELECT), 25
EFFECT statement, spline (KCLUS), 25
EFFECT statement, spline (LOGSELECT), 25
EFFECT statement, spline (NLMOD), 25
EFFECT statement, spline (PCA), 25
EFFECT statement, spline (REGSELECT), 25
EFFECT statement, spline (TREESPLIT), 25
PROC PLSMOD statement, 302
DETAILS= option
SELECTION statement (GENSELECT), 36
SELECTION statement (KCLUS), 36
SELECTION statement (LOGSELECT), 36
SELECTION statement (NLMOD), 36
SELECTION statement (PCA), 36
SELECTION statement (REGSELECT), 36
SELECTION statement (TREESPLIT), 36
DF= option
ESTIMATE statement (NLMOD), 236
PREDICT statement (NLMOD), 240
PROC NLNMOD statement, 233
DIFCHISQ= option
OUTPUT statement (GENSELECT), 100
OUTPUT statement (LOGSELECT), 190
DIFDEV= option
OUTPUT statement (GENSELECT), 100
OUTPUT statement (LOGSELECT), 190
DISPLAY statement
GENSELECT procedure, 15, 89
KCLUS procedure, 15, 146
LOGSELECT procedure, 15, 181
NLMOD procedure, 15, 234
PARTITION procedure, 526
PCA procedure, 15, 269
PLSMOD procedure, 304
QTRSELECT procedure, 341
REGSELECT procedure, 15, 383
TREESPLIT procedure, 15
VARREDUCE procedure, 556
DISPLAYOUT statement
GENSELECT procedure, 16
KCLUS procedure, 16
LOGSELECT procedure, 16
NLMOD procedure, 16
PCA procedure, 16
REGSELECT procedure, 16
TREESPLIT procedure, 16
DISTANCE= option
PROC KCLUS statement, 142
DISTANCENOM= option
PROC KCLUS statement, 142
DISTRIBUTION= option
MODEL statement (GENSELECT), 95
DLM= option
FITSTAT statement, 473
ECORR option
PROC NLNMOD statement, 233
ECOV option
### Syntax Index

**PROC NLMOD statement**, 233

**EFFECT statement**
- collection effect, 19
- GENSELECT procedure, 17, 91
- KCLUS procedure, 17
- LOGSELECT procedure, 17, 183
- multimember effect, 19
- NLMOD procedure, 17
- PCA procedure, 17
- PLSMOD procedure, 305
- polynomial effect, 21
- QTRSELECT procedure, 342
- REGSELECT procedure, 17, 385
- spline effect, 24
- TREESPLIT procedure, 17

**ENTROPY option**
- GROW statement (TREESPLIT), 432

**EPSILON= option**
- PROC PCA statement, METHOD=ITERGS option, 265
- PROC PCA statement, METHOD=NIPALS option, 265
- PROC PLSMOD statement, METHOD=PLS option, 302

**ESTIMATE statement**
- NLMOD procedure, 236

**EVENT= option**
- PROC PARTITION statement, 525
- TARGET statement, 474, 491

**EVENTPROP= option**
- PROC PARTITION statement, 525

**EXCLUDE option**
- DISPLAY statement (GENSELECT), 16, 89
- DISPLAY statement (KCLUS), 16, 147
- DISPLAY statement (LOGSELECT), 16, 182
- DISPLAY statement (NLMOD), 16, 235
- DISPLAY statement (PARTITION), 526
- DISPLAY statement (PCA), 16, 270
- DISPLAY statement (PLSMOD), 304
- DISPLAY statement (QTRSELECT), 341
- DISPLAY statement (REGSELECT), 16, 384
- DISPLAY statement (TREESPLIT), 16
- DISPLAY statement (VARREDUCE), 556

**EXCLUDEALL option**
- DISPLAY statement (GENSELECT), 16, 90
- DISPLAY statement (KCLUS), 16, 147
- DISPLAY statement (LOGSELECT), 16, 182
- DISPLAY statement (NLMOD), 16, 235
- DISPLAY statement (PARTITION), 526
- DISPLAY statement (PCA), 16, 270
- DISPLAY statement (PLSMOD), 304
- DISPLAY statement (QTRSELECT), 341
- DISPLAY statement (REGSELECT), 16, 384
- DISPLAY statement (TREESPLIT), 16

**DISPLAY statement (VARREDUCE), 556**

**FAST option**
- SELECTION statement (GENSELECT), 35
- SELECTION statement (KCLUS), 35
- SELECTION statement (LOGSELECT), 35
- SELECTION statement (NLMOD), 35
- SELECTION statement (PCA), 35
- SELECTION statement (REGSELECT), 35
- SELECTION statement (TREESPLIT), 35

**FCONV option**
- PROC GENSELECT statement, 39
- PROC KCLUS statement, 39
- PROC LOGSELECT statement, 39
- PROC NLMOD statement, 39
- PROC PCA statement, 39
- PROC REGSELECT statement, 39
- PROC TREESPLIT statement, 39

**FCONV2 option**
- PROC GENSELECT statement, 39
- PROC KCLUS statement, 39
- PROC LOGSELECT statement, 39
- PROC NLMOD statement, 39
- PROC PCA statement, 39
- PROC REGSELECT statement, 39
- PROC TREESPLIT statement, 39

**FILE= option**
- CODE statement, 146, 490, 538
- CODE statement (GENSELECT), 15
- CODE statement (KCLUS), 15
- CODE statement (LOGSELECT), 15
- CODE statement (NLMOD), 15
- CODE statement (PCA), 15
- CODE statement (REGSELECT), 15
- CODE statement (TREESPLIT), 15

**FITSTAT statement**
- ASSESS procedure, 472
- FITSTATOUT= option
- PROC ASSESS statement, 471

**FOREST procedure, AUTOTUNE statement**
- FRACTION= option, 427
- KFOLD= option, 428

**FORMATWIDTH= option**
- CODE statement (GENSELECT), 15
- CODE statement (KCLUS), 15
- CODE statement (LOGSELECT), 15
- CODE statement (NLMOD), 15
- CODE statement (PCA), 15
- CODE statement (REGSELECT), 15
- CODE statement (TREESPLIT), 15

**FRACTION option**
- PARTITION statement (GENSELECT), 32, 103
- PARTITION statement (KCLUS), 32
- PARTITION statement (LOGSELECT), 32, 193
<table>
<thead>
<tr>
<th>Syntax Index</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PARTITION statement (NLMOD), 32</strong></td>
<td><strong>PROC PCA statement, 41</strong></td>
</tr>
<tr>
<td><strong>PARTITION statement (PCA), 32</strong></td>
<td><strong>PROC REGSELECT statement, 41</strong></td>
</tr>
<tr>
<td><strong>PARTITION statement (QTRSELECT), 348</strong></td>
<td><strong>PROC TREESPLIT statement, 41</strong></td>
</tr>
<tr>
<td><strong>PARTITION statement (REGSELECT), 32, 391</strong></td>
<td><strong>GENSELECT procedure, 85</strong></td>
</tr>
<tr>
<td><strong>PARTITION statement (TREESPLIT), 32, 434</strong></td>
<td><strong>MODEL statement, 92</strong></td>
</tr>
<tr>
<td><strong>PLS MOD procedure, PARTITION statement, 310</strong></td>
<td><strong>PROC GENSELECT statement, 86</strong></td>
</tr>
<tr>
<td><strong>FRACTION= option</strong></td>
<td><strong>SELECTION statement, 103</strong></td>
</tr>
<tr>
<td><strong>AUTOTUNE statement, 427</strong></td>
<td><strong>syntax, 85</strong></td>
</tr>
<tr>
<td><strong>FREQ statement</strong></td>
<td><strong>GENSELECT procedure, CLASS statement, 88</strong></td>
</tr>
<tr>
<td><strong>ASSESS procedure, 473</strong></td>
<td><strong>DESCENDING option, 11, 555</strong></td>
</tr>
<tr>
<td><strong>BINNING procedure, 490</strong></td>
<td><strong>MISSING option, 11, 555</strong></td>
</tr>
<tr>
<td><strong>CARDINALITY procedure, 509</strong></td>
<td><strong>ORDER= option, 11</strong></td>
</tr>
<tr>
<td><strong>GENSELECT procedure, 92</strong></td>
<td><strong>PARAM= option, 12</strong></td>
</tr>
<tr>
<td><strong>KCLUS procedure, 148</strong></td>
<td><strong>REF= option, 13</strong></td>
</tr>
<tr>
<td><strong>LOOSE SELECT procedure, 184</strong></td>
<td><strong>SPLIT option, 13</strong></td>
</tr>
<tr>
<td><strong>PCA procedure, 271</strong></td>
<td><strong>GENSELECT procedure, CODE statement, 89</strong></td>
</tr>
<tr>
<td><strong>REGSELECT procedure, 386</strong></td>
<td><strong>COMMENT option, 14</strong></td>
</tr>
<tr>
<td><strong>TREESPLIT procedure, 431</strong></td>
<td><strong>FILE= option, 15</strong></td>
</tr>
<tr>
<td><strong>VAR IMPUTE procedure, 538</strong></td>
<td><strong>FORMATWIDTH= option, 15</strong></td>
</tr>
<tr>
<td><strong>VAR REDUCE procedure, 557</strong></td>
<td><strong>INDENTSIZE= option, 15</strong></td>
</tr>
<tr>
<td><strong>FREQNAME= option</strong></td>
<td><strong>LABELID= option, 15</strong></td>
</tr>
<tr>
<td><strong>OUTPUT statement, 528</strong></td>
<td><strong>LINESIZE= option, 15</strong></td>
</tr>
<tr>
<td><strong>FTEST option</strong></td>
<td><strong>NOTRIM option, 15</strong></td>
</tr>
<tr>
<td><strong>GROW statement (TREESPLIT), 433</strong></td>
<td><strong>PCATALL option, 15</strong></td>
</tr>
<tr>
<td><strong>FTOL option</strong></td>
<td><strong>GENSELECT procedure, DISPLAY statement</strong></td>
</tr>
<tr>
<td><strong>PROC GENSELECT statement, 39</strong></td>
<td><strong>CASESENSITIVE option, 16, 89</strong></td>
</tr>
<tr>
<td><strong>PROC KCLUS statement, 39</strong></td>
<td><strong>EXCLUDE option, 16, 89</strong></td>
</tr>
<tr>
<td><strong>PROC LOGSELECT statement, 39</strong></td>
<td><strong>EXCLUDEALL option, 16, 90</strong></td>
</tr>
<tr>
<td><strong>PROC NLMOD statement, 39</strong></td>
<td><strong>TRACE option, 16, 90</strong></td>
</tr>
<tr>
<td><strong>PROC PCA statement, 39</strong></td>
<td><strong>GENSELECT procedure, DISPLAYOUT statement</strong></td>
</tr>
<tr>
<td><strong>PROC REGSELECT statement, 39</strong></td>
<td><strong>NOREPLACE option, 17, 90</strong></td>
</tr>
<tr>
<td><strong>PROC TREESPLIT statement, 39</strong></td>
<td><strong>REPEATED option, 17, 90</strong></td>
</tr>
<tr>
<td><strong>FTOL2 option</strong></td>
<td><strong>GENSELECT procedure, EFFECT statement</strong></td>
</tr>
<tr>
<td><strong>PROC GENSELECT statement, 39</strong></td>
<td><strong>BASIS option (spline), 24</strong></td>
</tr>
<tr>
<td><strong>PROC KCLUS statement, 39</strong></td>
<td><strong>DATABOUNDARY option (spline), 24</strong></td>
</tr>
<tr>
<td><strong>PROC LOGSELECT statement, 39</strong></td>
<td><strong>DEGREE option (polynomial), 22</strong></td>
</tr>
<tr>
<td><strong>PROC NLMOD statement, 39</strong></td>
<td><strong>DEGREE option (spline), 25</strong></td>
</tr>
<tr>
<td><strong>PROC PCA statement, 39</strong></td>
<td><strong>DETAILS option (multimember), 20</strong></td>
</tr>
<tr>
<td><strong>PROC REGSELECT statement, 39</strong></td>
<td><strong>DETAILS option (polynomial), 22</strong></td>
</tr>
<tr>
<td><strong>PROC TREESPLIT statement, 39</strong></td>
<td><strong>DETAILS option (spline), 25</strong></td>
</tr>
<tr>
<td><strong>GCONV option</strong></td>
<td><strong>KNOTMAX= option (spline), 25</strong></td>
</tr>
<tr>
<td><strong>PROC GENSELECT statement, 40</strong></td>
<td><strong>KNOTMETHOD option (spline), 25</strong></td>
</tr>
<tr>
<td><strong>PROC KCLUS statement, 40</strong></td>
<td><strong>KNOTMIN= option (spline), 26</strong></td>
</tr>
<tr>
<td><strong>PROC LOGSELECT statement, 40</strong></td>
<td><strong>MDEGREE option (polynomial), 22</strong></td>
</tr>
<tr>
<td><strong>PROC NLMOD statement, 40</strong></td>
<td><strong>NATURALCUBIC option (spline), 26</strong></td>
</tr>
<tr>
<td><strong>PROC PCA statement, 40</strong></td>
<td><strong>NO EFFECT option (multimember), 20</strong></td>
</tr>
<tr>
<td><strong>PROC REGSELECT statement, 40</strong></td>
<td><strong>NOSEPARATE option (polynomial), 22</strong></td>
</tr>
<tr>
<td><strong>PROC TREESPLIT statement, 40</strong></td>
<td><strong>SEPARATE option (spline), 26</strong></td>
</tr>
<tr>
<td><strong>GCONV2 option</strong></td>
<td><strong>SPLIT option (spline), 27</strong></td>
</tr>
<tr>
<td><strong>PROC GENSELECT statement, 41</strong></td>
<td><strong>STANDARDIZE option (polynomial), 22</strong></td>
</tr>
<tr>
<td><strong>PROC KCLUS statement, 41</strong></td>
<td><strong>GENSELECT procedure, MODEL statement, 92</strong></td>
</tr>
<tr>
<td><strong>PROC LOGSELECT statement, 41</strong></td>
<td><strong>CLB option, 95</strong></td>
</tr>
<tr>
<td><strong>PROC NLMOD statement, 41</strong></td>
<td><strong>DESCENDING option, 93</strong></td>
</tr>
<tr>
<td></td>
<td><strong>DISTRIBUTION= option, 95</strong></td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>INCLUDE option</td>
<td></td>
</tr>
<tr>
<td>INFORMATIVE option</td>
<td></td>
</tr>
<tr>
<td>LINK= option</td>
<td></td>
</tr>
<tr>
<td>NOCHECK option</td>
<td></td>
</tr>
<tr>
<td>NOINT option</td>
<td></td>
</tr>
<tr>
<td>OFFSET= option</td>
<td></td>
</tr>
<tr>
<td>ORDER= option</td>
<td></td>
</tr>
<tr>
<td>PHI= option</td>
<td></td>
</tr>
<tr>
<td>START option</td>
<td></td>
</tr>
<tr>
<td>TYPE3 option</td>
<td></td>
</tr>
<tr>
<td>GENSELECT procedure, OUTPUT statement</td>
<td></td>
</tr>
<tr>
<td>ALL option</td>
<td></td>
</tr>
<tr>
<td>ALPHA= option</td>
<td></td>
</tr>
<tr>
<td>CBAR= option</td>
<td></td>
</tr>
<tr>
<td>COPYV AR= option</td>
<td></td>
</tr>
<tr>
<td>DIFCHISQ= option</td>
<td></td>
</tr>
<tr>
<td>DIFDEV= option</td>
<td></td>
</tr>
<tr>
<td>H= option</td>
<td></td>
</tr>
<tr>
<td>INDIVIDUAL= option</td>
<td></td>
</tr>
<tr>
<td>IPRED= option</td>
<td></td>
</tr>
<tr>
<td>keyword= option</td>
<td></td>
</tr>
<tr>
<td>LCL= option</td>
<td></td>
</tr>
<tr>
<td>LCLM= option</td>
<td></td>
</tr>
<tr>
<td>LINP= option</td>
<td></td>
</tr>
<tr>
<td>LOWER= option</td>
<td></td>
</tr>
<tr>
<td>LOWERXBETA= option</td>
<td></td>
</tr>
<tr>
<td>OBSCAT option</td>
<td></td>
</tr>
<tr>
<td>OUT= option</td>
<td></td>
</tr>
<tr>
<td>PREDICTED= option</td>
<td></td>
</tr>
<tr>
<td>RESCHI= option</td>
<td></td>
</tr>
<tr>
<td>RESDEV= option</td>
<td></td>
</tr>
<tr>
<td>RESIDUAL= option</td>
<td></td>
</tr>
<tr>
<td>RESLIK= option</td>
<td></td>
</tr>
<tr>
<td>RESRAW= option</td>
<td></td>
</tr>
<tr>
<td>RESWORK= option</td>
<td></td>
</tr>
<tr>
<td>ROLE= option</td>
<td></td>
</tr>
<tr>
<td>STDRESCHI= option</td>
<td></td>
</tr>
<tr>
<td>STDRESDEV= option</td>
<td></td>
</tr>
<tr>
<td>STDXBETA = option</td>
<td></td>
</tr>
<tr>
<td>UCL.= option</td>
<td></td>
</tr>
<tr>
<td>UCLM= option</td>
<td></td>
</tr>
<tr>
<td>UPPER= option</td>
<td></td>
</tr>
<tr>
<td>UPPERXBETA= option</td>
<td></td>
</tr>
<tr>
<td>XBETA= option</td>
<td></td>
</tr>
<tr>
<td>GENSELECT procedure, PARTITION statement</td>
<td></td>
</tr>
<tr>
<td>FRACTION option</td>
<td></td>
</tr>
<tr>
<td>ROLEV AR= option</td>
<td></td>
</tr>
<tr>
<td>GENSELECT procedure, PROC GENSELECT statement</td>
<td></td>
</tr>
<tr>
<td>ABSCONV option</td>
<td></td>
</tr>
<tr>
<td>ABSFCONV option</td>
<td></td>
</tr>
<tr>
<td>ABSFTOL option</td>
<td></td>
</tr>
<tr>
<td>ABSGCONV option</td>
<td></td>
</tr>
<tr>
<td>ABSGTOL option</td>
<td></td>
</tr>
<tr>
<td>ABSTOL option</td>
<td></td>
</tr>
<tr>
<td>ABSXCONV option</td>
<td></td>
</tr>
<tr>
<td>ABSXTOL option</td>
<td></td>
</tr>
<tr>
<td>ALPHA= option</td>
<td></td>
</tr>
<tr>
<td>CORRB option</td>
<td></td>
</tr>
<tr>
<td>COVB option</td>
<td></td>
</tr>
<tr>
<td>DATA= option</td>
<td></td>
</tr>
<tr>
<td>FCONV option</td>
<td></td>
</tr>
<tr>
<td>FCONV2 option</td>
<td></td>
</tr>
<tr>
<td>FTOL option</td>
<td></td>
</tr>
<tr>
<td>FTOL2 option</td>
<td></td>
</tr>
<tr>
<td>GCONV option</td>
<td></td>
</tr>
<tr>
<td>GCONV2 option</td>
<td></td>
</tr>
<tr>
<td>GTOL option</td>
<td></td>
</tr>
<tr>
<td>GTOL2 option</td>
<td></td>
</tr>
<tr>
<td>ITHIST option</td>
<td></td>
</tr>
<tr>
<td>LASSORHO= option</td>
<td></td>
</tr>
<tr>
<td>LASSOSTEPS= option</td>
<td></td>
</tr>
<tr>
<td>LASSOTOL= option</td>
<td></td>
</tr>
<tr>
<td>MAXFUNC= option</td>
<td></td>
</tr>
<tr>
<td>MAXITER= option</td>
<td></td>
</tr>
<tr>
<td>MAXOPTBATCH= option</td>
<td></td>
</tr>
<tr>
<td>MAXTIME= option</td>
<td></td>
</tr>
<tr>
<td>MINITER= option</td>
<td></td>
</tr>
<tr>
<td>NOCLPRINT option</td>
<td></td>
</tr>
<tr>
<td>NORMALIZE= option</td>
<td></td>
</tr>
<tr>
<td>NOSTDERR option</td>
<td></td>
</tr>
<tr>
<td>PARTFIT option</td>
<td></td>
</tr>
<tr>
<td>TECHNIQUE= option</td>
<td></td>
</tr>
<tr>
<td>XCONV option</td>
<td></td>
</tr>
<tr>
<td>XTOL option</td>
<td></td>
</tr>
<tr>
<td>ADAPTIVE option</td>
<td></td>
</tr>
<tr>
<td>CHOOSE= option</td>
<td></td>
</tr>
<tr>
<td>COMPETITIVE option</td>
<td></td>
</tr>
<tr>
<td>CRITERION= option</td>
<td></td>
</tr>
<tr>
<td>DETAILS= option</td>
<td></td>
</tr>
<tr>
<td>FAST option</td>
<td></td>
</tr>
<tr>
<td>HIERARCHY= option</td>
<td></td>
</tr>
<tr>
<td>LSCOEFFS option</td>
<td></td>
</tr>
<tr>
<td>MAXEFFECTS= option</td>
<td></td>
</tr>
<tr>
<td>MAXSTEPS= option</td>
<td></td>
</tr>
<tr>
<td>METHOD= option</td>
<td></td>
</tr>
<tr>
<td>MINEFFECTS= option</td>
<td></td>
</tr>
<tr>
<td>ORDERSELECT option</td>
<td></td>
</tr>
<tr>
<td>SELECT= option</td>
<td></td>
</tr>
<tr>
<td>SELECTION= option</td>
<td></td>
</tr>
<tr>
<td>SLE= option</td>
<td></td>
</tr>
<tr>
<td>SENTRY= option</td>
<td></td>
</tr>
<tr>
<td>SLS= option</td>
<td></td>
</tr>
<tr>
<td>SLSTAY= option</td>
<td></td>
</tr>
<tr>
<td>STOP= option</td>
<td></td>
</tr>
<tr>
<td>STOPHORIZON= option</td>
<td></td>
</tr>
<tr>
<td>GINI option</td>
<td></td>
</tr>
</tbody>
</table>
594 F Syntax Index

GROW statement (TREESPLIT), 432
GROW statement
TREESPLIT procedure, 431
GTOL option
PROC GENSELECT statement, 40
PROC KCLUS statement, 40
PROC LOGSELECT statement, 40
PROC NLMOD statement, 40
PROC PCA statement, 40
PROC REGSELECT statement, 40
PROC TREESPLIT statement, 40
GTOL2 option
PROC GENSELECT statement, 41
PROC KCLUS statement, 41
PROC LOGSELECT statement, 41
PROC NLMOD statement, 41
PROC PCA statement, 41
PROC REGSELECT statement, 41
PROC TREESPLIT statement, 41
H= option
OUTPUT statement (GENSELECT), 100
OUTPUT statement (LOGSELECT), 190
HIERARCHY= option
SELECTION statement (GENSELECT), 37
SELECTION statement (KCLUS), 37
SELECTION statement (LOGSELECT), 37
SELECTION statement (NLMOD), 37
SELECTION statement (PCA), 37
SELECTION statement (REGSELECT), 37
SELECTION statement (TREESPLIT), 37
ID statement
NLMOD procedure, 236
IGR option
GROW statement (TREESPLIT), 433
IMPUTE= option
PROC KCLUS statement, 143
IMPUTENOM= option
PROC KCLUS statement, 143
INCLUDE option
MODEL statement (GENSELECT), 96
MODEL statement (LOGSELECT), 187
MODEL statement (QTRSELECT), 345
MODEL statement (REGSELECT), 387
INDENTSIZE= option
CODE statement (GENSELECT), 15
CODE statement (KCLUS), 15
CODE statement (LOGSELECT), 15
CODE statement (NLMOD), 15
CODE statement (PCA), 15
CODE statement (REGSELECT), 15
CODE statement (TREESPLIT), 15
INDIVIDUAL= option

OUTPUT statement (GENSELECT), 100
OUTPUT statement (LOGSELECT), 191
INFORMATIVE option
MODEL statement (GENSELECT), 96
MODEL statement (LOGSELECT), 187
MODEL statement (QTRSELECT), 345
MODEL statement (REGSELECT), 388
INIT= option
PROC KCLUS statement, 143
INPUT statement
ASSESS procedure, 473
BINNING procedure, 490
KCLUS procedure, 148
TREESPLIT procedure, 452
VARIMPUTE procedure, 539
INTERCEPT option
MODEL statement (PLSMOD), 307
IPRED= option
OUTPUT statement (GENSELECT), 100
OUTPUT statement (LOGSELECT), 191
ITHIST option
PROC GENSELECT statement, 87
PROC LOGSELECT statement, 179
KCLUS procedure, CLASS statement
DESCENDING option, 11, 555
MISSING option, 11, 555
ORDER= option, 11
PARAM= option, 12
REF= option, 13
SPLIT option, 13
KCLUS procedure, CODE statement, 146
COMMENT option, 14
FILE= option, 15, 146
FORMATWIDTH= option, 15
INDENTSIZE= option, 15
LABELID= option, 15, 146
LINESIZE= option, 15
NOTRIM option, 15
PCATALL option, 15
KCLUS procedure, DISPLAY statement
CASESENSITIVE option, 16, 146
EXCLUDE option, 16, 147
EXCLUDEALL option, 16, 147
TRACE option, 16, 147
KCLUS procedure, DISPLAYOUT statement
NOREPLACE option, 17, 147
REPEATED option, 17, 148
KCLUS procedure, EFFECT statement
BASIS option (spline), 24
DATABOUNDARY option (spline), 24
DEGREE option (polynomial), 22
DEGREE option (spline), 25
DETAILS option (multimember), 20


DETAILS option (polynomial), 22
DETAILS option (spline), 25
KNOTMAX= option (spline), 25
KNOTMETHOD option (spline), 25
KNOTMIN= option (spline), 26
MDEGREE option (polynomial), 22
NATURALCUBIC option (spline), 26
NOEFFECT option (multimember), 20
NOSEPARATE option (polynomial), 22
SEPARATE option (spline), 26
SPLIT option (spline), 27
STANDARDIZE option (polynomial), 22
KCLUS procedure, INPUT statement, 148
   LEVEL= option, 148
KCLUS procedure, OUTPUT statement
   COPYVARS= option, 149
   OUT= option, 149
KCLUS procedure, PARTITION statement
   FRACTION option, 32
   ROLEVAR= option, 32
KCLUS procedure, PROC KCLUS statement, 141
   ABSCONV option, 38
   ABSFCONV option, 38
   ABSFTOL option, 38
   ABSGCONV option, 38
   ABSGTOL option, 38
   ABSTOL option, 38
   ABSXCONV option, 39
   ABSTOL option, 39
   DATA= option, 142
   DISTANCE= option, 142
   DISTANCENOM= option, 142
   FCONV option, 39
   FCONV2 option, 39
   FTOL option, 39
   FTOL2 option, 39
   GCONV option, 40
   GCONV2 option, 41
   GTOL option, 40
   GTOL2 option, 41
   IMPUTE= option, 143
   IMPUTENOM= option, 143
   INIT= option, 143
   MAXCLUSTERS= option, 143
   MAXFUNC= option, 41
   MAXITER= option, 41, 143
   MAXTIME= option, 41
   MINITER= option, 42
   NORMALIZE= option, 42
   NTHREADS, 145
   OUTSTAT= option, 145
   SEED= option, 145
   STANDARDIZE= option, 145
   STOPCRITERION= option, 144, 145
   TECHNIQUE= option, 42
   XCONV option, 42
   XTOL option, 42
KCLUS procedure, SCORE statement, 148
KCLUS procedure, SELECTION statement
   ADAPTIVE option, 34
   CHOOSE= option, 34
   COMPETITIVE option, 34
   CRITERION= option, 34
   DETAILS= option, 36
   FAST option, 35
   HIERARCHY= option, 37
   LSCEOFFS option, 35
   MAXEFFECTS= option, 35
   MAXSTEPS= option, 35
   METHOD= option, 33
   MINEFFECTS= option, 35
   ORDERSELECT option, 37
   SELECT= option, 35
   SELECTION= option, 37
   SLE= option, 35
   SLEENTRY= option, 35
   SLS= option, 35
   SLSTAY= option, 35
   STOP= option, 36
   STOPHORIZON= option, 37
KCLUS procedure, syntax, 141
   keyword option
     OUTPUT statement (PLSMOD), 308
   keyword= option
     OUTPUT statement (GENSELECT), 99
     OUTPUT statement (QTRSELECT), 346
     OUTPUT statement (REGSELECT), 389
KCLUS procedure, syntax, 141
   KFOLD= option
     AUTOTUNE statement, 428
   KNOTMAX= option
     EFFECT statement, spline (GENSELECT), 25
     EFFECT statement, spline (KCLUS), 25
     EFFECT statement, spline (LOGSELECT), 25
     EFFECT statement, spline (NLMOD), 25
     EFFECT statement, spline (PCA), 25
     EFFECT statement, spline (REGSELECT), 25
     EFFECT statement, spline (TREESPLIT), 25
   KNOTMETHOD option
     EFFECT statement, spline (GENSELECT), 25
     EFFECT statement, spline (KCLUS), 25
     EFFECT statement, spline (LOGSELECT), 25
     EFFECT statement, spline (NLMOD), 25
     EFFECT statement, spline (PCA), 25
     EFFECT statement, spline (REGSELECT), 25
     EFFECT statement, spline (TREESPLIT), 25
   KNOTMIN= option
     EFFECT statement, spline (GENSELECT), 26
     EFFECT statement, spline (KCLUS), 26
EFFECT statement, spline (LOGSELECT), 26
EFFECT statement, spline (NLMOD), 26
EFFECT statement, spline (PCA), 26
EFFECT statement, spline (REGSELECT), 26
EFFECT statement, spline (TREESPLIT), 26

LABELID= option
CODE statement, 146
CODE statement (GENSELECT), 15
CODE statement (KCLUS), 15
CODE statement (LOGSELECT), 15
CODE statement (NLMOD), 15
CODE statement (PCA), 15
CODE statement (REGSELECT), 15
CODE statement (TREESPLIT), 15

LASSORHO= option
PROC GENSELECT statement, 87
PROC LOGSELECT statement, 179

LASSOSTEPS= option
PROC GENSELECT statement, 87
PROC LOGSELECT statement, 179

LASSOTOL= option
PROC GENSELECT statement, 87
PROC LOGSELECT statement, 179

LEAVES= option
PRUNE statement (TREESPLIT), 435, 436

LEVEL= option
INPUT statement, 148
TARGET statement, 474

LIFOUT= option
PROC ASSESS statement, 471

LINESIZE= option
CODE statement (GENSELECT), 15
CODE statement (KCLUS), 15
CODE statement (LOGSELECT), 15
CODE statement (NLMOD), 15
CODE statement (PCA), 15
CODE statement (REGSELECT), 15
CODE statement (TREESPLIT), 15

LINK= option
MODEL statement (GENSELECT), 96
MODEL statement (LOGSELECT), 187

LINP= option
OUTPUT statement (GENSELECT), 102
OUTPUT statement (LOGSELECT), 192

LOGSELECT procedure, CODE statement, 181
COMMENT option, 14
FILE= option, 15
FORMATWIDTH= option, 15
INDENTSIZE= option, 15
LABELID= option, 15
LINESIZE= option, 15
NOTRIM option, 15
PCATALL option, 15

LOGSELECT procedure, DISPLAY statement
CASESENSITIVE option, 16, 182
EXCLUDE option, 16, 182
EXCLUDEALL option, 16, 182
TRACE option, 16, 182

LOGSELECT procedure, DISPLAYOUT statement
NOREPLACE option, 17, 183
REPEATED option, 17, 183

LOGSELECT procedure, EFFECT statement
BASIS option (spline), 24
DATABASEboundary option (spline), 24
DEGREE option (polynomial), 22
DEGREE option (spline), 25
DETAILS option (multimember), 20
DETAILS option (polynomial), 22
DETAILS option (spline), 25
KNOTMAX= option (spline), 25
KNOTMETHOD option (spline), 25
KNOTMIN= option (spline), 26
MDEGREE option (polynomial), 22
NATURALCUBIC option (spline), 26
NOEFFECT option (multimember), 20
NOSEPARATE option (polynomial), 22
SEPARATE option (spline), 26
SPLIT option (spline), 27
STANDARDIZE option (polynomial), 22

LOGSELECT procedure, MODEL statement, 185
CLB option, 187
DESCENDING option, 185
INCLUDE option, 187
INFORMATIVE option, 187
LINK= option, 187
NOCHECK option, 179
NOINT option, 188
OFFSET= option, 188
ORDER= option, 186
START option, 188
TYPE3 option, 188

LOGSELECT procedure, OUTPUT statement, 188
ALL option, 189
ALPHA= option, 189
CBAR= option, 190
COPYVAR= option, 189
DIFCHISQ= option, 190
DIFDEV= option, 190
H= option, 190
INDIVIDUAL= option, 191
IPRED= option, 191
LCL= option, 191
LCLM= option, 191
LINP= option, 192
LOWER= option, 191
LOWERXBETA= option, 191
OBSCAT option, 189
OUT= option, 189, 389
PREDICT= option, 191
PREDPROBS option, 189
RESCHI= option, 191
RESDEV= option, 191
RESIDUAL= option, 191
RESLIK= option, 191
RESLKL= option, 191
RESRAW= option, 191
RESWORK= option, 192
ROLE= option, 192
STDRESCHI= option, 192
STDRESDEV= option, 192
STDXBETA = option, 192
UCL= option, 192
UCLM= option, 192
UPPER= option, 192
UPPERXBETA= option, 192
XBETA= option, 192
LOGSELECT procedure, PARTITION statement
FRACTION option, 32, 193
ROLEVAR= option, 32, 193
LOGSELECT procedure, PROC LOGSELECT statement, 177
ABSCONV option, 38
ABSFCONV option, 38
ABSFTOL option, 38
ABSGCONV option, 38
ABSGTOL option, 38
ABSTOL option, 38
ABSXCONV option, 39
ABSXTOL option, 39
ALPHA= option, 178
CORRB option, 179
COVB option, 179
DATA= option, 179
FCONV option, 39
FCONV2 option, 39
FTOL option, 39
FTOL2 option, 39
GCONV option, 40
GCONV2 option, 41
GTOL option, 40
GTOL2 option, 41
ITHIST option, 179
LASSORHO= option, 179
LASSOSTEPS= option, 179
LASSOTOL= option, 179
MAXFUNC= option, 41
MAXITER= option, 41
MAXOPTBATCH= option, 179
MAXTIME= option, 41
MINITER= option, 42
NOCLPRINT option, 180
NORMALIZE= option, 42
NOSTDERR option, 180
PARFTIT option, 180
TECHNIQUE= option, 42
XCONV option, 42
XTOL option, 42
LOGSELECT procedure, SELECTION statement, 193
ADAPTIVE option, 34
CHOOSE= option, 34
COMPETITIVE option, 34
CRITERION= option, 34
DETAILS= option, 36
FAST option, 35
HIERARCHY= option, 37
LSCOEFFS option, 35
MAXEFFECTS= option, 35
MAXSTEPS= option, 35
METHOD= option, 33
MINEFFECTS= option, 35
ORDERSELECT option, 37
SELECT= option, 35
SELECTION= option, 37
SLE= option, 35
SLENTRY= option, 35
SLS= option, 35
SLSTAY= option, 35
STOP= option, 36
STOPHORIZON= option, 37
LOGSELECT procedure, syntax, 177
LOWER= option
OUTPUT statement (GENSELECT), 101
OUTPUT statement (LOGSELECT), 191
PREDICT statement (NLMOD), 240
LOWERXBETA= option
OUTPUT statement (GENSELECT), 101
OUTPUT statement (LOGSELECT), 191
LSCOEFFS option
SELECTION statement (GENSELECT), 35
SELECTION statement (KCLUS), 35
SELECTION statement (LOGSELECT), 35
SELECTION statement (NLMOD), 35
SELECTION statement (PCA), 35
<table>
<thead>
<tr>
<th>Option</th>
<th>Procedure/Statement</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>MATRIX= option</td>
<td>PROC VARREDUCE</td>
<td>554</td>
</tr>
<tr>
<td>MAXBRANCH= option</td>
<td>PROC TREESPLIT</td>
<td>424</td>
</tr>
<tr>
<td>MAXCLUSTERS= option</td>
<td>PROC KCLUS</td>
<td>143</td>
</tr>
<tr>
<td>MAXDEPTH= option</td>
<td>PROC TREESPLIT</td>
<td>424</td>
</tr>
<tr>
<td>MAXEFFECTS= option</td>
<td>REDUCE</td>
<td>559</td>
</tr>
<tr>
<td>MAXEVALS= option</td>
<td>AUTOTUNE</td>
<td>428</td>
</tr>
<tr>
<td>MAXFUNC= option</td>
<td>PROC GENSELECT</td>
<td>41</td>
</tr>
<tr>
<td>MAXLEVELS= option</td>
<td>PROC ASSESS</td>
<td>472</td>
</tr>
<tr>
<td>MAXITER= option</td>
<td>PROC PLSMOD</td>
<td>302</td>
</tr>
<tr>
<td>MDEGREE option</td>
<td>EFFECT</td>
<td>22</td>
</tr>
<tr>
<td>METHOD= option</td>
<td>PROC PCA</td>
<td>265</td>
</tr>
<tr>
<td>MINEFFECTS= option</td>
<td>PROC REGSELECT</td>
<td>35</td>
</tr>
<tr>
<td>MINTITER= option</td>
<td>PROC CARDINALITY</td>
<td>509</td>
</tr>
<tr>
<td></td>
<td>PROC GENSELECT</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td>PROC KCLUS</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td>PROC LOGSELECT</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td>PROC NLMOD</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td>PROC PCA</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td>PROC REGSELECT</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td>PROC TREESPLIT</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td>PROC TREESPLIT</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td>PROC TREESPLIT</td>
<td>41</td>
</tr>
</tbody>
</table>
MINLEAFSIZE= option
  PROC TREESPLIT statement, 424
MINUSEINSEARCH= option
  PROC TREESPLIT statement, 424
MINVARIANCEINCREMENT= option
  REDUCE statement, 559
MISSING option
  CLASS statement (GENSELECT), 11, 555
  CLASS statement (KCLUS), 11, 555
  CLASS statement (LOGSELECT), 11, 555
  CLASS statement (NLMOD), 11, 555
  CLASS statement (PCA), 11, 555
  CLASS statement (REGSELECT), 11, 555
  CLASS statement (TREESPLIT), 11, 555

MODEL statement
  GENSELECT procedure, 92
  LOGSELECT procedure, 185
  NLMOD procedure, 236
  PLSMOD procedure, 307
  QTRSELECT procedure, 344
  REGSELECT procedure, 387
  TREESPLIT procedure, 433

N= option
  PROC PCA statement, 266
NATURALCUBIC option
  EFFECT statement, spline (GENSELECT), 26
  EFFECT statement, spline (KCLUS), 26
  EFFECT statement, spline (LOGSELECT), 26
  EFFECT statement, spline (NLMOD), 26
  EFFECT statement, spline (PCA), 26
  EFFECT statement, spline (REGSELECT), 26
  EFFECT statement, spline (TREESPLIT), 26

NBINS= option
  PROC ASSESS statement, 472
NCUTS= option
  PROC ASSESS statement, 472
NFAC= option
  PROC PLSMOD statement, 302
NLMOD procedure, 231
  PROC NLMOD statement, 232
  syntax, 231
NLMOD procedure, BOUNDS statement, 234
NLMOD procedure, CLASS statement
  DESCENDING option, 11, 555
  MISSING option, 11, 555
  ORDER= option, 11
  PARAM= option, 12
  REF= option, 13
  SPLIT option, 13
NLMOD procedure, CODE statement
  COMMENT option, 14
  FILE= option, 15
  FORMATWIDTH= option, 15
  INDENTSIZE= option, 15
  LABELID= option, 15
  LINESIZE= option, 15
  NOTRIM option, 15
  PCATALL option, 15
NLMOD procedure, DISPLAY statement
  CASESENSITIVE option, 16, 234
  EXCLUDE option, 16, 235
  EXCLUDEALL option, 16, 235
  TRACE option, 16, 235
NLMOD procedure, DISPLAYOUT statement
  NOREPLACE option, 17, 235
  REPEATED option, 17, 235
NLMOD procedure, EFFECT statement
  BASIS option (spline), 24
  DENSITYBOUNDARY option (spline), 24
  DEGREE option (polynomial), 22
  DEGREE option (spline), 25
  DETAILS option (multimember), 20
  DETAILS option (polynomial), 22
  DETAILS option (spline), 25
  KNOTMAX= option (spline), 25
  KNOTMETHOD option (spline), 25
  KNOTMIN= option (spline), 26
  MDEGREE option (polynomial), 22
  NATURALCUBIC option (spline), 26
  NOEFFECT option (multimember), 20
  NOSEPARATE option (polynomial), 22
  SEPARATE option (spline), 26
  SPLIT option (spline), 27
  STANDARDIZE option (polynomial), 22
NLMOD procedure, ESTIMATE statement, 236
  ALPHA= option, 236
  DF= option, 236
NLMOD procedure, MODEL statement, 236
NLMOD procedure, PARAMETERS statement, 237
NLMOD procedure, PARTITION statement
  FRACTION option, 32
  ROLEVAR= option, 32
NLMOD procedure, PREDICT statement, 240
  ALPHA= option, 240
  DF= option, 240
  LOWER= option, 240
  PRED= option, 240
  PROBT= option, 240
  STDERR= option, 240
  TVALEU= option, 240
  UPPER= option, 240
NLMOD procedure, PROC NLMOD statement, 232
  ABSCONV option, 38
  ABSFCONV option, 38
  ABSFTOL option, 38
  ABSGCONV option, 38
  ABSGTOL option, 38
ABSTOL option, 38
ABSXCONV option, 39
ABSXTOL option, 39
ALPHA= option, 233
CORR option, 233
COV option, 233
DATA= option, 233
DF= option, 233
ECORR option, 233
ECOV option, 233
FCONV option, 39
FCONV2 option, 39
FTOL option, 39
FTOL2 option, 39
GCONV option, 40
GCONV2 option, 41
GTOL option, 40
GTOL2 option, 41
MAXFUNC= option, 41
MAXITER= option, 41
MAXTIME= option, 41
MINITER= option, 42
NOITPRINT option, 233
NOPRINT option, 233
NORMALIZE= option, 42
OUT= option, 233
SINGULAR= option, 233
TECHNIQUE= option, 42
XCONV option, 42
XTOL option, 42
NLMOD procedure, RESTRICT statement, 241
NLMOD procedure, SELECTION statement
ADAPTIVE option, 34
CHOOSE= option, 34
COMPETITIVE option, 34
CRITERION= option, 34
DETAILS= option, 36
FAST option, 35
HIERARCHY= option, 37
LSCOEFFS option, 35
MAXEFFECTS= option, 35
MAXSTEPS= option, 35
METHOD= option, 33
MINEFFECTS= option, 35
ORDERSELECT option, 37
SELECT= option, 35
SELECTION= option, 37
SLE= option, 35
SLENTRY= option, 35
SLS= option, 35
SLSTAY= option, 35
STOP= option, 36
STOPHORIZON= option, 37
NOCENTER option
PROC PCA statement, METHOD=ITERGS option, 266
PROC PCA statement, METHOD=NIPALS option, 266
PROC PLSMOD statement, 303
NOCLPRINT option
PROC GENSELECT statement, 88
PROC LOGSELECT statement, 180
PROC PLSMOD statement, 303
PROC QTRSELECT statement, 340
PROC REGSELECT statement, 382
NOCVSTDIZE option
PROC PLSMOD statement, 303
NOEFFECT option
EFFECT statement, multimember
(GENSELECT), 20
EFFECT statement, multimember (KCLUS), 20
EFFECT statement, multimember (LOGSELECT), 20
EFFECT statement, multimember (NLMOD), 20
EFFECT statement, multimember (PCA), 20
EFFECT statement, multimember (REGSELECT), 20
EFFECT statement, multimember (TREESPLIT), 20
NOINT option
MODEL statement (GENSELECT), 87, 97
MODEL statement (LOGSELECT), 179, 188
MODEL statement (QTRSELECT), 345
MODEL statement (REGSELECT), 388
PROC PCA statement, 266
NOITPRINT option
PROC NLMOD statement, 233
NOPRINT option
PROC NLMOD statement, 233
PROC TREESPLIT statement, 424
PROC VARREDUCE statement, 554
NOREPLACE option
DISPLAYOUT statement (GENSELECT), 17, 90
DISPLAYOUT statement (KCLUS), 17, 147
DISPLAYOUT statement (LOGSELECT), 17, 183
DISPLAYOUT statement (NLMOD), 17, 235
DISPLAYOUT statement (PARTITION), 527
DISPLAYOUT statement (PCA), 17, 271
DISPLAYOUT statement (PLSMOD), 305
DISPLAYOUT statement (QTRSELECT), 342
DISPLAYOUT statement (REGSELECT), 17, 385
DISPLAYOUT statement (TREESPLIT), 17
DISPLAYOUT statement (VARREDUCE), 557
NORMALIZE= option
PROC GENSELECT statement, 42
PROC KCLUS statement, 42
PROC LOGSELECT statement, 42
PROC NLMO... statement, 42
PROC PCA statement, 42
PROC REGSELECT statement, 42
PROC TREESPLIT statement, 42

NOSCALE option
PROC PCA statement, METHOD=ITERGS option, 266
PROC PCA statement, METHOD=NIPALS option, 266
PROC PLSMOD statement, 303

NOSEPARATE option
EFFECT statement, polynomial (GENSELECT), 22
EFFECT statement, polynomial (KCLUS), 22
EFFECT statement, polynomial (LOGSELECT), 22
EFFECT statement, polynomial (NLMO...), 22
EFFECT statement, polynomial (PCA), 22
EFFECT statement, polynomial (REGSELECT), 22
EFFECT statement, polynomial (TREESPLIT), 22

NOSTDERR option
PROC GENSELECT statement, 88
PROC LOGSELECT statement, 180

NOTRIM option
CODE statement (GENSELECT), 15
CODE statement (KCLUS), 15
CODE statement (LOGSELECT), 15
CODE statement (NLMO...), 15
CODE statement (PCA), 15
CODE statement (REGSELECT), 15
CODE statement (TREESPLIT), 15

NSAMP= option
PROC PLSMOD statement, CVTEST option, 301

NSURROGATES= option
PROC TREESPLIT statement, 425

NTAU option
QUANTILES option (QTRSELECT), 345

NTHREADS
PROC KCLUS statement, 145

NTHREADS= option
PROC ASSESS statement, 472
PROC PARTITION statement, 524
PROC VARIMPUTE statement, 538

NUMBIN= option
INPUT statement, 490
PROC BINNING statement, 489
PROC TREESPLIT statement, 424

OBSCAT option
OUTPUT statement (GENSELECT), 99
OUTPUT statement (LOGSELECT), 189

OFF option
PRUNE statement (TREESPLIT), 435

OFFSET= option
MODEL statement (GENSELECT), 97
MODEL statement (LOGSELECT), 188

ORDER= option
CLASS statement (GENSELECT), 11
CLASS statement (KCLUS), 11
CLASS statement (LOGSELECT), 11
CLASS statement (NLMO...), 11
CLASS statement (PCA), 11
CLASS statement (REGSELECT), 11
CLASS statement (TREESPLIT), 11
MODEL statement (GENSELECT), 94
MODEL statement (LOGSELECT), 186
PROC CARDINALITY statement, 509
VAR statement, 510

ORDERSELECT option
SELECTION statement (GENSELECT), 37
SELECTION statement (KCLUS), 37
SELECTION statement (LOGSELECT), 37
SELECTION statement (NLMO...), 37
SELECTION statement (PCA), 37
SELECTION statement (REGSELECT), 37
SELECTION statement (TREESPLIT), 37

OUT= option
OUTPUT statement (BINNING), 491
OUTPUT statement (GENSELECT), 98
OUTPUT statement (KCLUS), 149
OUTPUT statement (LOGSELECT), 189, 389
OUTPUT statement (PARTITION), 527
OUTPUT statement (PCA), 271
OUTPUT statement (PLSMOD), 307
OUTPUT statement (QTRSELECT), 346
OUTPUT statement (TREESPLIT), 434
OUTPUT statement (VARIMPUTE), 539
PROC NLMO... statement, 233

OUTCARD= option
PROC CARDINALITY statement, 508

OUTCP= option
PROC VARREDUCE statement, 554

OUTDETAILS= option
PROC CARDINALITY statement, 509

OUTMODEL option
PROC TREESPLIT statement, 425

OUTPUT statement
BINNING procedure, 490
GENSELECT procedure, 98
LOGSELECT procedure, 188
PARTITION procedure, 527
PCA procedure, 271
PLSMOD procedure, 307
QTRSELECT procedure, 346
REGSELECT procedure, 389
TREESPLIT procedure, 433
VARIMPUTE procedure, 539
OUTSTAT= option
PROC KCLUS statement, 145
PROC PCA statement, 266
PARAM= option
CLASS statement (GENSELECT), 12
CLASS statement (KCLUS), 12
CLASS statement (LOGSELECT), 12
CLASS statement (NLMOD), 12
CLASS statement (PCA), 12
CLASS statement (REGSELECT), 12
CLASS statement (TREESPLIT), 12
PARAMETERS statement
NLMOD procedure, 237
PARPREFIX= option
PROC PCA statement, 268
PARTFIT option
PROC GENSELECT statement, 88
PROC LOGSELECT statement, 180
PARTIAL statement
PCA procedure, 273
PARTIND option
PROC PARTITION statement, 524
PARTINDNAME= option
OUTPUT statement, 528
PARTITION procedure, BY statement, 525
PARTITION procedure, DISPLAY statement
CASESENSITIVE option, 526
EXCLUDE option, 526
EXCLUDEALL option, 526
TRACE option, 526
PARTITION procedure, DISPLAYOUT statement
NOREPLACE option, 527
REPEATED option, 527
PARTITION procedure, OUTPUT statement, 527
COPYVARS= option, 528
FREQNAME= option, 528
OUT= option, 527
PARTINDNAME= option, 528
PARTITION procedure, PROC PARTITION statement, 524
DATA= option, 524
EVENT= option, 525
EVENTPROP= option, 525
NTHREADS= option, 524
PARTIND option, 524
SAMPPCT2= option, 525
SAMPPCT= option, 525
SAMPPCTEVT= option, 525
SEED= option, 524
PARTITION procedure, syntax, 524
PARTITION statement
GENSELECT procedure, 32, 102
KCLUS procedure, 32
LOGSELECT procedure, 32, 193
NLMOD procedure, 32
PCA procedure, 32
PLSMOD procedure, 310
QTRSELECT procedure, 348
REGSELECT procedure, 32, 391
TREESPLIT procedure, 32, 434
PCA procedure
PROC PCA statement, 264
syntax, 264
PCA procedure, CLASS statement
DESCENDING option, 11, 555
MISSING option, 11, 555
ORDER= option, 11
PARAM= option, 12
REF= option, 13
SPLIT option, 13
PCA procedure, CODE statement, 269
COMMENT option, 14
FILE= option, 15
FORMATWIDTH= option, 15
INDENTSIZE= option, 15
LABELID= option, 15
LINESIZE= option, 15
NOTRIM option, 15
PCATALL option, 15
PCA procedure, DISPLAY statement
CASESENSITIVE option, 16, 270
EXCLUDE option, 16, 270
EXCLUDEALL option, 16, 270
TRACE option, 16, 270
PCA procedure, DISPLAYOUT statement
NOREPLACE option, 17, 271
REPEATED option, 17, 271
PCA procedure, EFFECT statement
BASIS option (spline), 24
DATABOUNDARY option (spline), 24
DEGREE option (polynomial), 22
DEGREE option (spline), 25
DETAILS option (multimember), 20
DETAILS option (polynomial), 22
DETAILS option (spline), 25
KNOTMAX= option (spline), 25
KNOTMETHOD option (spline), 25
KNOTMIN= option (spline), 26
MDEGREE option (polynomial), 22
NATURALCUBIC option (spline), 26
NOEFFECT option (multimember), 20
NOEFFECT option (polynomial), 22
SEPARATE option (spline), 26
SPLIT option (spline), 27
STANDARDIZE option (polynomial), 22
PCA procedure, OUTPUT statement, 271
  COPYVARS= option, 272
  keyword option, 272
  OUT= option, 271
PCA procedure, PARTIAL statement, 273
PCA procedure, PARTITION statement
  FRACTION option, 32
  ROLEVAR= option, 32
PCA procedure, PROC PCA statement, 264
  ABSCONV option, 38
  ABSFCNV option, 38
  ABSFTOL option, 38
  ABSGCONV option, 38
  ABSGTL option, 38
  ABSTOL option, 38
  ABSXCONV option, 39
  ABSXTOL option, 39
  COV option, 265
  COVARIANCE option, 265
  DATA= option, 265
  FCONV option, 39
  FCONV2 option, 39
  FTOL option, 39
  FTOL2 option, 39
  GCONV option, 40
  GCONV2 option, 41
  GTOL option, 40
  GTOL2 option, 41
  MAXFUNC= option, 41
  MAXITER= option, 41
  MAXTIME= option, 41
  METHOD= option, 265
  MINITER= option, 42
  N= option, 266
  NOINT option, 266
  NORMALIZE= option, 42
  OUTSTAT= option, 266
  PARPREFIX= option, 268
  PLOTS= option, 266
  PPREFIX= option, 268
  PREFIX= option, 268
  RPREFIX= option, 268
  SING= option, 268
  SINGULAR= option, 268
  STANDARD option, 268
  STD option, 268
  TECHNIQUE= option, 42
  VARDEF= option, 268
  XCONV option, 42
  XTOL option, 42
PCA procedure, PROC PCA statement,
  METHOD=ITERGS option
  EPSILON= option, 265
  MAXITER= option, 265
  NOCENTER option, 266
  NOSCALE option, 266
PCA procedure, PROC PCA statement,
  METHOD=NIPALS option
  EPSILON= option, 265
  MAXITER= option, 265
  NOCENTER option, 266
  NOSCALE option, 266
PCA procedure, SELECTION statement
  ADAPTIVE option, 34
  CHOOSE= option, 34
  COMPETITIVE option, 34
  CRITERION= option, 34
  DETAILS= option, 36
  FAST option, 35
  HIERARCHY= option, 37
  LSCOEFFS option, 35
  MAXEFFECTS= option, 35
  MAXSTEPS= option, 35
  METHOD= option, 33
  MINEFFECTS= option, 35
  ORDERSELECT option, 37
  SELECT= option, 35
  SELECTION= option, 37
  SLE= option, 35
  SLENTRY= option, 35
  SLS= option, 35
  SLSTAY= option, 35
  STOP= option, 36
  STOPHORIZON= option, 37
PCA procedure, VAR statement, 273
PCATALL option
  CODE statement (GENSELECT), 15
  CODE statement (KCLUS), 15
  CODE statement (LOGSELECT), 15
  CODE statement (NLMOD), 15
  CODE statement (PCA), 15
  CODE statement (REGSELECT), 15
  CODE statement (TREESPLIT), 15
PEVENT= option
  FITSTAT statement, 473
PHI= option
  MODEL statement (GENSELECT), 97
PLOTS= option
  PROC PCA statement, 266
  PROC TREESPLIT statement, 425
PLSMOD procedure
  syntax, 300
PLSMOD procedure, CLASS statement, 303
PLSMOD procedure, DISPLAY statement
  CASESENSITIVE option, 304
  EXCLUDE option, 304
  EXCLUDEALL option, 304
  TRACE option, 304
<table>
<thead>
<tr>
<th>Procedure</th>
<th>Statement</th>
<th>Option</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLSMOD</td>
<td>DISPLAYOUT</td>
<td>NOREPLACE</td>
<td>305</td>
</tr>
<tr>
<td>PLSMOD</td>
<td>MODEL</td>
<td>INTERCEPT</td>
<td>307</td>
</tr>
<tr>
<td>PLSMOD</td>
<td>SOLUTION</td>
<td>option</td>
<td>307</td>
</tr>
<tr>
<td>PLSMOD</td>
<td>OUTPUT</td>
<td>COPYVARS=</td>
<td>308</td>
</tr>
<tr>
<td>PLSMOD</td>
<td>OUTPUT</td>
<td>OUT=</td>
<td>307</td>
</tr>
<tr>
<td>PLSMOD</td>
<td>PARTITION</td>
<td>FRACTION</td>
<td>310</td>
</tr>
<tr>
<td>PLSMOD</td>
<td>PROC PLSMOD</td>
<td>CENSCALE</td>
<td>301</td>
</tr>
<tr>
<td>PLSMOD</td>
<td>PROC PLSMOD</td>
<td>DATA=</td>
<td>301</td>
</tr>
<tr>
<td>PLSMOD</td>
<td>PROC PLSMOD</td>
<td>DETAILS</td>
<td>302</td>
</tr>
<tr>
<td>PLSMOD</td>
<td>PROC PLSMOD</td>
<td>METHOD=</td>
<td>302</td>
</tr>
<tr>
<td>PLSMOD</td>
<td>PROC PLSMOD</td>
<td>NFAC=</td>
<td>302</td>
</tr>
<tr>
<td>PLSMOD</td>
<td>PROC PLSMOD</td>
<td>NOCENTER</td>
<td>303</td>
</tr>
<tr>
<td>PLSMOD</td>
<td>PROC PLSMOD</td>
<td>NOCLPRINT</td>
<td>303</td>
</tr>
<tr>
<td>PLSMOD</td>
<td>PROC PLSMOD</td>
<td>NOCVSTDIZE</td>
<td>303</td>
</tr>
<tr>
<td>PLSMOD</td>
<td>PROC PLSMOD</td>
<td>NOSCALE</td>
<td>303</td>
</tr>
<tr>
<td>PLSMOD</td>
<td>PROC PLSMOD</td>
<td>VARSS</td>
<td>303</td>
</tr>
<tr>
<td>PLSMOD</td>
<td>PROC PLSMOD</td>
<td>CVTEST</td>
<td>301</td>
</tr>
<tr>
<td>PLSMOD</td>
<td>PROC PLSMOD</td>
<td>NSAMP=</td>
<td>301</td>
</tr>
<tr>
<td>PLSMOD</td>
<td>PROC PLSMOD</td>
<td>PVAR=</td>
<td>301</td>
</tr>
<tr>
<td>PLSMOD</td>
<td>PROC PLSMOD</td>
<td>SEED=</td>
<td>301</td>
</tr>
<tr>
<td>PLSMOD</td>
<td>PROC PLSMOD</td>
<td>STAT=</td>
<td>301</td>
</tr>
<tr>
<td>PLSMOD</td>
<td>PROC PLSMOD</td>
<td>ALGORITHM=</td>
<td>option</td>
</tr>
<tr>
<td>PLSMOD</td>
<td>PROC PLSMOD</td>
<td>EPSILON=</td>
<td>302</td>
</tr>
<tr>
<td>PLSMOD</td>
<td>PROC PLSMOD</td>
<td>MAXITER=</td>
<td>302</td>
</tr>
<tr>
<td>PLSMOD</td>
<td>PROC PLSMOD</td>
<td>POPSIZE=</td>
<td>option</td>
</tr>
<tr>
<td>PLSMOD</td>
<td>PROC PLSMOD</td>
<td>AUTOTUNE</td>
<td>statement</td>
</tr>
<tr>
<td>PLSMOD</td>
<td>PROC PLSMOD</td>
<td>PPREFIX=</td>
<td>option</td>
</tr>
<tr>
<td>PROC PCA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROC PCA</td>
<td></td>
<td>PRED=</td>
<td>option</td>
</tr>
<tr>
<td>PROC PCA</td>
<td></td>
<td>PREDICT</td>
<td>statement</td>
</tr>
<tr>
<td>PROC PLSMOD</td>
<td>CVTEST</td>
<td>option</td>
<td>301</td>
</tr>
<tr>
<td>PROC QTRSELECT</td>
<td></td>
<td>QTRSELECT</td>
<td>statement</td>
</tr>
<tr>
<td>PROC QTRSELECT</td>
<td></td>
<td>CASESENSITIVE</td>
<td>option</td>
</tr>
<tr>
<td>PROC QTRSELECT</td>
<td></td>
<td>EXCLUDE</td>
<td>option</td>
</tr>
<tr>
<td>PROC QTRSELECT</td>
<td></td>
<td>EXCLUDEALL</td>
<td>option</td>
</tr>
<tr>
<td>PROC QTRSELECT</td>
<td></td>
<td>TRACE</td>
<td>option</td>
</tr>
<tr>
<td>PROC QTRSELECT</td>
<td></td>
<td>DISPLAY</td>
<td>statement</td>
</tr>
<tr>
<td>PROC QTRSELECT</td>
<td></td>
<td>NOREPLACE</td>
<td>option</td>
</tr>
<tr>
<td>PROC QTRSELECT</td>
<td></td>
<td>REPEATED</td>
<td>option</td>
</tr>
<tr>
<td>PROC QTRSELECT</td>
<td></td>
<td>CLB</td>
<td>option</td>
</tr>
<tr>
<td>PROC NLMOD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROC NLMOD</td>
<td></td>
<td>PREDICT</td>
<td>statement</td>
</tr>
<tr>
<td>PROC ASSESS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROC ASSESS</td>
<td></td>
<td></td>
<td>see ASSESS procedure</td>
</tr>
<tr>
<td>PROC PCA</td>
<td></td>
<td></td>
<td>see PCA procedure</td>
</tr>
<tr>
<td>PROC PLSMOD</td>
<td></td>
<td></td>
<td>see PLSMOD procedure</td>
</tr>
<tr>
<td>PROC QTRSELECT</td>
<td></td>
<td></td>
<td>see QTRSELECT procedure</td>
</tr>
<tr>
<td>PROC VARIMPUTE</td>
<td></td>
<td></td>
<td>see VARIMPUTE procedure</td>
</tr>
<tr>
<td>PROC VARREDUCE</td>
<td></td>
<td></td>
<td>see VARREDUCE procedure</td>
</tr>
<tr>
<td>PROC TREESPLIT</td>
<td></td>
<td></td>
<td>see TREESPLIT procedure</td>
</tr>
<tr>
<td>PROC QTRSELECT</td>
<td></td>
<td></td>
<td>see QTRSELECT procedure</td>
</tr>
<tr>
<td>PROC QTRSELECT</td>
<td></td>
<td></td>
<td>DISPLAY statement</td>
</tr>
<tr>
<td>PROC QTRSELECT</td>
<td></td>
<td></td>
<td>CASESENSITIVE option</td>
</tr>
<tr>
<td>PROC QTRSELECT</td>
<td></td>
<td></td>
<td>EXCLUDE option</td>
</tr>
<tr>
<td>PROC QTRSELECT</td>
<td></td>
<td></td>
<td>EXCLUDEALL option</td>
</tr>
<tr>
<td>PROC QTRSELECT</td>
<td></td>
<td></td>
<td>TRACE option</td>
</tr>
<tr>
<td>PROC QTRSELECT</td>
<td></td>
<td></td>
<td>DISPLAYOUT statement</td>
</tr>
<tr>
<td>PROC QTRSELECT</td>
<td></td>
<td></td>
<td>NOREPLACE option</td>
</tr>
<tr>
<td>PROC QTRSELECT</td>
<td></td>
<td></td>
<td>REPEATED option</td>
</tr>
<tr>
<td>PROC QTRSELECT</td>
<td></td>
<td></td>
<td>CLB option</td>
</tr>
<tr>
<td>PROC QTRSELECT</td>
<td></td>
<td></td>
<td>INCLUDE option</td>
</tr>
</tbody>
</table>
INFORMATIVE option, 345
NOINT option, 345
QUANTILES option, 345
START option, 345
QTRSELECT procedure, MODEL statement,
QUANTILES option
NTAU option, 345
SORT option, 346
QTRSELECT procedure, OUTPUT statement, 346
COPYV AR= option, 346
keyword= option, 346
OUT= option, 346
QTRSELECT procedure, PARTITION statement
FRACTION option, 348
ROLEV AR= option, 348
QTRSELECT procedure, PROC QTRSELECT
statement, 339
ALPHA= option, 340
COV=SPARSITY option, 340
DATA= option, 340
NOCLPRINT option, 340
QTRSELECT procedure, SELECTION statement, 349
QUANTILES option
MODEL statement (QTRSELECT), 345
REDUCE statement
VARREDUCE procedure, 557
REDUCERROR option
PRUNE statement (TREESPLIT), 435
REF= option
CLASS statement (GENSELECT), 13
CLASS statement (KCLUS), 13
CLASS statement (LOGSELECT), 13
CLASS statement (NLMOD), 13
CLASS statement (PCA), 13
CLASS statement (REGSELECT), 13
CLASS statement (TREESPLIT), 13
REGSELECT procedure
MODEL statement, 387
OUTPUT statement, 389
PROC REGSELECT statement, 381
REGSELECT procedure, CLASS statement, 382
DESCENDING option, 11, 555
MISSING option, 11, 555
ORDER= option, 11
PARAM= option, 12
REF= option, 13
SPLIT option, 13
REGSELECT procedure, CODE statement, 383
COMMENT option, 14
FILE= option, 15
FORMATWIDTH= option, 15
INDENTSIZE= option, 15
LABELID= option, 15
LINESIZE= option, 15
NOTRIM option, 15
PCATALL option, 15
REGSELECT procedure, DISPLAY statement
CASESENSITIVE option, 16, 384
EXCLUDE option, 16, 384
EXCLUDEALL option, 16, 384
TRACE option, 16, 384
REGSELECT procedure, DISPLAYOUT statement
NOREPLACE option, 17, 385
REPEATED option, 17, 385
REGSELECT procedure, EFFECT statement
BASIS option (spline), 24
DATABOUNDARY option (spline), 24
DEGREE option (polynomial), 22
DEGREE option (spline), 25
DETAILS option (multimember), 20
DETAILS option (polynomial), 22
DETAILS option (spline), 25
KNOTMAX= option (spline), 25
KNOTMETHOD option (spline), 25
MDEGREE option (polynomial), 22
NATURALCUBIC option (spline), 26
NOEFFECT option (multimember), 20
NOSEPARATE option (polynomial), 22
SEPARATE option (spline), 26
SPLIT option (spline), 27
STANDARDIZE option (polynomial), 22
REGSELECT procedure, MODEL statement, 387
CLB option, 387
INCLUDE option, 387
INFORMATIVE option, 388
NOINT option, 388
SS3 option, 388
START option, 388
STB option, 388
TOL option, 388
VIF option, 388
REGSELECT procedure, OUTPUT statement, 389
COPYV AR= option, 389
keyword= option, 389
REGSELECT procedure, PARTITION statement
FRACTION option, 32, 391
ROLEV AR= option, 32, 391
REGSELECT procedure, PROC REGSELECT
statement, 381
ABSCONV option, 38
ABSFCONV option, 38
ABSFTOL option, 38
ABSGCONV option, 38
ABSGTOL option, 38
ABSTOL option, 38
ABSXCONV option, 39
ABSXTOL option, 39
ALPHA= option, 382
DATA= option, 382
FCONV option, 39
FCONV2 option, 39
FTOL option, 39
FTOL2 option, 39
GCONV option, 40
GCONV2 option, 41
GTOL option, 40
GTOL2 option, 41
MAXFUNC= option, 41
MAXITER= option, 41
MINITER= option, 42
NOCLPRINT option, 382
NORMALIZE= option, 42
TECHNIQUE= option, 42
XCONV option, 42
XTOL option, 42
REGSELECT procedure, SELECTION statement, 391
ADAPTIVE option, 34
CHOOSE= option, 34
COMPETITIVE option, 34
CRITERION= option, 34
DETAILS= option, 36
FAST option, 35
HIERARCHY= option, 37
LSCOEFFS option, 35
MAXEFFECTS= option, 35
MAXSTEPS= option, 35
METHOD= option, 33
MINEFFECTS= option, 35
ORDERSELECT option, 37
SELECT= option, 35
SELECTION= option, 37
SLE= option, 35
SLENTRY= option, 35
SLS= option, 35
SLSTAY= option, 35
STOP= option, 36
STOPHORIZON= option, 37
REPEATED option
DISPLAYOUT statement (GENSELECT), 17, 90
DISPLAYOUT statement (KCLUS), 17, 148
DISPLAYOUT statement (LOGSELECT), 17, 183
DISPLAYOUT statement (NLMOD), 17, 235
DISPLAYOUT statement (PARTITION), 527
DISPLAYOUT statement (PCA), 17, 271
DISPLAYOUT statement (PLSMOD), 305
DISPLAYOUT statement (QTRSELECT), 342
DISPLAYOUT statement (REGSELECT), 17, 385
DISPLAYOUT statement (TREESPLIT), 17
DISPLAYOUT statement (VARREDUCE), 557
RESCHI= option
OUTPUT statement (GENSELECT), 101
OUTPUT statement (LOGSELECT), 191
RESDEV= option
OUTPUT statement (GENSELECT), 101
OUTPUT statement (LOGSELECT), 191
RESIDUAL= option
OUTPUT statement (GENSELECT), 101
OUTPUT statement (LOGSELECT), 191
RESLIK= option
OUTPUT statement (GENSELECT), 101
OUTPUT statement (LOGSELECT), 191
RESRAW= option
OUTPUT statement (GENSELECT), 101
OUTPUT statement (LOGSELECT), 191
RESWORK= option
OUTPUT statement (GENSELECT), 101
OUTPUT statement (LOGSELECT), 192
ROCUOUT= option
PROC ASSESS statement, 472
ROLE= option
OUTPUT statement (GENSELECT), 101
OUTPUT statement (LOGSELECT), 192
ROLEVAR= option
PARTITION statement (GENSELECT), 32, 103
PARTITION statement (KCLUS), 32
PARTITION statement (LOGSELECT), 32, 193
PARTITION statement (NLMOD), 32
PARTITION statement (PCA), 32
PARTITION statement (QTRSELECT), 348
PARTITION statement (REGSELECT), 32, 391
PARTITION statement (TREESELECT), 32, 434
PLSMOD procedure, PARTITION statement, 310
RPREFIX= option
PROC PCA statement, 268
RSS option
PROC PCA statement, 268
PROC PCA statement, 268
GROW statement (TREESPLIT), 433
SAMPPCT2= option
PROC PARTITION statement, 525
SAMPPCT= option
PROC PARTITION statement, 525
SAMPPCTEVT= option
PROC PARTITION statement, 525
SCORE statement
KCLUS procedure, 148
SEED= option
PROC KCLUS statement, 145
PROC PARTITION statement, 524
PROC PLSMOD statement, CVTEST option, 301
PROC TREESPLIT statement, 427
PROC VARIMPUTE statement, 538

SELECT= option
  SELECTION statement (GENSELECT), 35
  SELECTION statement (KCLUS), 35
  SELECTION statement (LOGSELECT), 35
  SELECTION statement (NLMOD), 35
  SELECTION statement (PCA), 35
  SELECTION statement (REGSELECT), 35
  SELECTION statement (TREESPLIT), 35

SELECTION statement
  GENSELECT procedure, 32, 103
  KCLUS procedure, 32
  LOGSELECT procedure, 32, 193
  NLMOD procedure, 32
  PCA procedure, 32
  QTRSELECT procedure, 349
  REGSELECT procedure, 32, 391
  TREESPLIT procedure, 32

SELECT= option
  SELECTION statement (GENSELECT), 37
  SELECTION statement (KCLUS), 37
  SELECTION statement (LOGSELECT), 37
  SELECTION statement (NLMOD), 37
  SELECTION statement (PCA), 37
  SELECTION statement (REGSELECT), 37
  SELECTION statement (TREESPLIT), 37

SEPARATE option
  EFFECT statement, spline (GENSELECT), 26
  EFFECT statement, spline (KCLUS), 26
  EFFECT statement, spline (LOGSELECT), 26
  EFFECT statement, spline (NLMOD), 26
  EFFECT statement, spline (PCA), 26
  EFFECT statement, spline (REGSELECT), 26
  EFFECT statement, spline (TREESPLIT), 26

SING= option
  PROC PCA statement, 268

SINGULAR= option
  PROC NLMOD statement, 233
  PROC PCA statement, 268

SLE= option
  SELECTION statement (GENSELECT), 35
  SELECTION statement (KCLUS), 35
  SELECTION statement (LOGSELECT), 35
  SELECTION statement (NLMOD), 35
  SELECTION statement (PCA), 35
  SELECTION statement (REGSELECT), 35
  SELECTION statement (TREESPLIT), 35

SLENTRY= option
  SELECTION statement (GENSELECT), 35
  SELECTION statement (KCLUS), 35
  SELECTION statement (LOGSELECT), 35
  SELECTION statement (NLMOD), 35
  SELECTION statement (PCA), 35
  SELECTION statement (REGSELECT), 35
  SELECTION statement (TREESPLIT), 35

SLS= option
  SELECTION statement (GENSELECT), 35
  SELECTION statement (KCLUS), 35
  SELECTION statement (LOGSELECT), 35
  SELECTION statement (NLMOD), 35
  SELECTION statement (PCA), 35
  SELECTION statement (REGSELECT), 35
  SELECTION statement (TREESPLIT), 35

SLSTAY= option
  SELECTION statement (GENSELECT), 35
  SELECTION statement (KCLUS), 35
  SELECTION statement (LOGSELECT), 35
  SELECTION statement (NLMOD), 35
  SELECTION statement (PCA), 35
  SELECTION statement (REGSELECT), 35
  SELECTION statement (TREESPLIT), 35

SOLUTION option
  MODEL statement (PLSMOD), 307

SORT option
  QUANTILES option (QTRSELECT), 346

SPLIT option
  CLASS statement (GENSELECT), 13
  CLASS statement (KCLUS), 13
  CLASS statement (LOGSELECT), 13
  CLASS statement (NLMOD), 13
  CLASS statement (PCA), 13
  CLASS statement (REGSELECT), 13
  CLASS statement (TREESPLIT), 13
  EFFECT statement, spline (GENSELECT), 27
  EFFECT statement, spline (KCLUS), 27
  EFFECT statement, spline (LOGSELECT), 27
  EFFECT statement, spline (NLMOD), 27
  EFFECT statement, spline (PCA), 27
  EFFECT statement, spline (REGSELECT), 27
  EFFECT statement, spline (TREESPLIT), 27

SPLITONCE option
  PROC TREESPLIT statement, 427

SS3 option
  MODEL statement (REGSELECT), 388

STANDARD option
  PROC PCA statement, 268

STANDARDIZE option
  EFFECT statement, polynomial (GENSELECT), 22
  EFFECT statement, polynomial (KCLUS), 22
  EFFECT statement, polynomial (LOGSELECT), 22
  EFFECT statement, polynomial (NLMOD), 22
  EFFECT statement, polynomial (PCA), 22
  EFFECT statement, polynomial (REGSELECT), 22
EFFECT statement, polynomial (TREESPLIT), 22

STANDARDIZE= option
   PROC KCLUS statement, 145
START option
   MODEL statement (GENSELECT), 98
   MODEL statement (LOGSELECT), 188
   MODEL statement (QTRSELECT), 345
   MODEL statement (REGSELECT), 388
STAT= option
   PROC PLSMOD statement, CVTEST option, 301
STB option
   MODEL statement (REGSELECT), 388
STD option
   PROC PCA statement, 268
STDERR= option
   PREDICT statement (NLMOD), 240
STDRESCHI= option
   OUTPUT statement (GENSELECT), 102
   OUTPUT statement (LOGSELECT), 192
STDRESDEV= option
   OUTPUT statement (GENSELECT), 102
   OUTPUT statement (LOGSELECT), 192
STDXBETA= option
   OUTPUT statement (GENSELECT), 102
   OUTPUT statement (LOGSELECT), 192
STOP= option
   SELECTION statement (GENSELECT), 36
   SELECTION statement (KCLUS), 36
   SELECTION statement (LOGSELECT), 36
   SELECTION statement (NLMOD), 36
   SELECTION statement (PCA), 36
   SELECTION statement (REGSELECT), 36
   SELECTION statement (TREESPLIT), 36
STOPCRITERION= option
   PROC KCLUS statement, 144, 145
STOPHORIZON= option
   SELECTION statement (GENSELECT), 37
   SELECTION statement (KCLUS), 37
   SELECTION statement (LOGSELECT), 37
   SELECTION statement (NLMOD), 37
   SELECTION statement (PCA), 37
   SELECTION statement (REGSELECT), 37
   SELECTION statement (TREESPLIT), 37

SVMACHINE procedure, OUTPUT statement
   COPYVARS= option, 540

syntax
   GENSELECT procedure, 85
   NLMOD procedure, 231

TARGET statement
   ASSESS procedure, 474
   BINNING procedure, 491
   TREESPLIT procedure, 452

TECHNIQUE= option
   PROC GENSELECT statement, 42
   PROC KCLUS statement, 42
   PROC LOGSELECT statement, 42
   PROC NLMOD statement, 42
   PROC PCA statement, 42
   PROC REGSELECT statement, 42
   PROC TREESPLIT statement, 42
   PROC VARREDUCE statement, 555

TOL option
   MODEL statement (REGSELECT), 388

TRACE option
   DISPLAY statement (GENSELECT), 16, 90
   DISPLAY statement (KCLUS), 16, 147
   DISPLAY statement (LOGSELECT), 16, 182
   DISPLAY statement (NLMOD), 16, 235
   DISPLAY statement (PARTITION), 526
   DISPLAY statement (PCA), 16, 270
   DISPLAY statement (PLSMOD), 304
   DISPLAY statement (QTRSELECT), 341
   DISPLAY statement (REGSELECT), 16, 384
   DISPLAY statement (TREESPLIT), 16
   DISPLAY statement (VARREDUCE), 556

TREESPLIT procedure
   AUTOTUNE statement, 427
   INPUT statement, 452
   MODEL statement, 433
   PROC TREESPLIT statement, 422
   TARGET statement, 452

TREESPLIT procedure, AUTOTUNE statement, 427
   MAXEVALS= option, 428
   MAXITER= option, 428
   MAXTIME= option, 428
   POPSIZE= option, 428
   TUNINGPARAMETERS= option, 429
   USEPARAMETERS= option, 430

TREESPLIT procedure, CLASS statement, 431
   DESCENDING option, 11, 555
   MISSING option, 11, 555
   ORDER= option, 11
   PARAM= option, 12
   REF= option, 13
   SPLIT option, 13

TREESPLIT procedure, CODE statement, 431
   COMMENT option, 14
   FILE= option, 15
   FORMATWIDTH= option, 15
   INDENTSIZE= option, 15
   LABELID= option, 15
   LINESIZE= option, 15
   NOTRIM option, 15
   PCATALL option, 15

TREESPLIT procedure, DISPLAY statement
   CASESENSITIVE option, 16
EXCLUDE option, 16
EXCLUDEALL option, 16
TRACE option, 16
TREESPLIT procedure, DISPLAYOUT statement
   NOREPLACE option, 17
   REPEATED option, 17
TREESPLIT procedure, EFFECT statement
   BASIS option (spline), 24
   DATABOUNDARY option (spline), 24
   DEGREE option (polynomial), 22
   DEGREE option (spline), 25
   DETAILS option (multimember), 20
   DETAILS option (polynomial), 22
   DETAILS option (spline), 25
   KNOTMAX= option (spline), 25
   KNOTMETHOD option (spline), 25
   KNOTMIN= option (spline), 26
   MDEGREE option (polynomial), 22
   NATURALCUBIC option (spline), 26
   NOEFFECT option (multimember), 20
   NOSEPARATE option (polynomial), 22
   SEPARATE option (spline), 26
   SPLIT option (spline), 27
   STANDARDIZE option (polynomial), 22
TREESPLIT procedure, GROW statement, 431
   ALPHA= option, 432
   BONFERRONI option, 432
   CHAID option, 432
   CHISQUARE option, 432
   ENTROPY option, 432
   FTEST option, 433
   GINI option, 432
   IGR option, 433
   RSS option, 433
TREESPLIT procedure, INPUT statement, 452
TREESPLIT procedure, MODEL statement, 433
TREESPLIT procedure, OUTPUT statement, 433
   COPYVARS= option, 434
   OUT= option, 434
TREESPLIT procedure, PARTITION statement
   FRACTION option, 32, 434
   ROLEVAR= option, 32, 434
TREESPLIT procedure, PROC TREESPLIT statement, 422
   ABSCONV option, 38
   ABSFCONV option, 38
   ABSTOL option, 38
   ABSGCONV option, 38
   ABSTOL option, 38
   ABSXCONV option, 39
   ABSXTOL option, 39
   ASSIGNMISSING= option, 422
   DATA= option, 423
   FCONV option, 39
   FCONV2 option, 39
   FTOL option, 39
   FTOL2 option, 39
   GCONV option, 40
   GCONV2 option, 41
   GTOL option, 40
   GTOL2 option, 41
   MAXBRANCH= option, 424
   MAXDEPTH= option, 424
   MAXFUNC= option, 41
   MAXITER= option, 41
   MAXTIME= option, 41
   MINITER= option, 42
   MINLEAFSIZE= option, 424
   MINUSEINSEARCH= option, 424
   NOPRINT option, 424
   NORMALIZE= option, 42
   NSURROGATES= option, 425
   NUMBIN= option, 424
   OUTMODEL option, 425
   PLOTS= option, 425
   PRUNINGTABLE option, 427
   SEED= option, 427
   SPLITONCE option, 427
   TECHNIQUE= option, 42
   XCONV option, 42
   XTOL option, 42
TREESPLIT procedure, PRUNE statement, 435
   ALPHA= option, 435
   C45 option, 435
   CONFIDENCE= option, 435
   COSTCOMPLEXITY option, 435
   LEAVES= option, 435, 436
   OFF option, 435
   REDUCEDERROR option, 435
TREESPLIT procedure, SELECTION statement
   ADAPTIVE option, 34
   CHOOSE= option, 34
   COMPETITIVE option, 34
   CRITERION= option, 34
   DETAILS= option, 36
   FAST option, 35
   HIERARCHY= option, 37
   LSCEOFFS option, 35
   MAXEFFECTS= option, 35
   MAXSTEPS= option, 35
   METHOD= option, 33
   MINEFFECTS= option, 35
   ORDERSELECT option, 37
   SELECT= option, 35
   SELECTION= option, 37
   SLE= option, 35
   SLENTRY= option, 35
VARREDUCE procedure, DISPLAY statement
CASESENSITIVE option, 556
EXCLUDE option, 556
EXCLUDEALL option, 556
TRACE option, 556
VARREDUCE procedure, DISPLAYOUT statement
NOREPLACE option, 557
REPEATED option, 557
VARREDUCE procedure, PROC VARREDUCE statement, 553
DATA= option, 553
MATRIX= option, 554
NOPRINT option, 554
OUTCP= option, 554
TECHNIQUE= option, 555
VARREDUCE procedure, REDUCE statement, 557
AIC option, 558
AICC option, 558
BIC option, 558
MAXEFFECTS= option, 559
MAXITER= option, 559
MINVARIANCEINCREMENT= option, 559
VAREXP= option, 559
VARIANCEEXPLAINED= option, 559
VARINC= option, 559
VARREDUCE procedure, syntax, 553
VARRSS option
PROC PLSMOD statement, 303
VIF option
MODEL statement (REGSELECT), 388

WEIGHT statement
GENSELECT procedure, 105
LOGSELECT procedure, 195
PCA procedure, 273
QTRSELECT procedure, 349
REGSELECT procedure, 392
WOE(WOEADJUST=) option
PROC BINNING statement, 489

XBETA= option
OUTPUT statement (GENSELECT), 102
OUTPUT statement (LOGSELECT), 192

XCONV option
PROC GENSELECT statement, 42
PROC KCLUS statement, 42
PROC LOGSELECT statement, 42
PROC NLMOD statement, 42
PROC PCA statement, 42
PROC REGSELECT statement, 42
PROC TREESPLIT statement, 42

XTOL option
PROC GENSELECT statement, 42
PROC KCLUS statement, 42
PROC LOGSELECT statement, 42
PROC NLMOD statement, 42
PROC PCA statement, 42
PROC REGSELECT statement, 42
PROC TREESPLIT statement, 42
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.