## Contents

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Shared Concepts</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>The CCOPULA Procedure</td>
<td>55</td>
</tr>
<tr>
<td>3</td>
<td>The CNTSELECT Procedure</td>
<td>69</td>
</tr>
<tr>
<td>4</td>
<td>The CPANEL Procedure</td>
<td>111</td>
</tr>
<tr>
<td>5</td>
<td>The CQLIM Procedure</td>
<td>171</td>
</tr>
<tr>
<td>6</td>
<td>The SEVSELECT Procedure</td>
<td>201</td>
</tr>
</tbody>
</table>

**Subject Index**

341

**Syntax Index**

349
# Chapter 1
## Shared Concepts

## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction to Shared Concepts</td>
<td>2</td>
</tr>
<tr>
<td>Using CAS Sessions and CAS Engine Librefs</td>
<td>2</td>
</tr>
<tr>
<td>Loading a SAS Data Set onto a CAS Server</td>
<td>3</td>
</tr>
<tr>
<td>Syntax Common to SAS Econometrics Procedures</td>
<td>4</td>
</tr>
<tr>
<td>CLASS Statement</td>
<td>4</td>
</tr>
<tr>
<td>EFFECT Statement</td>
<td>8</td>
</tr>
<tr>
<td>Collection Effects</td>
<td>10</td>
</tr>
<tr>
<td>Multimember Effects</td>
<td>10</td>
</tr>
<tr>
<td>Polynomial Effects</td>
<td>12</td>
</tr>
<tr>
<td>Spline Effects</td>
<td>15</td>
</tr>
<tr>
<td>Splines and Spline Bases</td>
<td>18</td>
</tr>
<tr>
<td>ODS Table Names</td>
<td>22</td>
</tr>
<tr>
<td>SELECTION Statement</td>
<td>23</td>
</tr>
<tr>
<td>Optimization Options</td>
<td>26</td>
</tr>
<tr>
<td>Details for SAS Econometrics Procedures</td>
<td>31</td>
</tr>
<tr>
<td>Levelization of Classification Variables</td>
<td>31</td>
</tr>
<tr>
<td>Specification and Parameterization of Model Effects</td>
<td>33</td>
</tr>
<tr>
<td>Effect Operators</td>
<td>34</td>
</tr>
<tr>
<td>GLM Parameterization of Classification Variables and Effects</td>
<td>36</td>
</tr>
<tr>
<td>Nonsingular Parameterization</td>
<td>40</td>
</tr>
<tr>
<td>Class Variable Parameterization with Unbalanced Designs</td>
<td>43</td>
</tr>
<tr>
<td>Model Selection Methods</td>
<td>44</td>
</tr>
<tr>
<td>Forward Selection</td>
<td>44</td>
</tr>
<tr>
<td>Backward Elimination</td>
<td>46</td>
</tr>
<tr>
<td>Stepwise Selection</td>
<td>46</td>
</tr>
<tr>
<td>Forward-Swap Selection</td>
<td>47</td>
</tr>
<tr>
<td>Informative Missingness</td>
<td>48</td>
</tr>
<tr>
<td>Multithreading</td>
<td>49</td>
</tr>
<tr>
<td>Choosing an Optimization Algorithm</td>
<td>50</td>
</tr>
<tr>
<td>First- or Second-Order Algorithms</td>
<td>50</td>
</tr>
<tr>
<td>Algorithm Descriptions</td>
<td>51</td>
</tr>
<tr>
<td>References</td>
<td>53</td>
</tr>
</tbody>
</table>
Introduction to Shared Concepts

SAS Econometrics 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 Econometrics Procedures” on page 4 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 Econometrics Procedures” on page 31 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:

  ```sas
  proc casutil sessref=mysess;
  load data=Sample casout="Sample";
  quit;
  
  The CASUTIL procedure can load data onto a CAS server more efficiently than the DATA step. For more information about the CASUTIL procedure, see *SAS Cloud Analytic Services: 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 SEVSELECT procedure statements use a data table that resides in the `mycas` caslib:
You can delete your data table by using the DELETE procedure as follows:

```sas
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 Econometrics Procedures

### CLASS Statement

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

This section applies to the following procedures: CNTSELECT, CPANEL, and SEVSELECT.

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 31.

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 1.1 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 1.1.

**Table 1.1** 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>
</tbody>
</table>
Table 1.1  continued

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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 1.2  Value of PARAM=

<table>
<thead>
<tr>
<th>Value of PARAM=</th>
<th>Coding</th>
</tr>
</thead>
<tbody>
<tr>
<td>EFFECT</td>
<td>Effect coding. The REF= option in the CLASS statement determines the reference level.</td>
</tr>
<tr>
<td>GLM</td>
<td>Less-than-full-rank reference cell coding. This <em>keyword</em> can be used only as a <em>global-option</em> 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>Cumulative parameterization for an ordinal CLASS variable</td>
</tr>
<tr>
<td>THERMOMETER</td>
<td></td>
</tr>
<tr>
<td>POLYNOMIAL</td>
<td>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>POLY</td>
<td></td>
</tr>
<tr>
<td>REFERENCE</td>
<td>Reference cell coding. The REF= option in the CLASS statement determines the reference level.</td>
</tr>
<tr>
<td>REF</td>
<td></td>
</tr>
<tr>
<td>ORTHHEFFECT</td>
<td>Orthogonalizes PARAM=EFFECT coding. The REF= option in the CLASS statement determines the reference level.</td>
</tr>
<tr>
<td>ORTHORDINAL</td>
<td>Orthogonalizes PARAM=ORDINAL coding</td>
</tr>
<tr>
<td>ORTHOTHERM</td>
<td></td>
</tr>
<tr>
<td>ORTHPOLY</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>ORTHREF</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 33.

**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 SEVSELECT run as follows:

```plaintext
proc sevselect data=mycas.data;
  loss y;
  class temp gender / split;
  scalemodel gender gender*temp;
run;
```

The two effects in the SCALEMODEL 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 SEVSELECT step is equivalent to the following:

```plaintext
proc sevselect data=mycas.data;
  loss y;
  scalemodel 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 SEVSELECT step:

```plaintext
proc sevselect data=mycas.data;
  loss y;
  class temp(split) gender;
  scalemodel 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 SEVSELECT step is equivalent to the following:
Chapter 1: Shared Concepts

### EFFECT Statement

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

This section applies to the following procedures: SEVSELECT.

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 36. 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 sevselect data=mycas.data;
  loss y;
  class A B;
  scalemodel 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 sevselect data=mycas.data;
  loss y;
  class A B;
  effect spl = spline(x);
  scalemodel 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 SCALEMODEL statement in the SEVSELECT procedure or the MODEL statement in other procedures.

You must specify the following arguments:

- `effect-name`  
  names the effect. This name can appear in only one EFFECT statement and cannot be the name of a variable in the input data set.
effect-type specifies the type of effect. You can specify the following effect-types:

**COLLECTION**
specifies a collection effect that defines one or more variables as a single effect that has multiple degrees of freedom. The variables in a collection are considered as a unit for purposes of estimation and inference. For more information, see the section “Collection Effects” on page 10.

**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 10.

**POLYNOMIAL | POLY**
specifies a multivariate polynomial effect in the specified numeric variables. For more information, see the section “Polynomial Effects” on page 12.

**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 15.

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 1.3 after a slash following the var-list.

**Table 1.3** 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>
</tbody>
</table>
Table 1.3 continued

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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 KNOTMIN= 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**

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

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

**Multimember Effects**

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

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

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

- nurses’ effects on patient recovery in hospitals
- teachers’ effects on student scores
- lineage effects in genetic studies
The levels of a multimember effect consist of the union of formatted values of the variables that define this effect. Each such level contributes one column to the design matrix. For each observation, the value that corresponds to each level of the multimember effect in the design matrix is the number of times that this level occurs for the observation.

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

<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 SEVSELECT steps yield the identical analysis:

proc sevselect data=mycas.data;
   loss y;
   effect MyPoly = polynomial(x1-x3/degree=2);
   scalemodel MyPoly;
run;

proc sevselect data=mycas.data;
   loss y;
   scalemodel 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 \( x_1, x_2, x_1^2, x_1 x_2, x_2^2, x_1 x_2, x_1^2 x_2 \) 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

\[ n = \text{number of observations used in the analysis} \]
\[ w = \text{weight variable} \]
\[ f = \text{frequency variable} \]
\[ x = \text{variable to be standardized} \]

\[ x_{(n)} = \max_{i=1}^{n}(x_i) \]
\[ x_{(1)} = \min_{i=1}^{n}(x_i) \]
\[ F = \text{sum of frequencies} \]
\[ = \sum_{i=1}^{n} f_i \]
\[ WF = \text{sum of weighted frequencies} \]
\[ = \sum_{i=1}^{n} w_i f_i \]

Table 1.4 shows how the center and scale are computed for each of the supported methods.

<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} = \frac{\sum_{i=1}^{n} f_i x_i}{F})</td>
<td>(\sqrt{\frac{\sum_{i=1}^{n} f_i (x_i - \bar{x})^2}{(F - 1)}})</td>
</tr>
<tr>
<td>WMOMENTS</td>
<td>(\bar{x}<em>W = \frac{\sum</em>{i=1}^{n} w_i f_i x_i}{WF})</td>
<td>(\sqrt{\frac{\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-opts`:

**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 18. You request a spline effect with the syntax

\begin{verbatim}
EFFECT name=SPLINE (var-list < / spline-options>) ;
\end{verbatim}

The variables in \textit{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\textit{ spline-options} after a slash (/):

\textbf{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.

\textbf{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\textit{ options}:

\textbf{NOINT} excludes the intercept column.
\textbf{NOPowers} excludes the intercept and polynomial columns.

By default, BASIS=BSPLINE.

\textbf{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 \(x_2\):

\[
\text{EFFECT spl = spline}(x_1 \ x_2 \ / \ \text{knotmethod}=\text{rangefractions}(0.1 \ 0.5 \ 0.75));
\]

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 18. 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:

\[
\text{EFFECT spl = spline}(x_1 \ x_2 \ / \ \text{separate});
\]

In procedures that support variable selection, such as the SEVSELECT 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 $d$.

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

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

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

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

If you remove the restriction that the knots of a spline must be distinct and allow repeated knots, then you can
obtain functions that have less smoothness and even discontinuities at the repeated knot location. For a spline
of degree $d$ and a repeated knot that has multiplicity $m \leq d$, the piecewise polynomials that join such a knot
are required to have only $d - m$ matching derivatives. Note that this increases the number of free parameters
by $m - 1$ but also decreases the number of distinct knots by $m - 1$. Hence the dimension of the vector space
of splines of degree $d$ with $n$ knots is still $n + d + 1$, provided that any repeated knot has a multiplicity less
than or equal to $d$. 
The EFFECT statement supports the commonly used *truncated power function* basis and *B-spline* basis. With exact arithmetic and by using the complete basis, you obtain the same fit with either of these bases. The following subsections provide details about constructing spline bases for the space of splines of degree \( d \) with \( n \) knots that satisfies \( k_1 \leq k_2 \leq k_3 \leq \cdots \leq k_n \).

**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 1.1 shows such functions for \( d = 1 \) and \( d = 3 \) with a knot at \( x = 1 \).

Figure 1.1 Truncated Power Functions with Knot at \( x = 1 \)

![Figure 1.1 Truncated Power Functions](image)

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 1.2** 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 1.2** Linear B-Spline Basis with Four Equally Spaced Interior Knots

**Figure 1.3** 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 1.3 Cubic B-Spline Basis with Four Equally Spaced Interior Knots

Figure 1.4 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 1.3, 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 1.4 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 1.5 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 1.5 Natural Cubic Spline Basis with Four Equally Spaced Knots](image)

ODS Table Names

Each constructed effect produces one or more informative tables when the DETAILS option is also specified. Each table has a name associated with it. This name is used to refer to the table by procedures that support a DISPLAY or DISPLAYOUT statement. You also use this name to refer to the table in ODS statements. These names are listed in Table 1.5.

<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>MMLevelInfo</td>
<td>Levels of multimember effects</td>
<td></td>
</tr>
<tr>
<td>PolyDetails</td>
<td>Number of variables and columns, polynomial degree,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>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>
SELECTION Statement

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

This section applies to the following procedures: SEVSELECT.

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 44. 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. The only SAS Viya procedure that supports this method is the SEVSELECT procedure.

Table 1.6 lists the applicable *method-options* for each of these methods.

**Table 1.6  Applicable method-options by method**

<table>
<thead>
<tr>
<th>method-option</th>
<th>FORWARD</th>
<th>BACKWARD</th>
<th>STEPWISE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHOOSE =</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>COMPETITIVE</td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>CRITERION =</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>MAXEFFECTS =</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAXSTEPS =</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>MINEFFECTS =</td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>SELECT =</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>STOP =</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 1.6, not all selection *method-options* are applicable to every *method*.

**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. 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.

**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 SCALEMODEL or MODEL statement when METHOD=FORWARD or METHOD=BACKWARD. The default is three times the number of effects when METHOD=STEPWISE.

**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=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. For supported criteria, see the chapter for the relevant procedure.

**STOP=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.

- **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: \texttt{METHOD=FORWARD}, \texttt{METHOD=BACKWARD}, or \texttt{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:

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

\textbf{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.

\textbf{SINGLECLASS} is the same as \texttt{HIERARCHY=SINGLE} except that only CLASS effects are subject to the hierarchy requirement.

By default, \texttt{HIERARCHY=NONE}.

\textbf{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 SCALEMODEL or MODEL statement.

\textbf{SELECTION=} 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 \texttt{STOP=AIC} and the sequence of AIC values at steps 1 to 6 of a selection are 10, 7, 4, 6, 5, 2. If \texttt{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 \texttt{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 \texttt{STOP=NONE} then the stop horizon value is ignored. By default, \texttt{STOPHORIZON=3}.

\textbf{Optimization Options}

This section applies to the following procedures: SEVSELECT.
The following options are typically available in the PROC statement or the NLOPTIOINS statement of the procedures in this book that perform optimizations:

**ABSCONV=r**

**ABSTOL=r**

specifies an absolute function value convergence criterion. For minimization, termination requires \( f(\theta^{(k)}) \leq r \). 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**

**ABSFTOL=r**

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

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

The same formula is used for the NMSIMP technique, but \( \theta^{(k)} \) is defined as the vertex with the lowest function value, and \( \theta^{(k-1)} \) is defined as the vertex with the highest function value in the simplex. The default value is \( r = 0 \).

**ABSGCONV=r**

**ABSGTOL=r**

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

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

This criterion is not used by the NMSIMP technique. The default value is \( r=1E^{-5} \).

**ABSEXCONV=r**

**ABSEXXTOL=r**

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

\[
\| \theta^{(k)} - \theta^{(k-1)} \|_2 \leq r
\]

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 \( \xi^{(k)} \) with the smallest function value to the other simplex points \( \theta_l^{(k)} \neq \xi^{(k)} \):

\[
\delta^{(k)} = \sum \| \theta_l^{(k)} - \xi^{(k)} \|_1
\]

The default is \( r=1E^{-8} \) for the NMSIMP technique and \( r = 0 \) otherwise.
**FCONV=r**

**FTOL=r**

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

\[
\frac{|f(\theta^{(k)}) - f(\theta^{(k-1)})|}{\max(|f(\theta^{(k-1)})|, \text{FSIZE})} \leq r
\]

where FSIZE is defined by the FSIZE= option. The same formula is used for the NMSIMP technique, but \(\theta^{(k)}\) is defined as the vertex with the lowest function value, and \(\theta^{(k-1)}\) is defined as the vertex with the highest function value in the simplex.

The default value is \(r = 2\epsilon\), where \(\epsilon\) denotes the machine precision constant, which is the smallest double-precision floating-point number such that \(1 + \epsilon > 1\).

**FCONV2=r**

**FTOL2=r**

specifies another function convergence criterion.

For all techniques except NMSIMP, termination requires a small predicted reduction of the objective function:

\[
df^{(k)} \approx f(\theta^{(k)}) - f(\theta^{(k)} + s^{(k)})
\]

The predicted reduction

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

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

\[
\leq r
\]

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 \(\theta^{(k)}_l, l = 0, \ldots, p\),

\[
\sqrt{\frac{1}{p+1} \sum_l \left[ f(\theta^{(k)}_l) - \overline{f}^{(k)}(\theta^{(k)}) \right]^2} \leq r
\]

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

The default value is \(r=1\times10^{-6}\) for the NMSIMP technique and \(r = 0\) otherwise.

**FSIZE=r**

specifies the FSIZE parameter of the relative function and relative gradient termination criteria. The default value is \(r = 0\). For more information, see the FCONV= and GCONV= options.
specifies a relative gradient convergence criterion. For all techniques except CONGRA and NMSIMP, termination requires that the normalized predicted function reduction is small,

\[
\frac{g(\theta^{(k)})^T [H^{(k)}]^{-1} g(\theta^{(k)})}{\max(|f(\theta^{(k)})|, \text{FSIZE})} \leq r
\]

where FSIZE is defined by the FSIZE= option. For the CONGRA technique (where a reliable Hessian estimate \( H \) is not available), the following criterion is used:

\[
\frac{\| g(\theta^{(k)}) \|_2^2 \| s(\theta^{(k)}) \|_2}{\| g(\theta^{(k)}) - g(\theta^{(k-1)}) \|_2 \max(|f(\theta^{(k)})|, \text{FSIZE})} \leq r
\]

This criterion is not used by the NMSIMP technique. The default value is \( r = 1 \times 10^{-8} \).

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

\[
\max_j \frac{|g_j(\psi^{(k)})|}{\sqrt{f(\psi^{(k)})H_{jj}^{(k)}}} \leq r
\]

This criterion is not used by the other techniques.

By default, GCONV2=0.

specifies the maximum number \( n \) of function calls in the optimization process. The default values are

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

Note that the optimization can terminate only after completing a full iteration. Therefore, the number of function calls that is actually performed can exceed the number that you specify by the MAXFUNC= option.

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.

**TECHNIQUE=**\( \text{technique} \)**

**TECH=**\( \text{technique} \)**

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.
- **NEWRAP** performs a Newton-Raphson optimization with line search.
- **NMSIMP** performs a Nelder-Mead simplex optimization.
- **NONE** performs no optimization. This option can be used as follows:
  - to perform a grid search without optimization
  - to compute estimates and predictions that cannot be obtained efficiently with any of the optimization techniques
- **NRRIDG** performs a Newton-Raphson optimization with ridging.
- **QUANEW** performs a dual quasi-Newton optimization.
- **TRUREG** performs a trust-region optimization

The default method varies by the procedure. For the SEVSELECT procedure, the default is TECHNIQUE=TRUREG.

For more information, see the section “Choosing an Optimization Algorithm” on page 50.

**XCONV=**\( r \)**

**XTOL=**\( r \)**

specifies the relative parameter convergence criterion. For all techniques except NMSIMP, termination requires a small relative parameter change in subsequent iterations:

\[
\frac{\max_j |\theta_j^{(k)} - \theta_j^{(k-1)}|}{\max(|\theta_j^{(k)}|, |\theta_j^{(k-1)}|, \text{XSIZE})} \leq r
\]

For the NMSIMP technique, the same formula is used, but \( \theta_j^{(k)} \) is defined as the vertex that has the lowest function value and \( \theta_j^{(k-1)} \) is defined as the vertex that has the highest function value in the simplex. The default value is \( r=1E-8 \) for the NMSIMP technique and \( r = 0 \) otherwise.
Levelization of Classification Variables

_XSIZE=_

specifies the _XSIZE_ parameter of the relative parameter termination criterion. The value of _r_ must be greater than or equal to 0; the default is _r_ = 0. For more information, see the _XCONV_ option.

Details for SAS Econometrics Procedures

Levelization of Classification Variables

This section applies to the following procedures: CNTSELECT, CPANEL, and SEVSELECT.

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 1.7. 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 1.7 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 1.7 leads to the level assignment in Table 1.8.
Chapter 1: Shared Concepts

Table 1.8  Values and Levels

<table>
<thead>
<tr>
<th>Obs</th>
<th>Value</th>
<th>Level</th>
<th>Value</th>
<th>Level</th>
<th>FORMAT x 3.0</th>
<th>Value</th>
<th>Level</th>
<th>FORMAT x 3.1</th>
<th>Value</th>
<th>Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1.09</td>
<td>1</td>
<td>1</td>
<td>1.1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1.13</td>
<td>2</td>
<td>1</td>
<td>1.1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1.27</td>
<td>3</td>
<td>1</td>
<td>1.3</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>2</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>2</td>
<td>2.26</td>
<td>4</td>
<td>2</td>
<td>2.3</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>2</td>
<td>2.48</td>
<td>5</td>
<td>2</td>
<td>2.5</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>3</td>
<td>3.34</td>
<td>7</td>
<td>3</td>
<td>3.3</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>3</td>
<td>3.34</td>
<td>7</td>
<td>3</td>
<td>3.3</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td>3</td>
<td>3.14</td>
<td>6</td>
<td>3</td>
<td>3.1</td>
<td>5</td>
<td></td>
<td></td>
<td></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 1.9 shows how values of the ORDER= option are interpreted.

Table 1.9  Interpretation of Values of ORDER= Option

<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 1.10 displays the results of levelizing the values in Table 1.7 when the MISSING option is in effect.
### Table 1.10 Values and Levels When the MISSING Option Is Specified

<table>
<thead>
<tr>
<th>Obs</th>
<th>Value</th>
<th>Level</th>
<th>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>1</td>
<td>2</td>
<td>1</td>
<td>1.09</td>
<td>2</td>
<td>1.1</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<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></td>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</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>6</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>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>3.34</td>
<td>8</td>
<td>3.3</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</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: CNTSELECT, CPANEL, and SEVSELECT.

Procedures in this book that have a MODEL or SCALEMODEL 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 8.

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 scale regression models as in the SEVSELECT procedure or count regression models as in the CNTSELECT 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 31. 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 34.

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 36 and “Nonsingular Parameterization” on page 40.

**NOTE:** This section uses the `MODEL` statement in all examples for illustration purposes. All discussion also applies to the `SCALEMODEL` statement in the `SEVSELECT` procedure (where you specify only the regression effects) and to other types of model specification statements in other procedures (such as the `ZEROMODEL` statement in the `CNTSELECT` procedure).

### Effect Operators

Table 1.11 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:

\[
A \mid B \mid C \rightarrow \{A \mid B\} \mid C \\
\rightarrow \{A \mid B\} \mid C \\
\rightarrow A \mid B \mid A*\mid B \mid C \mid A*\mid C \mid B*\mid C \mid A*\mid B*\mid C
\]

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

- Duplicate variables are removed. For example, A(C) \mid B(C) generates A*B(C C), among other terms, and the extra C is removed.

- Effects are discarded if a variable occurs on both the crossed and nested parts of an effect. For example, A(B) \mid B(D E) generates A*B(B D E), but this effect is eliminated immediately.

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

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

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

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

More examples of using the bar and at operators follow:

```
A \mid C(B) \quad \text{is equivalent to} \quad A \mid C(B) \quad A*\mid C(B)
A(B) \mid C(B) \quad \text{is equivalent to} \quad A(B) \mid C(B) \quad A*\mid C(B)
A(B) \mid B(D E) \quad \text{is equivalent to} \quad A(B) \mid B(D E)
A \mid B(A) \mid C \quad \text{is equivalent to} \quad A \mid B(A) \mid C \quad A*\mid C(A)
A \mid B(A) \mid C@2 \quad \text{is equivalent to} \quad A \mid B(A) \mid C \quad A*\mid C
A \mid B \mid C \mid D@2 \quad \text{is equivalent to} \quad A \mid B \mid C \mid D \quad A\mid B \mid C \mid D \quad A\mid B \mid C \mid D
A*\mid B(C*D) \quad \text{is equivalent to} \quad A*\mid B(C D)
```

**Colon, Dash, and Double Dash Operators**

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

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

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

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

model Y = X1-X9;

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

model Y = X1--X9;

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

model Y = A--C;

**GLM Parameterization of Classification Variables and Effects**

Table 1.12 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 X1 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 1.13 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 1.13  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, \( X | X | X \) expands to \( X X X X X X \), which is a cubic model.

**Main Effects**

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

Table 1.14 is an example where \( \beta_0 \) denotes the intercept and A and B are classification variables that have two and three levels, respectively.

Table 1.14  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 1.15).

**Table 1.15** Example of Interaction Effects

<table>
<thead>
<tr>
<th>Data</th>
<th>I</th>
<th>A</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></td>
<td></td>
<td>A</td>
<td>B</td>
<td></td>
<td>A*B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</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 1.16).

**Table 1.16** Example of Nested Effects

<table>
<thead>
<tr>
<th>Data</th>
<th>I</th>
<th>A</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>A</td>
<td>B(A)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>B</td>
<td>( \beta_0 )</td>
<td>A1</td>
<td>A2</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Table 1.16  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 1.17).

Table 1.17  Example of Continuous-Nesting-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>A1</td>
<td>A2</td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>24</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>22</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>28</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>19</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>23</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</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 1.18 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 1.17.

Table 1.18  Example of Continuous-by-Class Effects

<table>
<thead>
<tr>
<th>Data</th>
<th>I</th>
<th>X</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>
<td>A2</td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>1</td>
<td>21</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>24</td>
<td>1</td>
<td>1</td>
<td>24</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>22</td>
<td>1</td>
<td>1</td>
<td>22</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>28</td>
<td>2</td>
<td>1</td>
<td>28</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>19</td>
<td>2</td>
<td>1</td>
<td>19</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>23</td>
<td>2</td>
<td>1</td>
<td>23</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

You can use continuous-by-class effects together with pure continuous effects to test for homogeneity of slopes.
Chapter 1: Shared Concepts

General Effects
An example that combines all the effects is $X_1*X_2*A*B*C(D E)$. 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:

```
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.

<table>
<thead>
<tr>
<th>Design Matrix</th>
<th>A</th>
<th>A1</th>
<th>A2</th>
<th>A5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>–1</td>
<td>–1</td>
<td>–1</td>
<td></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.

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.

<table>
<thead>
<tr>
<th>Design Matrix</th>
<th>A</th>
<th>A1</th>
<th>A2</th>
<th>A5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>–1</td>
<td>–1</td>
<td>–1</td>
<td></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.

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.

<table>
<thead>
<tr>
<th>Design Matrix</th>
<th>A</th>
<th>A2</th>
<th>A5</th>
<th>A7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></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, . . . according to their sort order. The design matrix columns for \(A\) are as follows.

<table>
<thead>
<tr>
<th>Polynomial Coding</th>
<th>Design Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>APOLY1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<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>
<th>Design Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>AOEFF1</td>
</tr>
<tr>
<td>1</td>
<td>1.41421</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>–1.41421</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>
<th>Design Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>AOORD1</td>
</tr>
<tr>
<td>1</td>
<td>–1.73205</td>
</tr>
<tr>
<td>2</td>
<td>0.57735</td>
</tr>
<tr>
<td>5</td>
<td>0.57735</td>
</tr>
<tr>
<td>7</td>
<td>0.57735</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></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.

Orthogonal Reference Coding

<table>
<thead>
<tr>
<th></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></th>
<th>Design Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A(B=1)</td>
</tr>
<tr>
<td>B</td>
<td>A</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</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.
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.

Forward Selection

This section applies to the following procedures: SEVSELECT.

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 that is implemented by the procedures in this book is to address the critical problem of when to stop the selection process by assessing 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.

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, consider the following statement:

```
selection method=forward(choose=AIC);
```

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. 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=SBC stop=AIC) stophorizon=1;

In most cases, provided that the stop horizon is long 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=SBC choose=CRITERION) stophorizon=10;
selection method=forward(select=SBC stop=CRITERION) stophorizon=10;

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 smallest value of the corrected Akaike’s information criterion:

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

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 Schwarz Bayesian information criterion (SBC) and a local minimum of the corrected Akaike’s information criterion (AICC):

selection method=forward(stop=AICC choose=SBC);

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. The optimality of this one step choice depends on the optimality of the selection criterion for the model that contains that effect. 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 SBC:

selection method=forward(select=SBC);

When all effects are variables (that is, effects have one degree of freedom and no hierarchy), using AIC, AICC, 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 SBC value, but the selection terminates when SBC 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.
Chapter 1: Shared Concepts

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

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

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

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

Backward Elimination

This section applies to the following procedures: SEVSELECT.

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.

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 44.

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

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

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

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

Stepwise Selection

This section applies to the following procedures: SEVSELECT.

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

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 44.
The procedures in this book support further modification to the stepwise method. 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 stepwise method where effects enter and leave based on the AIC statistic, 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=AIC stop=SBC) stophorizon=1;
```

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.

**Forward-Swap Selection**

This section applies to the following procedures: SEVSELECT.

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.

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.

Informative Missingness

This section applies to the following procedures: SEVSELECT.

The concept of informative missingness is one way to account for missing values in statistical analyses and, in particular, statistical modeling. Missing values can be a problem because they reduce the amount of available data. When you work with classification variables (factors, which are levelized variables), you can treat a missing value as an actual level of the variable and allow it to participate in the analysis.

However, when continuous variables have missing values, the observation is removed from the analysis. In data that have many missing values, removing observations can reduce the amount of available data greatly, and the sets of observations used in one model versus another model can vary based on which variables are included in the model.

Of course, there are many reasons for missing values, and substituting values for missing values has to be done with caution. For example, the famous Framingham Heart study data set contains 5,209 observations on subjects in a longitudinal study that helped understand the relationship between smoking, cholesterol, and coronary heart disease. One of the variables in the data set is AgeCHDdiag. This variable represents the age at which a patient was diagnosed with coronary heart disease (CHD). If you include this variable in a statistical model, only 1,449 observations are available, because the value cannot be observed unless a patient has experienced CHD. Including this variable acts as a filter that reduces the analysis set to the subjects who have CHD. You cannot impute the value for subjects where the variable has a missing value, because you cannot impute an age at which someone who has not had CHD would have contracted coronary heart disease.

With informative missingness, you are not so much substituting imputed values for the missing values as you are modeling the missingness. Consider a simple linear regression model:

\[ y = \beta_0 + \beta_1 x + \epsilon \]

Suppose that some of the values for the regressor variable \( x \) are missing. The fitted model uses only observations for which \( y \) and \( x \) have been observed.

In order to predict the outcome \( y \) for an observation that has a missing \( x \), either you assume that \( y \) is missing or you substitute a value (such as the average value, \( \bar{x} \)) for the missing \( x \). Because the estimate for the intercept is in the simple linear regression model, the predicted value would be the average response of the nonmissing values, \( \bar{y} \).
With informative missingness, you extend the model by adding extra effects for each effect that contains at least one continuous variable. In the simple linear regression model, you add one column to the model and slightly change the content of the \( x \) variable:

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

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 \( \bar{x} \).

---

**Multithreading**

This section applies to the following procedures: CCOPULA, CNTSELECT, CPANEL, CQLIM, and SEVSELECT.

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: SEVSELECT.

First- or Second-Order Algorithms

The factors that go into choosing a particular optimization technique for a particular problem are complex. Trial and error can be involved.

For many optimization problems, computing the gradient takes more computer time than computing the function value. Computing the Hessian sometimes takes much more computer time and memory than computing the gradient, especially when there are many decision variables. Unfortunately, optimization techniques that do not use some kind of Hessian approximation usually require many more iterations than techniques that do use a Hessian matrix, and as a result the total run time of these techniques is often longer. Techniques that do not use the Hessian also tend to be less reliable. For example, they can terminate more easily at stationary points than at global optima.

Table 1.19 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>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
Choosing an Optimization Algorithm

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.

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 \( \text{ABSGCONV} < 1\times 10^{-5} \), \( \text{FCONV} < 2 \times \varepsilon \), 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 \texttt{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 1.19.

**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.

**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


### Overview: CCOPULA Procedure

A multivariate distribution for a random vector contains a description of both the marginal distributions and their dependence structure. A copula approach to formulating a multivariate distribution provides a way to isolate the description of the dependence structure from the marginal distributions. A copula is a function that combines marginal distributions of variables into a specific multivariate distribution. All of the one-dimensional marginals in the multivariate distribution are the cumulative distribution functions of the factors. Copulas help perform large-scale multivariate simulation from separate models, each of which can be fitted using different, even nonnormal, distributional specifications.
**PROC CCOPULA Features**

The CCOPULA procedure enables you to simulate a specified copula, and it supports the following types of copulas:

- normal copula
- t copula
- Archimedean copulas:
  - Clayton copula
  - Frank copula
  - Gumbel copula

**PROC CCOPULA Compared with Other SAS Procedures**

The CCOPULA procedure provides copula simulation functionality comparable to that of the HPCOPULA and COPULA procedures in SAS/ETS software.

**PROC CCOPULA Compared with the HPCOPULA Procedure**

The functionality of the CCOPULA procedure closely resembles that of the HPCOPULA procedure, which is a high-performance procedure. The CCOPULA procedure is the next generation of the HPCOPULA 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 on multiple machines.

The CCOPULA provides all the functionality of the HPCOPULA procedure.

**PROC CCOPULA Compared with the COPULA Procedure**

The CCOPULA procedure is specifically designed to operate on SAS Viya and performs computations in multiple threads on multiple machines. By contrast, the COPULA procedure runs only in a single thread on a single machine.

This release of the CCOPULA procedure contains some, but not all, of the functionality present in the COPULA procedure. The COPULA procedure provides the following features, which are not present in the CCOPULA procedure:

- estimation of parameters for various copulas
- hierarchical Archimedean copulas
- plots
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 2 in Chapter 1, “Shared Concepts.”

---

### Getting Started: CCOPULA Procedure

This example illustrates the use of PROC CCOPULA. The data are daily returns on several major stocks. The main purpose of this example is to simulate from the joint distribution of stock returns a new sample of a specified size, provided that the parameter estimates of the copula model that is used are available.

In the following statements, the DEFINE statement specifies a normal copula named COP, and the CORR= option specifies that the data set `Estimates` be used as the source for the model parameters. The NDRAWS=1000000 option in the SIMULATE statement generates one million observations from the normal copula. The OUTUNIFORM= option specifies the name of the SAS data set to contain the simulated sample that has uniform marginal distributions. Note that this syntax does not require the DATA= option.
/* Copula simulation of uniforms */
proc ccopula;
    var ret_ibm ret_msft ret_bp ret_ko ret_duk;
    define cop normal (corr = estimates);
    simulate cop / ndraws = 1000000
           outuniform = simulated_uniforms;
run;

The simulated data are contained in the new SAS data set, Simulated_Uniforms.

Syntax: CCOPULA Procedure

The following statements are available in the CCOPULA procedure:

PROC CCOPULA options ;
    VAR variables ;
    DEFINE name copula-type < ( parameter-value-options . . . ) > ;
    SIMULATE < copula-name-list > / options ;

Functional Summary

Table 2.1 summarizes the statements and options that the CCOPULA procedure uses.

<table>
<thead>
<tr>
<th>Description</th>
<th>Statement</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Set Options</td>
<td>DEFINE</td>
<td>CORR=</td>
</tr>
<tr>
<td>Specifies the input data set that contains the correlation matrix for elliptical copulas</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Declaring the Role of Variables</td>
<td>VAR</td>
<td></td>
</tr>
<tr>
<td>Specifies the names of the variables to use in copula fitting or in simulation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Copula Simulation Options</td>
<td>SIMULATE</td>
<td>NDRAWS=</td>
</tr>
<tr>
<td>Specifies the random sample size</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specifies the random number generator seed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output Control Options</td>
<td>SIMULATE</td>
<td>OUTUNIFORM=</td>
</tr>
<tr>
<td>Specifies the output data set to contain the random samples from the simulation with uniform marginal distribution</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prints timing information</td>
<td>CCOPULA</td>
<td>PRINTTIMING</td>
</tr>
</tbody>
</table>

PROC CCOPULA Statement

PROC CCOPULA <options> ;

The following options can be used in the PROC CCOPULA statement.

Printing Options

PRINTTIMING

prints a timing report.

DEFINE Statement

DEFINE name copula-type < ( parameter-value-options . . . ) > ;

The DEFINE statement specifies the relevant information about the copula that is used for the simulation. You can specify the following arguments:

name specifies the name of the copula definition. You can use this name later in the SIMULATE statement.

copula-type specifies the type of copula. You must specify one of the following copula types, which are described in the section “Details: CCOPULA Procedure” on page 60:

NORMAL fits the normal copula.
T fits the t copula.
CLAYTON fits the Clayton copula.
FRANK fits the Frank copula.
GUMBEL fits the Gumbel copula.

parameter-value-options specify the input parameters that are used to simulate the specified copula. These options must be appropriate for the type of copula specified. You can specify the following parameter-value-options:

CORR=SAS-data-set

specifies the data set that contains the correlation matrix to use for elliptical copulas. If the correlation matrix is valid but its elements are not submitted in order, then you must provide the variable names in the first column of the matrix, and these names must match the variable names in the VAR statement. See Output 2.1.1 for an example of a correlation matrix input in this form. If the correlation matrix elements are submitted in order, the first column of variable names is not required. You can use this option for normal and t copulas.

DF=value

specifies the degrees of freedom. You can use this option for t copulas.
**THETA=value**  
specifies the parameter value for the Archimedean copulas.

The DEFINE statement is used with the SIMULATE statement.

---

**SIMULATE Statement**

```
SIMULATE <copula-name-list> / options ;
```

The SIMULATE statement simulates data from a specified copula model. The copula name specification is the name of a defined copula as specified by `name` in the DEFINE statement. You can specify the following options:

- **NDRAWS=integer**  
specifies the number of draws to generate for this simulation. By default, NDRAWS=100.

- **OUTUNIFORM=SAS-data-set**  
specifies the output data set to contain the result of the simulation in uniform margins. The data are not created if you do not specify this option.

- **SEED=integer**  
specifies the seed for generating random numbers for the simulation. If you do not provide the seed, a random number is used as the seed.

- **TOLERANCE=value**  
specifies the tolerance allowed for the simulation.

---

**VAR Statement**

```
VAR variables ;
```

The VAR statement specifies the variable names in the input data set that is specified by the DATA= option in the PROC CCOPULA statement. The subset of variables in the data set is used for the copula models in the FIT statement. If there is no input data set, the VAR statement creates the list of variable names for the SIMULATE statement.

---

**Details: CCOPULA Procedure**

**Sklar’s Theorem**

The copula models are tools for studying the dependence structure of multivariate distributions. The usual joint distribution function contains the information both about the marginal behavior of the individual random variables and about the dependence structure between the variables. The copula is introduced to decouple the marginal properties of the random variables and the dependence structures. An $m$-dimensional copula is a
joint distribution function on $[0,1]^m$, where all marginal distributions are standard uniform. The common notation for a copula is $C(u_1, \ldots, u_m)$.

The Sklar (1959) theorem shows the importance of copulas in modeling multivariate distributions. The first part of the theorem states that a copula can be derived from any joint distribution functions, and the second part asserts the opposite: that any copula can be combined with any set of marginal distributions to result in a multivariate distribution function. The theorem follows:

- Let $F$ be a joint distribution function, and let $F_j$, $j = 1, \ldots, m$, be the marginal distributions. Then there exists a copula $C : [0,1]^m \to [0,1]$ such that

$$F(x_1, \ldots, x_m) = C(F_1(x_1), \ldots, F_m(x_m))$$

for all $x_1, \ldots, x_m$ in $[-\infty, \infty]$. Moreover, if the margins are continuous, then $C$ is unique; otherwise $C$ is uniquely determined on $\text{Ran} F_1 \times \cdots \times \text{Ran} F_m$, where $\text{Ran} F_j = F_j([-\infty, \infty])$ is the range of $F_j$.

- The converse is also true. That is, if $C$ is a copula and $F_1, \ldots, F_m$ are univariate distribution functions, then the multivariate function that is defined in the preceding equation is a joint distribution function with marginal distributions $F_j$, $j = 1, \ldots, m$.

### Dependence Measures

There are three basic types of dependence measures: linear correlation, rank correlation, and tail dependence. Linear correlation is given by

$$\rho \equiv \text{corr}(X,Y) = \frac{\text{cov}(X,Y)}{\sqrt{\text{var}(X)} \sqrt{\text{var}(Y)}}$$

The linear correlation coefficient contains very limited information about the joint properties of the variables. A well-known property is that zero correlation does not imply independence, whereas independence implies zero correlation. In addition, there are distinct bivariate distributions that have the same marginal distribution and the same correlation coefficient. These results suggest that caution must be used in interpreting the linear correlation.

Another statistical measure of dependence is rank correlation, which is nonparametric. For example, Kendall’s tau is the covariance between the sign statistics $X_1 - \tilde{X}_1$ and $X_2 - \tilde{X}_2$, where $(\tilde{X}_1, \tilde{X}_2)$ is an independent copy of $(X_1, X_2)$:

$$\rho_\tau \equiv E[\text{sign}(X_1 - \tilde{X}_1)(X_2 - \tilde{X}_2)]$$

The sign function (sometimes written as sgn) is defined as

$$\text{sign}(x) = \begin{cases} 
-1 & \text{if } x \leq 0 \\
0 & \text{if } x = 0 \\
1 & \text{if } x \geq 0
\end{cases}$$

Spearman’s rho is the correlation between the transformed random variables:

$$\rho_S(X_1, X_2) \equiv \rho(F_1(X_1), F_2(X_2))$$
The variables are transformed by their distribution functions so that the transformed variables are uniformly distributed on \([0, 1]\). The rank correlations depend only on the copula of the random variables and are indifferent to the marginal distributions. Like linear correlation, rank correlation has its limitations. In particular, different copulas result in the same rank correlation.

A third measure, tail dependence, focuses on only part of the joint properties between the variables. Tail dependence measures the dependence when both variables have extreme values. Formally, they can be defined as the conditional probabilities of quantile exceedances. There are two types of tail dependence:

- **Upper tail dependence** is defined as
  \[
  \lambda_u(X_1, X_2) \equiv \lim_{q \to 1^-} P(X_2 > F_2^{-1}(q) | X_1 > F_1^{-1}(q))
  \]
  when the limit exists and \(\lambda_u \in [0, 1]\). Here \(F_j^{-1}\) is the quantile function (that is, the inverse of the CDF).

- **Lower tail dependence** is defined symmetrically.

### Normal Copula

Let \(u_j \sim U(0, 1)\) for \(j = 1, \ldots, m\), where \(U(0, 1)\) represents the uniform distribution on the \([0, 1]\) interval. Let \(\Sigma\) be the correlation matrix, where \(m(m-1)/2\) parameters satisfy the positive semidefiniteness constraint. The normal copula can be written as

\[
C_\Sigma(u_1, u_2, \ldots, u_m) = \Phi_\Sigma(\Phi^{-1}(u_1), \ldots, \Phi^{-1}(u_m))
\]

where \(\Phi\) is the distribution function of a standard normal random variable and \(\Phi_\Sigma\) is the \(m\)-variate standard normal distribution with mean vector \(0\) and covariance matrix \(\Sigma\). That is, the distribution \(\Phi_\Sigma\) is \(N_m(0, \Sigma)\).

### Simulation

For the normal copula, the input of the simulation is the correlation matrix \(\Sigma\). The normal copula can be simulated by the following steps, in which \(U = (U_1, \ldots, U_m)\) denotes one random draw from the copula:

1. Generate a multivariate normal vector \(Z \sim N(0, \Sigma)\), where \(\Sigma\) is an \(m\)-dimensional correlation matrix.

2. Transform the vector \(Z\) into \(U = (\Phi(Z_1), \ldots, \Phi(Z_m))^T\), where \(\Phi\) is the distribution function of univariate standard normal.

The first step can be achieved by Cholesky decomposition of the correlation matrix \(\Sigma = LL^T\), where \(L\) is a lower triangular matrix with positive elements on the diagonal. If \(\tilde{Z} \sim N(0, I)\), then \(L\tilde{Z} \sim N(0, \Sigma)\).
Student’s t copula

Let \( \Theta = \{(\nu, \Sigma) : \nu \in (1, \infty), \Sigma \in \mathbb{R}^{m \times m}\}, \) and let \( t_\nu \) be a univariate \( t \) distribution with \( \nu \) degrees of freedom.

The Student’s \( t \) copula can be written as

\[
C_\Theta(u_1, u_2, \ldots, u_m) = t_{\nu, \Sigma}(t_\nu^{-1}(u_1), t_\nu^{-1}(u_2), \ldots, t_\nu^{-1}(u_m))
\]

where \( t_{\nu, \Sigma} \) is the multivariate Student’s \( t \) distribution that has a correlation matrix \( \Sigma \) with \( \nu \) degrees of freedom.

Simulation

The input parameters for the simulation are \((\nu, \Sigma)\). The \( t \) copula can be simulated by the following steps:

1. Generate a multivariate vector \( X \sim t_m(\nu, 0, \Sigma) \) that follows the centered \( t \) distribution with \( \nu \) degrees of freedom and correlation matrix \( \Sigma \).
2. Transform the vector \( X \) into \( U = (t_\nu(X_1), \ldots, t_\nu(X_m))^T \), where \( t_\nu \) is the distribution function of univariate \( t \) distribution with \( \nu \) degrees of freedom.

To simulate centered multivariate \( t \) random variables, you can use the property that \( X \sim t_m(\nu, 0, \Sigma) \) if \( X = \sqrt{\nu/s} Z \), where \( Z \sim N(0, \Sigma) \) and the univariate random variable \( s \sim f^2_\nu \).

Archimedean Copulas

Overview of Archimedean Copulas

Let function \( \phi : [0, 1] \to [0, \infty) \) be a strict Archimedean copula generator function, and suppose that its inverse \( \phi^{-1} \) is completely monotonic on \([0, \infty)\). A strict generator is a decreasing function \( \phi : [0, 1] \to [0, \infty) \) that satisfies \( \phi(0) = \infty \) and \( \phi(1) = 0 \). A decreasing function \( f(t) : [a, b] \to (-\infty, \infty) \) is completely monotonic if it satisfies

\[
(-1)^k \frac{d^k}{dt^k} f(t) \geq 0, k \in \mathbb{N}, t \in (a, b)
\]

An Archimedean copula is defined as follows:

\[
C(u_1, u_2, \ldots, u_m) = \phi^{-1}\left(\phi(u_1) + \cdots + \phi(u_m)\right)
\]

The Archimedean copulas available in the CCOPULA procedure are the Clayton copula, the Frank copula, and the Gumbel copula.
Clayton Copula

Let the generator function \( \phi(u) = \theta^{-1} \left(u^{-\theta} - 1\right) \). A Clayton copula is defined as

\[
C_{\theta}(u_1, u_2, \ldots, u_m) = \left[ \sum_{i=1}^{m} u_i^{-\theta} - m + 1 \right]^{-1/\theta}
\]

where \( \theta > 0 \).

Frank Copula

Let the generator function be

\[
\phi(u) = -\log \left[ \frac{\exp(-\theta u) - 1}{\exp(-\theta) - 1} \right]
\]

A Frank copula is defined as

\[
C_{\theta}(u_1, u_2, \ldots, u_m) = \frac{1}{\theta} \log \left\{ 1 + \frac{\prod_{i=1}^{m} [\exp(-\theta u_i) - 1]}{[\exp(-\theta) - 1]^{m-1}} \right\}
\]

where \( \theta \in (-\infty, \infty) \setminus \{0\} \) for \( m = 2 \) and \( \theta > 0 \) for \( m \geq 3 \).

Gumbel Copula

Let the generator function \( \phi(u) = (-\log u)^\theta \). A Gumbel copula is defined as

\[
C_{\theta}(u_1, u_2, \ldots, u_m) = \exp \left\{ - \left[ \sum_{i=1}^{m} (-\log u_i)^\theta \right]^{1/\theta} \right\}
\]

where \( \theta > 1 \).

Simulation

Suppose that the generator of the Archimedean copula is \( \phi \). Then the simulation method that uses a Laplace-Stieltjes transformation of the distribution function is given by Marshall and Olkin (1988), where \( \tilde{F}(t) = \int_0^\infty e^{-tx} dF(x) \):

1. Generate a random variable \( V \) that has the distribution function \( F \) such that \( \tilde{F}(t) = \phi^{-1}(t) \).
2. Draw samples from the independent uniform random variables \( X_1, \ldots, X_m \).
3. Return \( U = (\tilde{F}(-\log(X_1)/V), \ldots, \tilde{F}(-\log(X_m)/V))^T \).

The Laplace-Stieltjes transformations are as follows:

- For the Clayton copula, \( \tilde{F} = (1 + t)^{-1/\theta} \), and the distribution function \( F \) is associated with a gamma random variable that has a shape parameter of \( \theta^{-1} \) and a scale parameter of 1.
- For the Gumbel copula, \( \tilde{F} = \exp(-t^{1/\theta}) \), and \( F \) is the distribution function of the stable variable \( St(\theta^{-1}, 1, \gamma, 0) \), where \( \gamma = [\cos(\pi/(2\theta))]^{\theta} \).
For the Frank copula where $\theta > 0$, \[ \tilde{F} = -\log\{1 - \exp(-t)[1 - \exp(-\theta)]\}/\theta, \]
and $F$ is a discrete probability function $P(V = k) = (1 - \exp(-\theta))^k/(k\theta)$. This probability function is related to a logarithmic random variable that has a parameter value of $1 - e^{-\theta}$.

For more information about simulating a random variable from a stable distribution, see Theorem 1.19 in Nolan (2010). For more information about simulating a random variable from a logarithmic series, see Chapter 10.5 in Devroye (1986).

For a Frank copula where $m = 2$ and $\theta < 0$, the simulation can be done through conditional distributions as follows:

1. Draw independent $v_1, v_2$ from a uniform distribution.
2. Let $u_1 = v_1$.
3. Let $u_2 = -\frac{1}{\theta} \log \left( 1 + \frac{v_2(1-e^{-\theta})}{v_2(e^{-\theta v_1} - 1) - e^{-\theta v_1}} \right)$.

OUTUNIFORM= Data Sets

The number of columns and the names of columns in OUTUNIFORM= data sets match the number and names of the variables in the VAR statement.

ODS Table Names

PROC CCOPULA assigns a name to each table that it creates. You can use these names to denote the table when you use the Output Delivery System (ODS) to select tables and create output data sets. These table names are listed in Table 2.2.

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Statement</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>TimingDetails</td>
<td>Detailed summary of time taken for all phases of execution</td>
<td>PROC</td>
<td>PRINTTIMING=(DETAILS)</td>
</tr>
<tr>
<td>TimingSummary</td>
<td>Summary of time taken for main phases of execution</td>
<td>PROC</td>
<td>PRINTTIMING</td>
</tr>
</tbody>
</table>
Chapter 2: The CCOPULA Procedure

Examples: CCOPULA Procedure

Example 2.1: Simulating Default Times

Suppose the correlation structure that is required for a normal copula function is already known. For example, the correlation structure can be estimated from the historical data on default times in some industries, but this estimation is not within the scope of this example. The correlation structure is saved in a SAS data set called Inparm. The following statements and their output in Output 2.1.1 show that the correlation parameter is set at 0.8:

```sas
proc print data = inparm;
run;
```

![Output 2.1.1 Copula Correlation Matrix](image)

The following statements use PROC CCOPULA to simulate the data:

```sas
/* simulate the data from bivariate normal copula */
proc ccopula;
  var Y1-Y2;
  define cop normal (corr=mycas.inparm);
  simulate cop /
    ndraws = 1000000
    seed   = 1234
    outuniform = mycas.normal_unifdata;
run;
```

The VAR statement specifies the list of variables that contains the simulated data. The DEFINE statement assigns the name COP and specifies a normal copula that reads the correlation matrix from the Inparm data set. The SIMULATE statement refers to the COP label that is defined in the VAR statement and specifies several options: the NDRAWS= option specifies a sample size, the SEED= option specifies 1234 as the random number generator seed, and the OUTUNIFORM=NORMAL_UNIFDATA option names the output data set to contain the result of simulation in uniforms.

The following DATA step transforms the variables from zero-one uniformly distributed to nonnegative exponentially distributed with parameter 0.5 and adds three indicator variables to the data set: SURVIVE1 and SURVIVE2 are equal to 1 if company 1 or company 2, respectively, has remained in business for more than three years, and SURVIVE is equal to 1 if both companies survived the same period together.

```sas
/* default time has exponential marginal distribution with parameter 0.5 */
data default;
  set mycas.normal_unifdata;
  array arr(2) Y1-Y2;
  array time(2) timel-time2;
```
Example 2.1: Simulating Default Times

array surv(2) survive1-survive2;
lambda = 0.5;
do i=1 to 2;
    time[i] = -log(1-arr[i])/lambda;
surv[i] = 0;
    if (time[i] >3) then surv[i]=1;
end;
survive = 0;
    if (time1 >3) && (time2 >3) then survive = 1;
run;

The first analysis step is to look at correlations between survival times of the two companies. You can perform this step by using the CORR procedure as follows:

    proc corr data = default pearson kendall;
    var time1 time2;
    run;

Output 2.1.2 shows the output of this code. The output contains some descriptive statistics and two measures of correlation: Pearson and Kendall. Both measures indicate high and statistically significant dependence between the life spans of the two companies.

Output 2.1.2  Default Time Descriptive Statistics and Correlations

The CORR Procedure

2 Variables: time1 time2

                      Simple Statistics
                      Variable      N  Mean    Std Dev   Median      Minimum      Maximum
                      time1     1000000  2.00324  2.00150   1.38948   4.74646E-6     29.97492
                      time2     1000000  2.00365  2.00000   1.39228    6.84299E-7    31.14652

   Pearson Correlation Coefficients, N = 1000000
                  Prob > |r| under H0: Rho=0
                  time1    time2
    time1   1.00000   0.77006   <.0001
    time2   0.77006   1.00000   <.0001

Kendall Tau b Correlation Coefficients, N = 1000000
                  Prob > |tau| under H0: Tau=0
                  time1    time2
    time1   1.00000  0.59061   <.0001
    time2   0.59061   1.00000   <.0001

The second and final step is to empirically estimate the default probabilities of the two companies. This is done by using the FREQ procedure as follows:
proc freq data=default;
   table survive survive1-survive2;
run;

The results are shown in Output 2.1.3.

Output 2.1.3  Probabilities of Default

The FREQ Procedure

<table>
<thead>
<tr>
<th>survive</th>
<th>Frequency</th>
<th>Percent</th>
<th>Cumulative Frequency</th>
<th>Cumulative Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>852061</td>
<td>85.21</td>
<td>852061</td>
<td>85.21</td>
</tr>
<tr>
<td>1</td>
<td>147939</td>
<td>14.79</td>
<td>1000000</td>
<td>100.00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>survive1</th>
<th>Frequency</th>
<th>Percent</th>
<th>Cumulative Frequency</th>
<th>Cumulative Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>775941</td>
<td>77.59</td>
<td>775941</td>
<td>77.59</td>
</tr>
<tr>
<td>1</td>
<td>224059</td>
<td>22.41</td>
<td>1000000</td>
<td>100.00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>survive2</th>
<th>Frequency</th>
<th>Percent</th>
<th>Cumulative Frequency</th>
<th>Cumulative Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>776267</td>
<td>77.63</td>
<td>776267</td>
<td>77.63</td>
</tr>
<tr>
<td>1</td>
<td>223733</td>
<td>22.37</td>
<td>1000000</td>
<td>100.00</td>
</tr>
</tbody>
</table>

Output 2.1.3 shows that the empirical default probabilities are 78% and 78%. Assuming that these companies are independent yields the probability estimate that both companies default during the period of three years as 0.78*0.78=0.61 (61%). Comparing this naive estimate with the much higher actual 85% joint default probability illustrates that neglecting the correlation between the two companies significantly underestimates the probability of default.

References


# Chapter 3
## The CNTSELECT Procedure

<table>
<thead>
<tr>
<th>Contents</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview: CNTSELECT Procedure</td>
<td>70</td>
</tr>
<tr>
<td>PROC CNTSELECT Features</td>
<td>70</td>
</tr>
<tr>
<td>PROC CNTSELECT Compared with Other SAS Procedures</td>
<td>71</td>
</tr>
<tr>
<td>Using CAS Sessions and CAS Engine Librefs</td>
<td>72</td>
</tr>
<tr>
<td>Getting Started: CNTSELECT Procedure</td>
<td>72</td>
</tr>
<tr>
<td>Syntax: CNTSELECT Procedure</td>
<td>75</td>
</tr>
<tr>
<td>Functional Summary</td>
<td>76</td>
</tr>
<tr>
<td>PROC CNTSELECT Statement</td>
<td>77</td>
</tr>
<tr>
<td>BOUNDS Statement</td>
<td>79</td>
</tr>
<tr>
<td>BY Statement</td>
<td>80</td>
</tr>
<tr>
<td>CLASS Statement</td>
<td>80</td>
</tr>
<tr>
<td>DISPMODEL Statement</td>
<td>80</td>
</tr>
<tr>
<td>FREQ Statement</td>
<td>81</td>
</tr>
<tr>
<td>INIT Statement</td>
<td>81</td>
</tr>
<tr>
<td>MODEL Statement</td>
<td>81</td>
</tr>
<tr>
<td>OUTPUT Statement</td>
<td>83</td>
</tr>
<tr>
<td>RESTRICT Statement</td>
<td>85</td>
</tr>
<tr>
<td>TEST Statement</td>
<td>85</td>
</tr>
<tr>
<td>WEIGHT Statement</td>
<td>87</td>
</tr>
<tr>
<td>ZEROMODEL Statement</td>
<td>87</td>
</tr>
<tr>
<td>Details: CNTSELECT Procedure</td>
<td>88</td>
</tr>
<tr>
<td>Missing Values</td>
<td>88</td>
</tr>
<tr>
<td>Poisson Regression</td>
<td>88</td>
</tr>
<tr>
<td>Conway-Maxwell-Poisson Regression .</td>
<td>89</td>
</tr>
<tr>
<td>Negative Binomial Regression .</td>
<td>93</td>
</tr>
<tr>
<td>Zero-Inflated Count Regression Overview</td>
<td>95</td>
</tr>
<tr>
<td>Zero-Inflated Poisson Regression</td>
<td>96</td>
</tr>
<tr>
<td>Zero-Inflated Conway-Maxwell-Poisson Regression</td>
<td>97</td>
</tr>
<tr>
<td>Zero-Inflated Negative Binomial Regression</td>
<td>99</td>
</tr>
<tr>
<td>Parameter Naming Conventions for the RESTRICT, TEST, BOUNDS, and INIT Statements</td>
<td>100</td>
</tr>
<tr>
<td>Computational Resources</td>
<td>104</td>
</tr>
<tr>
<td>Covariance Matrix Types</td>
<td>104</td>
</tr>
<tr>
<td>Displayed Output</td>
<td>104</td>
</tr>
<tr>
<td>OUTPUT OUT= Data Set</td>
<td>106</td>
</tr>
<tr>
<td>ODS Table Names</td>
<td>106</td>
</tr>
</tbody>
</table>
Overview: CNTSELECT Procedure

The CNTSELECT (count regression) procedure analyzes regression models in which the dependent variable takes nonnegative integer or count values. The dependent variable is usually an event count, which refers to the number of times an event occurs. For example, an event count might represent the number of ship accidents per year for a given fleet. In count regression, the conditional mean $E(y_i|x_i)$ of the dependent variable $y_i$ is assumed to be a function of a vector of covariates $x_i$.

The Poisson (log-linear) regression model is the most basic model that explicitly takes into account the nonnegative integer-valued aspect of the outcome. For this model, the probability of an event count is determined by a Poisson distribution, where the conditional mean of the distribution is a function of a vector of covariates. However, the basic Poisson regression model is limited because it forces the conditional mean of the outcome to equal the conditional variance. This assumption is often violated in real-life data. Negative binomial regression is an extension of Poisson regression in which the conditional variance can exceed the conditional mean. Also, a common characteristic of count data is that the number of zeros in the sample exceeds the number of zeros that are predicted by either the Poisson or negative binomial model. Zero-inflated Poisson (ZIP) and zero-inflated negative binomial (ZINB) models explicitly model the production of zero counts to account for excess zeros and also enable the conditional variance of the outcome to differ from the conditional mean.

In zero-inflated models, additional zeros occur with probability $\varphi_i$, which is determined by a separate model, $\varphi_i = F(z'_i \gamma)$, where $F$ is the normal or logistic distribution function that results in a probit or logistic model and $z_i$ is a set of covariates.

PROC CNTSELECT Features

The CNTSELECT procedure estimates the parameters of a count regression model by maximum likelihood techniques.

The CNTSELECT procedure supports the following models for count data:

- Poisson regression
- Conway-Maxwell-Poisson regression
- negative binomial regression with quadratic and linear variance functions (Cameron and Trivedi 1986)
- zero-inflated Poisson (ZIP) model (Lambert 1992)
- zero-inflated Conway-Maxwell-Poisson (ZICMP) model
- zero-inflated negative binomial (ZINB) model
PROC CNTSELECT Compared with Other SAS Procedures

- fixed-effects and random-effects Poisson models for panel data
- fixed-effects and random-effects negative binomial models for panel data

The following list summarizes some basic features of the CNTSELECT procedure:

- has model-building syntax that uses CLASS and effect-based MODEL statements familiar from SAS/Econometrics analytic procedures
- performs maximum likelihood estimation
- supports multiple link functions
- uses the WEIGHT statement for weighted analysis
- uses the FREQ statement for grouped analysis
- uses the OUTPUT statement to produce a data set that contains predicted probabilities and other observationwise statistics

PROC CNTSELECT Compared with Other SAS Procedures

The CNTSELECT procedure provides count-data modeling functionality comparable to that of the HPCOUNTREG and COUNTREG procedures in SAS/ETS software.

PROC CNTSELECT Compared with the HPCOUNTREG Procedure

The functionality of the CNTSELECT procedure closely resembles that of the HPCOUNTREG procedure, which is a high-performance procedure. The CNTSELECT procedure is the next generation of the HPCOUNTREG 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 on multiple machines.

The CNTSELECT provides all the functionality of the HPCOUNTREG procedure.

PROC CNTSELECT Compared with the COUNTREG Procedure

The CNTSELECT procedure is specifically designed to operate on SAS Viya and performs computations in multiple threads on multiple machines. Although the COUNTREG procedure runs in multiple threads, it executes only on a single machine.

This release of the CNTSELECT procedure contains some, but not all, of the functionality present in the COUNTREG procedure. The COUNTREG procedure provides the following features, which are not present in the CNTSELECT procedure:

- variable selection
- spatial effect models
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 2 in Chapter 1, “Shared Concepts.”

Getting Started: CNTSELECT Procedure

Except for its ability to operate in the high-performance distributed environment, the CNTSELECT procedure is similar in use to other regression model procedures in the SAS System. For example, the following statements are used to estimate a Poisson regression model:

```sas
proc cntselect data=one;
   model y = x / dist=poisson;
run;
```
The response variable \( y \) is numeric and has nonnegative integer values.

This section illustrates two simple examples that use PROC CNTSELECT. The data are taken from Long (1997). This study examines how factors such as gender (fem), marital status (mar), number of young children (kid5), prestige of the graduate program (phd), and number of articles published by a scientist’s mentor (ment) affect the number of articles (art) published by the scientist.

The first 10 observations are shown in Figure 3.1.

**Figure 3.1** Article Count Data

<table>
<thead>
<tr>
<th>Obs</th>
<th>art</th>
<th>fem</th>
<th>mar</th>
<th>kid5</th>
<th>phd</th>
<th>ment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>1.38000</td>
<td>8.0000</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4.29000</td>
<td>7.0000</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3.85000</td>
<td>47.0000</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>3.59000</td>
<td>19.0000</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1.81000</td>
<td>0.0000</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>3.59000</td>
<td>6.0000</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2.12000</td>
<td>10.0000</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>4.29000</td>
<td>2.0000</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2.58000</td>
<td>2.0000</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1.80000</td>
<td>4.0000</td>
</tr>
</tbody>
</table>

The following SAS statements estimate the Poisson regression model.

```sas
/*--- Poisson Regression ---*/
proc cntselect data=mycas.long97data;
    model art = fem mar kid5 phd ment / dist=poisson method=quanew;
run;
```

The “Model Fit Summary” table that is shown in Figure 3.2 lists several details about the model. By default, the CNTSELECT procedure uses the Newton-Raphson optimization technique. The maximum log-likelihood value is shown, in addition to two information measures—Akaike’s information criterion (AIC) and Schwarz’s Bayesian information criterion (SBC)—which can be used to compare competing Poisson models. Smaller values of these criteria indicate better models.

**Figure 3.2** Estimation Summary Table for a Poisson Regression

### The CNTSELECT Procedure

<table>
<thead>
<tr>
<th>Model Fit Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dependent Variable</td>
</tr>
<tr>
<td>Number of Observations</td>
</tr>
<tr>
<td>Data Set</td>
</tr>
<tr>
<td>Model</td>
</tr>
<tr>
<td>Log Likelihood</td>
</tr>
<tr>
<td>Maximum Absolute Gradient</td>
</tr>
<tr>
<td>Number of Iterations</td>
</tr>
<tr>
<td>Optimization Method</td>
</tr>
<tr>
<td>AIC</td>
</tr>
<tr>
<td>SBC</td>
</tr>
<tr>
<td>Covariance Estimation</td>
</tr>
</tbody>
</table>
Chapter 3: The CNTSELECT Procedure

Figure 3.3 shows the parameter estimates of the model and their standard errors. All covariates are significant predictors of the number of articles, except for the prestige of the program (phd), which has a \( p \)-value of 0.6271.

**Figure 3.3** Parameter Estimates of Poisson Regression

![Parameter Estimates Table]

| Parameter | DF | Estimate | Standard Error | t Value | Pr > |t|   |
|-----------|----|----------|----------------|---------|------|----|
| Intercept | 1  | 0.304617 | 0.102982       | 2.96    | 0.0031 |
| fem       | 1  | -0.224595| 0.054614       | -4.11   | <.0001 |
| mar       | 1  | 0.155243 | 0.061375       | 2.53    | 0.0114 |
| kid5      | 1  | -0.184882| 0.040127       | -4.61   | <.0001 |
| phd       | 1  | 0.012823 | 0.026397       | 0.49    | 0.6271 |
| ment      | 1  | 0.025543 | 0.002006       | 12.73   | <.0001 |

To allow for variance greater than the mean, you can fit the negative binomial model instead of the Poisson model by specifying the DIST=NEGBIN option, as shown in the following statements. Whereas the Poisson model requires that the conditional mean and conditional variance be equal, the negative binomial model allows for overdispersion, in which the conditional variance can exceed the conditional mean.

```latex
/*-- Negative Binomial Regression --*/
proc cntselect data=mycas.long97data;
   model art = fem mar kid5 phd ment / dist=negbin(p=2) method=quanew;
run;
```

Figure 3.4 shows the fit summary and Figure 3.5 shows the parameter estimates.

**Figure 3.4** Estimation Summary Table for a Negative Binomial Regression

<table>
<thead>
<tr>
<th>The CNTSELECT Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Model Fit Summary</strong></td>
</tr>
<tr>
<td>Dependent Variable</td>
</tr>
<tr>
<td>Number of Observations</td>
</tr>
<tr>
<td>Data Set</td>
</tr>
<tr>
<td>Model</td>
</tr>
<tr>
<td>Log Likelihood</td>
</tr>
<tr>
<td>Maximum Absolute Gradient</td>
</tr>
<tr>
<td>Number of Iterations</td>
</tr>
<tr>
<td>Optimization Method</td>
</tr>
<tr>
<td>AIC</td>
</tr>
<tr>
<td>SBC</td>
</tr>
<tr>
<td>Covariance Estimation</td>
</tr>
</tbody>
</table>
### Syntax: CNTSELECT Procedure

The following statements are available in the CNTSELECT procedure. Items within angle brackets (< >) or square brackets ([ ] ) are optional.

```
PROC CNTSELECT <options> ;
   BOUNDS bound1 [, bound2 . . . ] ;
   BY variables ;
   CLASS variables ;
   DISPMODEL dependent variable ~ < dispersion-related regressors > ;
   FREQ freq-variable ;
   INIT initialization1 < , initialization2 . . . > ;
   MODEL dependent-variable = regressors < / options > ;
   OUTPUT < output-options > ;
   RESTRICT restriction1 [, restriction2 . . . ] ;
   TEST equation1 < , equation2 . . . > / < test-options > ;
   WEIGHT variable </ option > ;
   ZEROMODEL dependent-variable ~ zero-inflated-regressors </ options > ;
```

There can be only one MODEL statement. Any ZEROMODEL, DISPMODEL, OUTPUT, INIT, BOUNDS, RESTRICT, or TEST statement, if used, must appear after the MODEL statement. There can be no more than one ZEROMODEL statement and no more than one DISPMODEL statement. Multiple INIT, BOUNDS, RESTRICT, and TEST statements are allowed. If a FREQ or WEIGHT statement is specified more than once, the variable specified in the first instance is used.

---

**Figure 3.5** Parameter Estimates of Negative Binomial Regression

| Parameter | DF | Estimate  | Standard Error | t Value | Approx Pr > |t| |
|-----------|----|-----------|----------------|---------|-------------|---|
| Intercept | 1  | 0.256154  | 0.138560       | 1.85    | 0.0645      |
| fem       | 1  | -0.216404 | 0.072672       | -2.98   | 0.0029      |
| mar       | 1  | 0.150485  | 0.082106       | 1.83    | 0.0668      |
| kid5      | 1  | -0.176423 | 0.053060       | -3.32   | 0.0009      |
| phd       | 1  | 0.015266  | 0.036040       | 0.42    | 0.6719      |
| ment      | 1  | 0.029083  | 0.003470       | 8.38    | <.0001      |
| _Alpha    | 1  | 0.441618  | 0.052967       | 8.34    | <.0001      |

The parameter estimate for _Alpha of 0.4416 is an estimate of the dispersion parameter in the negative binomial distribution. A t test for the hypothesis $H_0 : \alpha = 0$ is provided. It is highly significant, indicating overdispersion ($p < 0.0001$).

The null hypothesis $H_0 : \alpha = 0$ can be also tested against the alternative $\alpha > 0$ by using the likelihood ratio test, as described by Cameron and Trivedi (1998, pp. 45, 77–78). The likelihood ratio test statistic is equal to $-2(\mathcal{L}_P - \mathcal{L}_{NB}) = -2(-1651 + 1561) = 180$, which is highly significant, providing strong evidence of overdispersion.
# Functional Summary

Table 3.1 summarizes the statements and options used with the CNTSELECT procedure.

<table>
<thead>
<tr>
<th>Description</th>
<th>Statement</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data Set Options</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specifies the input data set</td>
<td>CNTSELECT</td>
<td>DATA=</td>
</tr>
<tr>
<td>Specifies the identification variable for panel data analysis</td>
<td>CNTSELECT</td>
<td>GROUPID=</td>
</tr>
<tr>
<td>Writes estimates to an output data set</td>
<td>OUTPUT</td>
<td>OUT=</td>
</tr>
<tr>
<td>Specifies BY-group processing</td>
<td>BY</td>
<td></td>
</tr>
<tr>
<td>Specifies an optional frequency variable</td>
<td>FREQ</td>
<td></td>
</tr>
<tr>
<td>Specifies an optional weight variable</td>
<td>WEIGHT</td>
<td></td>
</tr>
<tr>
<td><strong>Printing Control Options</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prints the correlation matrix of the estimates</td>
<td>CNTSELECT</td>
<td>CORRB</td>
</tr>
<tr>
<td>Prints the covariance matrix of the estimates</td>
<td>CNTSELECT</td>
<td>COVB</td>
</tr>
<tr>
<td>Suppresses the normal printed output</td>
<td>CNTSELECT</td>
<td>NOPRINT</td>
</tr>
<tr>
<td>Requests all printing options</td>
<td>CNTSELECT</td>
<td>PRINTALL</td>
</tr>
<tr>
<td>Prints timing information</td>
<td>CNTSELECT</td>
<td>PRINTTIMING</td>
</tr>
<tr>
<td>Prints the names used internally for the parameters</td>
<td>CNTSELECT</td>
<td>PRINTINTERNALNAMES</td>
</tr>
<tr>
<td><strong>Options to Control the Optimization Process</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Selects the iterative minimization method to use</td>
<td>CNTSELECT</td>
<td>METHOD=</td>
</tr>
<tr>
<td>Specifies maximum number of iterations allowed</td>
<td>CNTSELECT</td>
<td>MAXITER=</td>
</tr>
<tr>
<td>Specifies maximum number of function calls</td>
<td>CNTSELECT</td>
<td>MAXFUNC=</td>
</tr>
<tr>
<td>Specifies the upper limit of CPU time in seconds</td>
<td>CNTSELECT</td>
<td>MAXTIME=</td>
</tr>
<tr>
<td>Sets boundary restrictions on parameters</td>
<td>BOUNDS</td>
<td></td>
</tr>
<tr>
<td>Sets initial values for parameters</td>
<td>INIT</td>
<td></td>
</tr>
<tr>
<td>Sets linear restrictions on parameters</td>
<td>RESTRICT</td>
<td></td>
</tr>
<tr>
<td><strong>Model Estimation Options</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specifies the dispersion variables</td>
<td>DISPMODEL</td>
<td></td>
</tr>
<tr>
<td>Specifies the type of model</td>
<td>CNTSELECT</td>
<td>DIST=</td>
</tr>
<tr>
<td>Specifies the type of covariance matrix</td>
<td>CNTSELECT</td>
<td>COVEST=</td>
</tr>
<tr>
<td>Specifies the type of error components model for panel data</td>
<td>MODEL</td>
<td>ERRORCOMP=</td>
</tr>
<tr>
<td>Suppresses the intercept parameter</td>
<td>MODEL</td>
<td>NOINT</td>
</tr>
<tr>
<td>Specifies the offset variable</td>
<td>MODEL</td>
<td>OFFSET=</td>
</tr>
<tr>
<td>Specifies the parameterization for the Conway-Maxwell-Poisson (CMP) model</td>
<td>MODEL</td>
<td>PARAMETER=</td>
</tr>
<tr>
<td>Specifies the zero-inflated offset variable</td>
<td>ZEROMODEL</td>
<td>OFFSET=</td>
</tr>
<tr>
<td>Specifies the zero-inflated link function</td>
<td>ZEROMODEL</td>
<td>LINK=</td>
</tr>
<tr>
<td>Description</td>
<td>Statement</td>
<td>Option</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------</td>
<td>--------</td>
</tr>
<tr>
<td>Outputs SAS variables to the output data set</td>
<td>OUTPUT</td>
<td>COPYVAR=</td>
</tr>
<tr>
<td>Outputs the estimates of dispersion for the CMP model</td>
<td>OUTPUT</td>
<td>DISPERSION</td>
</tr>
<tr>
<td>Outputs the estimates of GDelta = g_iδ for the CMP model</td>
<td>OUTPUT</td>
<td>GDELTA=</td>
</tr>
<tr>
<td>Outputs the estimates of λ for the CMP model</td>
<td>OUTPUT</td>
<td>LAMBDA=</td>
</tr>
<tr>
<td>Outputs the estimates of ν for the CMP model</td>
<td>OUTPUT</td>
<td>NU=</td>
</tr>
<tr>
<td>Outputs the estimates of μ for the CMP model</td>
<td>OUTPUT</td>
<td>MU=</td>
</tr>
<tr>
<td>Outputs the estimates of mode for the CMP model</td>
<td>OUTPUT</td>
<td>MODE=</td>
</tr>
<tr>
<td>Outputs the probability that the response variable will take the current value</td>
<td>OUTPUT</td>
<td>PROB=</td>
</tr>
<tr>
<td>Outputs probabilities for particular response values</td>
<td>OUTPUT</td>
<td>PROBCOUNT( )</td>
</tr>
<tr>
<td>Outputs expected value of response variable</td>
<td>OUTPUT</td>
<td>PRED=</td>
</tr>
<tr>
<td>Outputs the estimates of variance for the CMP model</td>
<td>OUTPUT</td>
<td>VARIANCE=</td>
</tr>
<tr>
<td>Outputs estimates of XBeta = x_iβ</td>
<td>OUTPUT</td>
<td>XBETA=</td>
</tr>
<tr>
<td>Outputs estimates of ZGamma = z_iγ</td>
<td>OUTPUT</td>
<td>ZGAMMA=</td>
</tr>
<tr>
<td>Outputs probability of a zero value as a result of the zero-generating process</td>
<td>OUTPUT</td>
<td>PROBZERO=</td>
</tr>
</tbody>
</table>

**PROC CNTSELECT Statement**

```plaintext
PROC CNTSELECT <options> ;
```

The following `options` can be used in the PROC CNTSELECT statement.

**Input Data Set Options**

- **DATA=SAS-data-set**
  - specifies the input SAS data set. If the `DATA=` option is not specified, PROC CNTSELECT uses the most recently created SAS data set.

- **GROUPID=variable**
  - specifies an identification variable when a panel data model is estimated. The identification variable is used as a cross-sectional ID variable.

**Printing Options**

You can specify the following options in either the PROC CNTSELECT statement or the MODEL statement:
Chapter 3: The CNTSELECT Procedure

**CORRB**
prints the correlation matrix of the parameter estimates.

**COVB**
prints the covariance matrix of the parameter estimates.

**NOPRINT**
suppresses all printed output.

**PRINTALL**
requests all printing options.

**PRINTTIMING**
prints a timing report.

**PRINTINTERNALNAMES**
prints internal names assigned to parameters.

**Estimation Control Options**
You can specify the following options in either the PROC CNTSELECT statement or the MODEL statement:

**COVEST=HESSIAN | OP | QML**
specifies the type of covariance matrix for the parameter estimates.

The default is COVEST=HESSIAN. You can specify the following values:

**HESSIAN** specifies the covariance from the Hessian matrix.

**OP** specifies the covariance from the outer product matrix.

**QML** specifies the covariance from the outer product and Hessian matrices.

**Optimization Control Options**
PROC CNTSELECT uses the nonlinear optimization (NLO) subsystem to perform nonlinear optimization tasks. You can specify the following options in either the PROC CNTSELECT statement or the MODEL statement.

**MAXFUNC=i**

**MAXFU=i**
specifies the maximum number of function calls in the optimization process. The default is 1,000.

The optimization can terminate only after completing a full iteration. Therefore, the number of function calls that are actually performed can exceed the number of calls that are specified by this option.

**MAXITER=i**

**MAXIT=i**
specifies the maximum number of iterations in the optimization process. The default is 200.
**BOUNDS Statement**

The BOUNDS statement imposes simple boundary constraints on the parameter estimates. You can specify any number of BOUNDS statements.

Each bound is composed of parameter names, constants, and inequality operators as follows:

```
item operator item [ operator item [ operator item ... ] ]
```

Each item is a constant, a parameter name, or a list of parameter names. Each operator is <, >, <=, or >=. Parameter names are as shown in the Parameter column of the “Parameter Estimates” table. If a parameter name contains a blank or some other special character (such as '*', '-', ',', or ')'), then you must use the internal name of the parameter in order to refer to that parameter in the BOUNDS statement. You can specify the PRINTINTERNALNAMES option in the PROC CNTSELECT statement if you want to see the internal names of the parameters. When you specify the PRINTINTERNALNAMES option, an extra column is added to the “Parameter Estimates” table, which shows the internal name of each parameter.

For more information about how parameters are named in the BOUNDS statement, see the section “Parameter Naming Conventions for the RESTRICT, TEST, BOUNDS, and INIT Statements” on page 100.

You can use both the BOUNDS statement and the RESTRICT statement to impose boundary constraints. However, the BOUNDS statement provides a simpler syntax for specifying these kinds of constraints. For more information, see the section “RESTRICT Statement” on page 85.

**MAXTIME=r**

specifies an upper limit of $r$ seconds of CPU time for the optimization process. The default value is the largest floating-point double representation of your computer. The time that is specified by this option is checked only once at the end of each iteration. Therefore, the actual run time can be much longer than $r$. The actual run time includes the remaining time needed to finish the iteration and the time needed to generate the output of the results.

**METHOD=value**

specifies the iterative minimization method to use. The default is METHOD=NEWRAP. You can specify the following values:

- **CONGRA** specifies the conjugate-gradient method.
- **DBLDOG** specifies the double-dogleg method.
- **NEWRAP** specifies the Newton-Raphson method (this is the default).
- **NONE** specifies that no optimization be performed beyond using the ordinary least squares method to compute the parameter estimates.
- **NRRIDG** specifies the Newton-Raphson Ridge method.
- **QUANEW** specifies the quasi-Newton method.
- **TRUREG** specifies the trust region method.
The following BOUNDS statement illustrates the use of parameter lists to specify boundary constraints. It constrains the estimates of the parameter for \( z \) to be negative, the parameters for \( x_1 \) through \( x_{10} \) to be between 0 and 1, and the parameter for \( x_1 \) in the zero-inflation model to be less than 1.

\[
\begin{align*}
\text{bounds} & \quad z < 0, \\
& \quad 0 < x_1-x_{10} < 1, \\
& \quad \text{Inf}_x x_1 < 1;
\end{align*}
\]

**BY Statement**

\[
\text{BY variables ;}
\]

A BY statement can be used with PROC CNTSELECT to obtain separate analyses on observations in groups defined by the BY variables. When a BY statement appears, the input data set should be sorted in order of the BY variables.

BY statement processing is not supported when the CNTSELECT procedure runs alongside the database or alongside the Hadoop Distributed File System (HDFS). These modes are used if the input data are stored in a database or HDFS and the grid host is the appliance that houses the data.

**CLASS Statement**

\[
\text{CLASS variables ;}
\]

The CLASS statement names the classification variables to be used in the analysis. Classification variables can be either character or numeric.

Class levels are determined from the formatted values of the CLASS variables. Thus, you can use formats to group values into levels. For more information, see the discussion of the FORMAT procedure in SAS Language Reference: Dictionary.

**DISPMODEL Statement**

\[
\text{DISPMODEL dependent-variable} \sim <\text{dispersion-related-regressors}> ;
\]

The DISPMODEL statement specifies the dispersion-related-regressors that are used to model dispersion. This statement is ignored unless you specify either DIST=CMPOISSON or DIST=ZICMPOISSON in the MODEL statement. The dependent-variable in the DISPMODEL statement must be the same as the dependent-variable in the MODEL statement.

The dependent-variable that appears in the DISPMODEL statement is directly used to model dispersion. Each of the \( q \) variables to the right of the tilde (\( \sim \)) has a parameter to be estimated in the regression. For example, let \( g'_i \) be the \( i \)th observation’s \( 1 \times (q + 1) \) vector of values of the \( q \) dispersion explanatory variables (\( q_0 \) is set to 1 for the intercept term). Then the dispersion is a function of \( g'_i \delta \), where \( \delta \) is the \( (q + 1) \times 1 \) vector of parameters to be estimated, the dispersion model intercept is \( \delta_0 \), and the coefficients for the \( q \) dispersion covariates are \( \delta_1, \ldots, \delta_q \). If you specify DISP=CMPOISSON in the MODEL statement but do not include a DISPMODEL statement, then only the intercept term \( \delta_0 \) is estimated. The “Parameter Estimates” table in the displayed output shows the estimates for the dispersion intercept and dispersion explanatory variables;
they are labeled with the prefix “Disp_”. For example, the dispersion intercept is labeled “Disp_Intercept”. If you specify Age (a variable in your data set) as a dispersion explanatory variable, then the “Parameter Estimates” table labels the corresponding parameter estimate “Disp_Age”. The following statements fit a Conway-Maxwell-Poisson model by using the regressors SEX, ILLNESS, and INCOME and by using AGE as a dispersion-related regressor:

```
proc cntselect data=docvisit;
   model doctorvisits=sex illness income / dist=cmpoisson;
   dispmodel doctorvisits ~ age;
run;
```

**FREQ Statement**

```
FREQ freq-variable;
```

The FREQ statement identifies a variable (freq-variable) that contains the frequency of occurrence of each observation. PROC CNTSELECT treats each observation as if it appears \( n \) times, where \( n \) is the value of freq-variable for the observation. If the value for the observation is not an integer, it is truncated to an integer. If the value is less than 1 or missing, the observation is not used in the model fitting. When the FREQ statement is not specified, each observation is assigned a frequency of 1.

**INIT Statement**

```
INIT initialization1 < , initialization2 . . . > ;
```

The INIT statement sets initial values for parameters in the optimization.

Each initialization is written as a parameter or parameter list, followed by an optional equal sign (=), followed by a number:

```
parameter = number
```

Parameter names are as shown in the Parameter column of the “Parameter Estimates” table. If a parameter name contains a blank or some other special character (such as ‘*’, ‘-’, ‘(‘, or ‘)’), then you must use the internal name of the parameter in order to refer to that parameter in the INIT statement. You can specify the PRINTINTERNALNAMES option in the PROC CNTSELECT statement if you want to see the internal names of the parameters. When you specify the PRINTINTERNALNAMES option, an extra column is added to the “Parameter Estimates” table, which shows the internal name of each parameter.

For more information about how parameters are named in the INIT statement, see the section “Parameter Naming Conventions for the RESTRICT, TEST, BOUNDS, and INIT Statements” on page 100.

**MODEL Statement**

```
MODEL dependent-variable = regressors </options> ;
```

The MODEL statement specifies the dependent variable and independent regressor variables for the regression model. The dependent count variable should take only nonnegative integer values from the input data set.
PROC CNTSELECT rounds any positive noninteger count value to the nearest integer. PROC CNTSELECT discards any observation that has a negative count.

Only one MODEL statement can be specified. You can specify the following options in the MODEL statement after a slash (/):

\textbf{DIST=value}\hspace{1cm}
specifies a type of model to be analyzed. You can specify the following values:

\begin{itemize}
  \item \textbf{POISSON} \mid \textbf{P} \hspace{1cm} \text{specifies the Poisson regression model.}
  \item \textbf{CMPOISSON} \mid \textbf{C} \mid \textbf{CMP} \hspace{1cm} \text{specifies a Conway-Maxwell-Poisson regression model.}
  \item \textbf{NEGBIN}(P=1) \hspace{1cm} \text{specifies the negative binomial regression model that uses a linear variance function.}
  \item \textbf{NEGBIN}(P=2) \mid \textbf{NEGBIN} \hspace{1cm} \text{specifies the negative binomial regression model that uses a quadratic variance function.}
  \item \textbf{ZIPOISSON} \mid \textbf{ZIP} \hspace{1cm} \text{specifies zero-inflated Poisson regression.}
  \item \textbf{ZICMPOISSON} \mid \textbf{ZICMP} \hspace{1cm} \text{specifies a zero-inflated Conway-Maxwell-Poisson regression. The ZERO-MODEL statement must be specified when this model type is specified.}
  \item \textbf{ZINEGBIN} \mid \textbf{ZINB} \hspace{1cm} \text{specifies zero-inflated negative binomial regression.}
\end{itemize}

You can also specify the DIST option in the CNTSELECT statement.

\textbf{ERRORCOMP=FIXED} \mid \textbf{RANDOM}\hspace{1cm}
specifies a type of conditional panel model to be analyzed. You can specify the following model types:

\begin{itemize}
  \item \textbf{FIXED} \hspace{1cm} \text{specifies a fixed-effect error component regression model.}
  \item \textbf{RANDOM} \hspace{1cm} \text{specifies a random-effect error component regression model.}
\end{itemize}

\textbf{NOINT}\hspace{1cm}
suppresses the intercept parameter.

\textbf{OFFSET=offset-variable}\hspace{1cm}
specifies a variable in the input data set to be used as an offset variable. The \textit{offset-variable} is used to allow the observational units to vary across observations. For example, when the number of shipping accidents could be measured across different time periods or the number of students who participate in an activity could be reported across different class sizes, the observational units need to be adjusted to a common denominator by using the offset variable. The offset variable appears as a covariate in the model with its parameter restricted to 1. The offset variable cannot be the response variable, the zero-inflation offset variable (if any), or any of the explanatory variables. The “Model Fit Summary” table gives the name of the data set variable that is used as the offset variable; it is labeled “Offset.”

\textbf{PARAMETER=MU} \mid \textbf{LAMBDA}\hspace{1cm}
specifies the parameterization for the Conway-Maxwell-Poisson model. The following parameterizations are supported:

\begin{itemize}
  \item \textbf{LAMBDA} \hspace{1cm} \text{estimates the original Conway-Maxwell-Poisson model (Shmueli et al. 2005).}
\end{itemize}
reparameterizes $\lambda$ as documented by Guikema and Coffelt (2008), where $\mu = \lambda^{1/\nu}$ and the integral part of $\mu$ represents the mode, which can be considered a measure of central tendency (mean).

By default, PARAMETER=MU.

**Printing Options**

You can specify the following options in either the PROC CNTSELECT statement or the MODEL statement:

- **CORRB**
  prints the correlation matrix of the parameter estimates.

- **COVB**
  prints the covariance matrix of the parameter estimates.

- **NOPRINT**
  suppresses all printed output.

- **PRINTALL**
  requests all printing options.

**OUTPUT Statement**

The OUTPUT statement creates a new SAS data set that includes variables created by the output-options. These variables include the estimates of $x_i \beta$, the expected value of the response variable, and the probability of the response variable taking on the current value. Furthermore, if a zero-inflated model was fit, you can request that the output data set contain the estimates of $z_i \gamma$ and the probability that the response is zero as a result of the zero-generating process. For the Conway-Maxwell-Poisson model, the estimates of $g_i \delta$, $\lambda$, $\nu$, $\mu$, mode, variance, and dispersion are also available. Except for the probability of the current value, these statistics can be computed for all observations in which the regressors are not missing, even if the response is missing. By adding observations that have missing response values to the input data set, you can compute these statistics for new observations or for settings of the regressors that are not present in the data without affecting the model fit.

You can specify only one OUTPUT statement. You can specify the following output-options:

- **OUT=SAS-data-set**
  names the output data set

- **COPYVAR=SAS-variable-names**
  adds SAS variables to the output data set.

- **COPYVARS=SAS-variable-names**
  adds SAS variables to the output data set.

- **XBETA=name**
  names the variable to contain estimates of $x_i \beta$. 
PRED=name
MEAN=name
  names the variable to contain the predicted value of the response variable.

PROB=name
  names the variable to contain the probability that the response variable will take the actual value, \( \Pr(Y = y_i) \).

PROBCOUNT(value1 < value2 . . . >)
  outputs the probability that the response variable will take particular values. Each value should be a nonnegative integer. If you specify a noninteger, it is rounded to the nearest integer. The value can also be a list of the form X TO Y BY Z. For example, PROBCOUNT(0 1 2 TO 10 BY 2 15) requests predicted probabilities for counts 0, 1, 2, 5, 6, 8, 10, and 15. This option is not available for the fixed-effects and random-effects panel models.

ZGAMMA=name
  names the variable to contain estimates of \( z_i \).

PROBZERO=name
  names the variable to contain the value of \( \varphi_i \), which is the probability that the response variable will take the value of 0 as a result of the zero-generating process. This variable is written to the output file only if the model is zero-inflated.

GDELTA=name
  assigns a name to the variable that contains estimates of \( g_i \delta \) for the Conway-Maxwell-Poisson distribution.

LAMBDAAA=name
  assigns a name to the variable that contains the estimate of \( \lambda \) for the Conway-Maxwell-Poisson distribution.

NU=name
  assigns a name to the variable that contains the estimate of \( \nu \) for the Conway-Maxwell-Poisson distribution.

MU=name
  assigns a name to the variable that contains the estimate of \( \mu \) for the Conway-Maxwell-Poisson distribution.

MODE=name
  assigns a name to the variable that contains the integral part of \( \mu \) (mode) for the Conway-Maxwell-Poisson distribution.

VARIANCE=name
  assigns a name to the variable that contains the estimate of variance for the Conway-Maxwell-Poisson distribution.

DISPERSION=name
  assigns a name to the variable that contains the value of dispersion for the Conway-Maxwell-Poisson distribution.
RESTRICT Statement

RESTRICT restriction1 [, restriction2 ] ;

The RESTRICT statement imposes linear restrictions on the parameter estimates. You can specify any number of RESTRICT statements.

Each restriction is written as an expression, followed by an equality operator (=) or an inequality operator (<, >, <=, >=) and then by a second expression, as follows:

expression operator expression

The operator can be =, <, >, <=, or >=.

Restriction expressions can be composed of parameter names, constants, and the following operators: times (*), plus (+), and minus (−). The restriction expressions must be a linear function of the variables.

Parameter names are as shown in the Parameter column of the “Parameter Estimates” table. If a parameter name contains a blank or some other special character (such as *, '-', '(', or ')'), then you must use the internal name of the parameter in order to refer to that parameter in the RESTRICT statement. You can specify the PRINTINTERNALNAMES option in the PROC CNTSELECT statement if you want to see the internal names of the parameters. When you specify the PRINTINTERNALNAMES option, an extra column is added to the “Parameter Estimates” table, which shows the internal name of each parameter.

For more information about how parameters are named in the RESTRICT statement, see the section “Parameter Naming Conventions for the RESTRICT, TEST, BOUNDS, and INIT Statements” on page 100.

Lagrange multipliers are reported in the “Parameter Estimates” table for all the active linear constraints. They are identified by the names Restrict1, Restrict2, and so on. The probabilities of these Lagrange multipliers are computed using a beta distribution (LaMotte 1994). Nonactive (nonbinding) restrictions have no effect on the estimation results and are not noted in the output.

The following RESTRICT statement constrains the negative binomial dispersion parameter α to 1, which restricts the conditional variance to be μ + μ²:

restrict _Alpha = 1;

TEST Statement

<label> TEST '<string>' equation1 <, equation2 > /< test-options> ;

The TEST statement performs Wald, Lagrange multiplier, and likelihood ratio tests of linear hypotheses about the regression parameters that are specified in the preceding MODEL statement.

You can add a label (which is printed in the output) to a TEST statement in two ways: add an unquoted label followed by a colon before the TEST keyword, or add a quoted string after the TEST keyword. The unquoted label cannot contain any spaces. If you include both an unquoted label and a quoted string, PROC CNTSELECT uses the unquoted label. If you specify neither an unquoted label nor a quoted string, PROC CNTSELECT automatically labels the tests.

Each equation specifies a linear hypothesis to be tested and consists of regression parameter names and relational operators. The regression parameter names are as shown in the Parameter column of the “Parameter
Estimates’ table. If a parameter name contains a blank or some other special character (such as ‘*’, ‘-‘, ‘(‘, or ‘)’), then you must use the internal name of the parameter in order to refer to that parameter in the TEST statement. You can specify the PRINTINTERNALNAMES option in the PROC CNTSELECT statement if you want to see the internal names of the parameters. When you specify the PRINTINTERNALNAMES option, an extra column is added to the “Parameter Estimates” table, which shows the internal name of each parameter.

For more information about how parameters are named in the TEST statement, see the section “Parameter Naming Conventions for the RESTRICT, TEST, BOUNDS, and INIT Statements” on page 100. Only linear equality restrictions and tests are permitted in PROC COUNTREG. Test equations can consist only of algebraic operations that involve the addition symbol (+), subtraction symbol (-), and multiplication symbol (*).

All hypotheses in one TEST statement are tested jointly.

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

- **ALL** requests Wald, Lagrange multiplier, and likelihood ratio tests.
- **LM** requests the Lagrange multiplier test.
- **LR** requests the likelihood ratio test.
- **WALD** requests the Wald test.

By default, the Wald test is performed.

The following illustrates the use of the TEST statement:

```plaintext
proc cntselect;
    model y = x1 x2 x3;
    test x1 = 0, x2 * .5 + 2 * x3 = 0;
    test _int: test intercept = 0, x3 = 0;
run;
```

The first test investigates the joint hypothesis that

\[ \beta_1 = 0 \]

and

\[ 0.5\beta_2 + 2\beta_3 = 0 \]

Only linear equality restrictions and tests are permitted in PROC CNTSELECT. Tests expressions can consist only of algebraic operations that involve the addition symbol (+), subtraction symbol (-), and multiplication symbol (*).
WEIGHT Statement

WEIGHT variable < / option> ;

The WEIGHT statement specifies a variable to supply weighting values to use for each observation in estimating parameters. The log likelihood for each observation is multiplied by the corresponding weight variable value.

If the weight of an observation is nonpositive, that observation is not used in the estimation.

The following option can be added to the WEIGHT statement after a slash (/):

NONORMALIZE
does not normalize the weights. (By default, the weights are normalized so that they add up to the actual sample size. The weights $w_i$ are normalized by multiplying them by $\frac{n}{\sum_{i=1}^{n} w_i}$, where $n$ is the sample size.) If the weights are required to be used as they are, then specify the NONORMALIZE option.

ZEROMODEL Statement

ZEROMODEL dependent-variable ~ zero-inflated-regressors < / options > ;

The ZEROMODEL statement is required if either ZIP or ZINB is specified in the DIST= option in the MODEL statement. If ZIP or ZINB is specified, then the ZEROMODEL statement must follow the MODEL statement. The dependent variable in the ZEROMODEL statement must be the same as the dependent variable in the MODEL statement.

The zero-inflated (ZI) regressors appear in the equation that determines the probability $q_i$ of a zero count. Each of these $q$ variables has a parameter to be estimated in the regression. For example, let $z_i'$ be the $i$th observation’s $1 \times (q + 1)$ vector of values of the $q$ ZI explanatory variables ($w_0$ is set to 1 for the intercept term). Then $q_i$ is a function of $z_i' \gamma$, where $\gamma$ is the $(q + 1) \times 1$ vector of parameters to be estimated. (The zero-inflated intercept is $\gamma_0$; the coefficients for the $q$ zero-inflated covariates are $\gamma_1, \ldots, \gamma_q$.) If $q$ is equal to 0 (no ZI explanatory variables are provided), then only the intercept term $\gamma_0$ is estimated. The “Parameter Estimates” table in the displayed output shows the estimates for the ZI intercept and ZI explanatory variables; they are labeled with the prefix “Inf_”. For example, the ZI intercept is labeled “Inf_intercept”. If you specify Age (a variable in your data set) as a ZI explanatory variable, then the “Parameter Estimates” table labels the corresponding parameter estimate “Inf_Age”.

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

LINK=LOGISTIC | NORMAL
specifies the distribution function used to compute probability of zeros. The supported distribution functions are as follows:

LOGISTIC specifies logistic distribution.
NORMAL specifies standard normal distribution.

If this option is omitted, then the default ZI link function is logistic.
**OFFSET=zero-inflated-offset-variable**

specifies a variable in the input data set to be used as a zero-inflated (ZI) offset variable. The ZI offset variable zero-inflated-offset-variable is included as a term, with coefficient restricted to 1, in the equation that determines the probability \( q_i \) of a zero count and represents an adjustment to a common observational unit. The ZI offset variable cannot be the response variable, the offset variable (if any), or any of the explanatory variables. The name of the data set variable that is used as the ZI offset variable is displayed in the “Model Fit Summary” table, where it is labeled as “Inf_offset”.

---

**Details: CNTSELECT Procedure**

**Missing Values**

Any observations in the input data set that have a missing value for one or more of the regressors are ignored by PROC CNTSELECT and not used in the model fit. PROC CNTSELECT rounds any positive noninteger count values to the nearest integer and ignores any observations that have a negative count.

If the input data set contains any observations that have missing response values but nonmissing regressors, PROC CNTSELECT can compute several statistics and store them in an output data set by using the OUTPUT statement. For example, you can request that the output data set contain the estimates of \( x_i' \beta \), the expected value of the response variable, and the probability that the response variable will take the current value. Furthermore, if a zero-inflated model was fit, you can request that the output data set contain the estimates of \( z_i' \gamma \), and the probability that the response is 0 as a result of the zero-generating process. Note that the presence of such observations (that have missing response values) does not affect the model fit.

---

**Poisson Regression**

The most widely used model for count data analysis is Poisson regression. Poisson regression assumes that \( y_i \), given the vector of covariates \( x_i \), is independently Poisson distributed with

\[
P(Y_i = y_i|x_i) = \frac{e^{-\mu_i} \mu_i^{y_i}}{y_i!}, \quad y_i = 0, 1, 2, \ldots
\]

and the mean parameter—that is, the mean number of events per period—is given by

\[
\mu_i = \exp(x_i' \beta)
\]

where \( \beta \) is a \((k + 1) \times 1\) parameter vector. (The intercept is \( \beta_0 \); the coefficients for the \( k \) regressors are \( \beta_1, \ldots, \beta_k \).) Taking the exponential of \( x_i' \beta \) ensures that the mean parameter \( \mu_i \) is nonnegative. It can be shown that the conditional mean is given by

\[
E(y_i|x_i) = \mu_i = \exp(x_i' \beta)
\]

Note that the conditional variance of the count random variable is equal to the conditional mean in the Poisson regression model:

\[
V(y_i|x_i) = E(y_i|x_i) = \mu_i
\]
The equality of the conditional mean and variance of \( y_i \) is known as *equidispersion*.

The standard estimator for the Poisson model is the maximum likelihood estimator (MLE). Because the observations are independent, the log-likelihood function is written as

\[
L = \sum_{i=1}^{N} (-\mu_i + y_i \ln \mu_i - \ln y_i!)
\]

For more information about the Poisson regression model, see *SAS/ETS User’s Guide*.

The Poisson model has been criticized for its restrictive property that the conditional variance equals the conditional mean. Real-life data are often characterized by *overdispersion*—that is, the variance exceeds the mean. Allowing for overdispersion can improve model predictions because the Poisson restriction of equal mean and variance results in the underprediction of zeros when overdispersion exists. The most commonly used model that accounts for overdispersion is the negative binomial model. Conway-Maxwell-Poisson regression enables you to model both overdispersion and underdispersion.

### Conway-Maxwell-Poisson Regression

The Conway-Maxwell-Poisson (CMP) distribution is a generalization of the Poisson distribution that enables you to model both underdispersed and overdispersed data. It was originally proposed by Conway and Maxwell (1962), but its implementation to model under- and overdispersed count data is attributed to Shmueli et al. (2005).

Recall that \( y_i \), given the vector of covariates \( x_i \), is independently Poisson-distributed as

\[
P(Y_i = y_i | x_i) = \frac{e^{-\lambda_i} \lambda_i^{y_i}}{y_i!}, \quad y_i = 0, 1, 2, \ldots
\]

The Conway-Maxwell-Poisson distribution is defined as

\[
P(Y_i = y_i | x_i, \lambda_i, v_i) = \frac{1}{Z(\lambda_i, v_i)} \frac{\lambda_i^{y_i}}{(y_i!)^{v_i}}, \quad y_i = 0, 1, 2, \ldots
\]

where the normalization factor is

\[
Z(\lambda_i, v_i) = \sum_{n=0}^{\infty} \frac{\lambda_i^n}{(n!)^{v_i}}
\]

and

\[
\lambda_i = \exp(x_i^T \beta)
\]

\[
v_i = -\exp(g_i^T \delta)
\]

The \( \beta \) vector is a \((k + 1) \times 1\) parameter vector. (The intercept is \( \beta_0 \), and the coefficients for the \( k \) regressors are \( \beta_1, \ldots, \beta_k \).) The \( \delta \) vector is an \((m + 1) \times 1\) parameter vector. (The intercept is represented by \( \delta_0 \), and the coefficients for the \( m \) regressors are \( \delta_1, \ldots, \delta_k \).) The covariates are represented by \( x_i \) and \( g_i \) vectors.

One of the restrictive properties of the Poisson model is that the conditional mean and variance must be equal:

\[
E(y_i | x_i) = V(y_i | x_i) = \lambda_i = \exp(x_i^T \beta)
\]
The CMP distribution overcomes this restriction by defining an additional parameter, \( \nu \), which governs the rate of decay of successive ratios of probabilities such that

\[
P(Y_i = y_i - 1)/P(Y_i = y_i) = \frac{(y_i)^{\nu_i}}{\lambda_i}
\]

The introduction of the additional parameter, \( \nu \), allows for flexibility in modeling the tail behavior of the distribution. If \( \nu = 1 \), the ratio is equal to the rate of decay of the Poisson distribution. If \( \nu < 1 \), the rate of decay decreases, enabling you to model processes that have longer tails than the Poisson distribution (overdispersed data). If \( \nu > 1 \), the rate of decay increases in a nonlinear fashion, thus shortening the tail of the distribution (underdispersed data).

There are several special cases of the Conway-Maxwell-Poisson distribution. If \( \lambda < 1 \) and \( \nu \to \infty \), the Conway-Maxwell-Poisson results in the Bernoulli distribution. In this case, the data can take only the values 0 and 1, which represents an extreme underdispersion. If \( \nu = 1 \), the Poisson distribution is recovered with its equidispersion property. When \( \nu = 0 \) and \( \lambda < 1 \), the normalization factor is convergent and forms a geometric series,

\[
Z(\lambda_i, 0) = \frac{1}{1 - \lambda_i}
\]

and the probability density function becomes

\[
P(Y = y_i; \lambda_i, \nu = 0) = (1 - \lambda_i)\lambda_i^{y_i}
\]

The geometric distribution represents a case of severe overdispersion.

**Mean, Variance, and Dispersion for the Conway-Maxwell-Poisson Model**

The mean and variance of the Conway-Maxwell-Poisson distribution are defined as

\[
E[Y] = \frac{\partial \ln Z}{\partial \ln \lambda}
\]

\[
V[Y] = \frac{\partial^2 \ln Z}{\partial^2 \ln \lambda}
\]

The Conway-Maxwell-Poisson distribution does not have closed-form expressions for its moments in terms of its parameters \( \lambda \) and \( \nu \). However, the moments can be approximated. Shmueli et al. (2005) use asymptotic expressions for \( Z \) to derive \( E(Y) \) and \( V(Y) \) as

\[
E[Y] \approx \lambda^{1/\nu} + \frac{1}{2\nu} - \frac{1}{2}
\]

\[
V[Y] \approx \frac{1}{\nu} \lambda^{1/\nu}
\]

In the Conway-Maxwell-Poisson model, the summation of infinite series is evaluated using a logarithmic expansion. The mean and variance are calculated as follows for the Shmueli et al. (2005) model:

\[
E(Y) = \frac{1}{Z(\lambda, \nu)} \sum_{j=0}^{\infty} \frac{j\lambda^j}{(j!)^\nu}
\]
Conway-Maxwell-Poisson Regression

\[ V(Y) = \frac{1}{Z(\lambda, \nu)} \sum_{j=0}^{\infty} \frac{j^2 \lambda^j}{(j!)^\nu} - E(Y)^2 \]

The dispersion is defined as

\[ D(Y) = \frac{V(Y)}{E(Y)} \]

**Likelihood Function for the Conway-Maxwell-Poisson Model**

The likelihood for a set of \( n \) independently and identically distributed variables \( y_1, y_2, \ldots, y_n \) is written as

\[
L(y_1, y_2, \ldots, y_n | \lambda, \nu) = \frac{\prod_{i=1}^{n} \lambda^{y_i}}{(\prod_{i=1}^{n} y_i!)^\nu} Z(\lambda, \nu)^{-n} \\
= \lambda^{\sum_{i=1}^{n} y_i} \exp(-\nu \sum_{i=1}^{n} \ln(y_i!)) Z(\lambda, \nu)^{-n} \\
= \lambda^{S_1} \exp(-\nu S_2) Z(\lambda, \nu)^{-n}
\]

where \( S_1 \) and \( S_2 \) are sufficient statistics for \( y_1, y_2, \ldots, y_n \). You can see from the preceding equation that the Conway-Maxwell-Poisson distribution is a member of the exponential family. The log-likelihood function can be written as

\[
\mathcal{L} = -n \ln(Z(\lambda, \nu)) + \sum_{i=1}^{n} (y_i \ln(\lambda) - \nu \ln(y_i!))
\]

The gradients can be written as

\[
\mathcal{L}_\beta = \left( \sum_{k=1}^{N} y_k - n \frac{\lambda Z(\lambda, \nu)}{Z(\lambda, \nu)} \right) x \\
\mathcal{L}_\delta = \left( \sum_{k=1}^{N} \ln(y_k!) - n \frac{Z(\lambda, \nu)}{Z(\lambda, \nu)} \right) vz
\]

**Conway-Maxwell-Poisson Regression: Guikema and Coffelt (2008) Reparameterization**

Guikema and Coffelt (2008) propose a reparameterization of the Shmueli et al. (2005) Conway-Maxwell-Poisson model to provide a measure of central tendency that can be interpreted in the context of the generalized linear model. By substituting \( \lambda = \mu^\nu \), the Guikema and Coffelt (2008) formulation is written as

\[
P(Y = y_i; \mu, \nu) = \frac{1}{S(\mu, \nu)} \left( \frac{\mu^{y_i}}{y_i!} \right)^\nu
\]

where the new normalization factor is defined as

\[
S(\mu, \nu) = \sum_{j=0}^{\infty} \left( \frac{\mu^j}{j!} \right)^\nu
\]
Chapter 3: The CNTSELECT Procedure

In terms of their new formulations, the mean and variance of \( Y \) are given as

\[
E[Y] = \frac{1}{v} \frac{\partial \ln S}{\partial \ln \mu}
\]

\[
V[Y] = \frac{1}{v^2} \frac{\partial^2 \ln S}{\partial^2 \ln \mu}
\]

They can be approximated as

\[
E[Y] \approx \mu + \frac{1}{2} v - \frac{1}{2}
\]

\[
V[Y] \approx \frac{\mu}{v}
\]

In the CNTSELECT procedure, the mean and variance are calculated according to the following formulas, respectively, for the Guikema and Coffelt (2008) model:

\[
E(Y) = \frac{1}{Z(\lambda, \mu)} \sum_{j=0}^{\infty} j \mu^v j \frac{\mu^v j}{(j!)^v}
\]

\[
V(Y) = \frac{1}{Z(\lambda, \mu)} \sum_{j=0}^{\infty} j^2 \mu^v j \frac{\mu^v j}{(j!)^v} - E(Y)^2
\]

In terms of the new parameter \( \mu \), the log-likelihood function is specified as

\[
\mathcal{L} = \ln(S(\mu, \nu)) + \nu \sum_{i=1}^{N} (y_i \ln(\mu) - \ln(y_i!))
\]

and the gradients are calculated as

\[
\mathcal{L}_\beta = \left( \nu \sum_{i=1}^{N} y_i - \frac{\mu S(\mu, \nu)}{S(\mu, \nu)} \right) x
\]

\[
\mathcal{L}_\delta = \left( \sum_{i=1}^{N} (y_i \ln(\mu) - \ln(y_i!)) - \frac{S(\mu, \nu) y_i}{S(\mu, \nu)} \right) \nu g
\]

By default, the CNTSELECT procedure uses the Guikema and Coffelt (2008) specification. The Shmueli et al. (2005) model can be estimated by specifying the PARAMETER=LAMBDA option. If you specify DISP=CMPOISSON in the MODEL statement and you omit the DISPMODEL statement, the model is estimated according to the Lord, Guikema, and Geedipally (2008) specification, where \( \nu \) represents a single parameter that does not depend on any covariates. The Lord, Guikema, and Geedipally (2008) specification makes the model comparable to the negative binomial model because it has only one parameter.

The dispersion is defined as

\[
D(Y) = \frac{V(Y)}{E(Y)}
\]
Using the Guikema and Coffelt (2008) specification results in the integral part of $\mu$ representing the mode, which is a reasonable approximation for the mean. The dispersion can be written as

$$D(Y) = \frac{V(Y)}{E(Y)} \approx \frac{\mu}{\mu + \frac{1}{2}v - \frac{1}{2}} \approx \frac{1}{v}$$

When $v < 1$, the variance can be shown to be greater than the mean and the dispersion greater than 1. This is a result of overdispersed data. When $v = 1$ and the mean and variance are equal, the dispersion is equal to 1 (Poisson model). When $v > 1$, the variance is smaller than the mean and the dispersion is less than 1. This is a result of underdispersed data.

All Conway-Maxwell-Poisson models in the CNTSELECT procedure are parameterized in terms of dispersion, where

$$-\ln(v) = \delta_0 + \sum_{n=1}^{q} \delta_n g_n$$

Negative values of $\ln(v)$ indicate that the data are approximately overdispersed, and positive values of $\ln(v)$ indicate that the data are approximately underdispersed.

---

**Negative Binomial Regression**

The Poisson regression model can be generalized by introducing an unobserved heterogeneity term for observation $i$. Thus, the individuals are assumed to differ randomly in a manner that is not fully accounted for by the observed covariates. This is formulated as

$$E(y_i|x_i, \tau_i) = \mu_i \tau_i = e^{x_i^T \beta + \epsilon_i}$$

where the unobserved heterogeneity term $\tau_i = e^{\epsilon_i}$ is independent of the vector of regressors $x_i$. Then the distribution of $y_i$ conditional on $x_i$ and $\tau_i$ is Poisson with conditional mean and conditional variance $\mu_i \tau_i$:

$$f(y_i|x_i, \tau_i) = \frac{\exp(-\mu_i \tau_i) (\mu_i \tau_i)^{y_i}}{y_i!}$$

Let $g(\tau_i)$ be the probability density function of $\tau_i$. Then, the distribution $f(y_i|x_i)$ (no longer conditional on $\tau_i$) is obtained by integrating $f(y_i|x_i, \tau_i)$ with respect to $\tau_i$:

$$f(y_i|x_i) = \int_0^{\infty} f(y_i|x_i, \tau_i) g(\tau_i) d\tau_i$$

An analytical solution to this integral exists when $\tau_i$ is assumed to follow a gamma distribution. This solution is the negative binomial distribution. If the model contains a constant term, then in order to identify the mean of the distribution, it is necessary to assume that $E(e^{\epsilon_i}) = E(\tau_i) = 1$. Thus, it is assumed that $\tau_i$ follows a gamma($\theta, \theta$) distribution with $E(\tau_i) = 1$ and $V(\tau_i) = 1/\theta$.

$$g(\tau_i) = \frac{\theta^\theta}{\Gamma(\theta)} \tau_i^{\theta - 1} \exp(-\theta \tau_i)$$
where \( \Gamma(x) = \int_0^\infty z^{x-1} \exp(-z) dz \) is the gamma function and \( \theta \) is a positive parameter. Then, the density of \( y_i \) given \( x_i \) is derived as

\[
f(y_i|x_i) = \int_0^\infty f(y_i|x_i, \tau_i)g(\tau_i)d\tau_i
\]

\[
= \frac{\theta^\theta \mu_i^{y_i}}{y_i! \Gamma(\theta)} \int_0^\infty e^{-(\mu_i+\theta)\tau_i} \tau_i^{\theta+y_i-1} d\tau_i
\]

\[
= \frac{\theta^\theta \mu_i^{y_i} \Gamma(y_i+\theta)}{y_i! \Gamma(\theta)(\theta+\mu_i)^{\theta+y_i}}
\]

\[
= \frac{\Gamma(y_i+\theta)}{y_i! \Gamma(\theta)} \left( \frac{\theta}{\theta+\mu_i} \right)^{\theta} \left( \frac{\mu_i}{\theta+\mu_i} \right)^{y_i}
\]

If you make the substitution \( \alpha = \frac{1}{\theta} (\alpha > 0) \), the negative binomial distribution can then be rewritten as

\[
f(y_i|x_i) = \frac{\Gamma(y_i+\alpha^{-1})}{y_i! \Gamma(\alpha^{-1})} \left( \frac{\alpha^{-1}}{\alpha^{-1}+\mu_i} \right)^{\alpha^{-1}} \left( \frac{\mu_i}{\alpha^{-1}+\mu_i} \right)^{y_i}, \quad y_i = 0, 1, 2, \ldots
\]

Thus, the negative binomial distribution is derived as a gamma mixture of Poisson random variables. It has the conditional mean

\[E(y_i|x_i) = \mu_i = e^{\beta'x_i}\]

and the conditional variance

\[V(y_i|x_i) = \mu_i[1 + \frac{1}{\theta} \mu_i] = \mu_i[1 + \alpha \mu_i] > E(y_i|x_i)\]

The conditional variance of the negative binomial distribution exceeds the conditional mean. Overdispersion results from neglected unobserved heterogeneity. The negative binomial model with variance function \( V(y_i|x_i) = \mu_i + \alpha \mu_i^2 \), which is quadratic in the mean, is referred to as the NEGBIN2 model Cameron and Trivedi (1986). To estimate this model, specify DIST=NEGBIN(P=2) in the MODEL statement. The Poisson distribution is a special case of the negative binomial distribution where \( \alpha = 0 \). A test of the Poisson distribution can be carried out by testing the hypothesis that \( \alpha = \frac{1}{\theta'} = 0 \). A Wald test of this hypothesis is provided (it is the reported \( t \) statistic for the estimated \( \alpha \) in the negative binomial model).

The log-likelihood function of the negative binomial regression model (NEGBIN2) is given by

\[
L = \sum_{i=1}^{N} \left\{ \sum_{j=0}^{y_i-1} \ln(j + \alpha^{-1}) - \ln(y_i!) \right.
\]

\[
- (y_i + \alpha^{-1}) \ln(1 + \alpha \exp(x'_i \beta)) + y_i \ln(\alpha) + y_i x'_i \beta \right\}
\]

where use of the following fact is made if \( y \) is an integer:

\[
\Gamma(y + a)/\Gamma(a) = \prod_{j=0}^{y-1} (j + a)
\]
Cameron and Trivedi (1986) consider a general class of negative binomial models that have mean \( \mu_i \) and variance function \( \mu_i + \alpha \mu_i^p \). The NEGBIN2 model, with \( p = 2 \), is the standard formulation of the negative binomial model. Models that have other values of \( p, -\infty < p < \infty \), have the same density \( f(y_i | x_i) \), except that \( \alpha^{-1} \) is replaced everywhere by \( \alpha^{-1} \mu^{2-p} \). The negative binomial model NEGBIN1, which sets \( p = 1 \), has the variance function \( V(y_i | x_i) = \mu_i + \alpha \mu_i \), which is linear in the mean. To estimate this model, specify DIST=NEGBIN(P=1) in the MODEL statement.

The log-likelihood function of the NEGBIN1 regression model is given by

\[
L = \sum_{i=1}^{N} \left\{ \sum_{j=0}^{y_i-1} \ln \left( j + \alpha^{-1} \exp(x'_i \beta) \right) - \ln(y_i!) - \left( y_i + \alpha^{-1} \exp(x'_i \beta) \right) \ln(1 + \alpha) + y_i \ln(\alpha) \right\}
\]

For more information about the negative binomial regression model, see SAS/ETS User’s Guide.

Zero-Inflated Count Regression Overview

The main motivation for using zero-inflated count models is that real-life data frequently display overdispersion and excess zeros. Zero-inflated count models provide a way to both model the excess zeros and allow for overdispersion. In particular, there are two possible data generation processes for each observation. The result of a Bernoulli trial is used to determine which of the two processes to use. For observation \( i \), Process 1 is chosen with probability \( \varphi_i \) and Process 2 with probability \( 1 - \varphi_i \). Process 1 generates only zero counts. Process 2 generates counts from either a Poisson or a negative binomial model. In general,

\[
y_i \sim \begin{cases} 0 & \text{with probability } \varphi_i \\ g(y_i) & \text{with probability } 1 - \varphi_i \end{cases}
\]

Therefore, the probability of \( \{Y_i = y_i\} \) can be described as

\[
P(y_i = 0 | x_i) = \varphi_i + (1 - \varphi_i) g(0)
\]

\[
P(y_i | x_i) = (1 - \varphi_i) g(y_i), \quad y_i > 0
\]

where \( g(y_i) \) follows either the Poisson or the negative binomial distribution.

If the probability \( \varphi_i \) depends on the characteristics of observation \( i \), then \( \varphi_i \) is written as a function of \( z'_i \gamma \), where \( z'_i \) is the \( 1 \times (q + 1) \) vector of zero-inflated covariates and \( \gamma \) is the \( (q + 1) \times 1 \) vector of zero-inflated coefficients to be estimated. (The zero-inflated intercept is \( \gamma_0 \); the coefficients for the \( q \) zero-inflated covariates are \( \gamma_1, \ldots, \gamma_q \).) The function \( F \) that relates the product \( z'_i \gamma \) (which is a scalar) to the probability \( \varphi_i \) is called the zero-inflated link function,

\[
\varphi_i = F_i = F(z'_i \gamma)
\]

In the CNTSELECT procedure, the zero-inflated covariates are indicated in the ZEROMODEL statement. Furthermore, the zero-inflated link function \( F \) can be specified as either the logistic function,

\[
F(z'_i \gamma) = \Lambda(z'_i \gamma) = \frac{\exp(z'_i \gamma)}{1 + \exp(z'_i \gamma)}
\]
or the standard normal cumulative distribution function (also called the probit function),

\[
F(z_i') = \Phi(z_i') = \int_{0}^{z_i'} \frac{1}{\sqrt{2\pi}} \exp(-u^2/2) du
\]

The zero-inflated link function is indicated by using the LINK= option in the ZEROMODEL statement. The default ZI link function is the logistic function.

---

**Zero-Inflated Poisson Regression**

In the zero-inflated Poisson (ZIP) regression model, the data generation process that is referred to earlier as Process 2 is

\[
g(y_i) = \frac{\exp(-\mu_i)\mu_i^{y_i}}{y_i!}
\]

where \( \mu_i = e^{x_i'\beta} \). Thus the ZIP model is defined as

\[
P(y_i = 0|x_i, z_i) = F_i + (1 - F_i) \exp(-\mu_i)
\]

\[
P(y_i|x_i, z_i) = (1 - F_i) \frac{\exp(-\mu_i)\mu_i^{y_i}}{y_i!}, \quad y_i > 0
\]

The conditional expectation and conditional variance of \( y_i \) are given by

\[
E(y_i|x_i, z_i) = \mu_i(1 - F_i)
\]

\[
V(y_i|x_i, z_i) = E(y_i|x_i, z_i)(1 + \mu_i F_i)
\]

Note that the ZIP model (in addition to the ZINB model) exhibits overdispersion because \( V(y_i|x_i, z_i) > E(y_i|x_i, z_i) \).

In general, the log-likelihood function of the ZIP model is

\[
\mathcal{L} = \sum_{i=1}^{N} \ln [P(y_i|x_i, z_i)]
\]

After a specific link function (either logistic or standard normal) for the probability \( \varphi_i \) is chosen, it is possible to write the exact expressions for the log-likelihood function and the gradient.

**ZIP Model with Logistic Link Function**

First, consider the ZIP model in which the probability \( \varphi_i \) is expressed by a logistic link function, namely

\[
\varphi_i = \frac{\exp(z_i'\gamma)}{1 + \exp(z_i'\gamma)}
\]
The log-likelihood function is

$$
\mathcal{L} = \sum_{\{i: y_i = 0\}} \ln \left[ \exp(z_i' \gamma) + \exp(- \exp(x_i' \beta)) \right] \\
+ \sum_{\{i: y_i > 0\}} \left[ y_i x_i' \beta - \exp(x_i' \beta) - \sum_{k=2}^{y_i} \ln(k) \right] \\
- \sum_{i=1}^{N} \ln \left[ 1 + \exp(z_i' \gamma) \right]
$$

ZIP Model with Standard Normal Link Function

Next, consider the ZIP model in which the probability \( \psi_i \) is expressed by a standard normal link function: \( \psi_i = \Phi(z_i' \gamma) \). The log-likelihood function is

$$
\mathcal{L} = \sum_{\{i: y_i = 0\}} \ln \left\{ \Phi(z_i' \gamma) + \left[ 1 - \Phi(z_i' \gamma) \right] \exp(- \exp(x_i' \beta)) \right\} \\
+ \sum_{\{i: y_i > 0\}} \left\{ \ln \left[ (1 - \Phi(z_i' \gamma)) \right] - \exp(x_i' \beta) + y_i x_i' \beta - \sum_{k=2}^{y_i} \ln(k) \right\}
$$

For more information about the zero-inflated Poisson regression model, see SAS/ETS User’s Guide.

Zero-Inflated Conway-Maxwell-Poisson Regression

In the Conway-Maxwell-Poisson regression model, the data generation process is defined as

$$
P(Y_i = y_i | x_i, z_i) = \frac{1}{Z(\lambda_i, v_i)} \frac{\lambda_i^{y_i}}{(y_i!)^{v_i}}, \quad y_i = 0, 1, 2, \ldots
$$

where the normalization factor is

$$
Z(\lambda_i, v_i) = \sum_{n=0}^{\infty} \frac{\lambda_i^n}{(n!)^{v_i}}
$$

and

$$
\lambda_i = \exp(x_i' \beta) \\
v_i = - \exp(g_i' \delta)
$$

The zero-inflated Conway-Maxwell-Poisson model can be written as

$$
P(y_i | x_i, z_i) = F_i + (1 - F_i) \frac{1}{Z(\lambda_i, v_i)} \frac{\lambda_i^{y_i}}{(y_i!)^{v_i}}, \quad y_i = 0 \\
P(y_i | x_i, z_i) = (1 - F_i) \frac{1}{Z(\lambda_i, v_i)} \frac{\lambda_i^{y_i}}{(y_i!)^{v_i}}, \quad y_i > 0
$$
The conditional expectation and conditional variance of \( y_i \) are given respectively by

\[
E(y_i|x_i, z_i) = (1 - F_i) \frac{1}{Z(\lambda, v)} \sum_{j=0}^{\infty} \frac{j \lambda^j}{(j!)^v}
\]

\[
V(y_i|x_i, z_i) = (1 - F_i) \frac{1}{Z(\lambda, v)} \sum_{j=0}^{\infty} \frac{j^2 \lambda^j}{(j!)^v} - E(y_i|x_i, z_i)^2
\]

The general form of the log-likelihood function for the Conway-Maxwell-Poisson zero-inflated model is

\[
L = \sum_{i=1}^{N} w_i \ln [P(y_i|x_i, z_i)]
\]

**Zero-Inflated Conway-Maxwell-Poisson Model with Logistic Link Function**

For this model, the probability \( \varphi_i \) is expressed by using a logistic link function as

\[
\varphi_i = \Lambda(z_i' \gamma) = \frac{\exp(z_i' \gamma)}{1 + \exp(z_i' \gamma)}
\]

The log-likelihood function is

\[
L = \sum_{\{i: y_i = 0\}} w_i \ln \left\{ \Lambda(z_i' \gamma) + \left[ 1 - \Lambda(z_i' \gamma) \right] \frac{1}{Z(\lambda_i, v_i)} \right\}
+ \sum_{\{i: y_i > 0\}} w_i \left\{ \ln \left[ (1 - \Lambda(z_i' \gamma)) \right] - \ln(Z(\lambda_i, v_i)) + (y_i \ln(\lambda) - v \ln(y_i!)) \right\}
\]

**Zero-Inflated Conway-Maxwell-Poisson Model with Normal Link Function**

For this model, the probability \( \varphi_i \) is specified by using the standard normal distribution function (probit function): \( \varphi_i = \Phi(z_i' \gamma) \).

The log-likelihood function is written as

\[
L = \sum_{\{i: y_i = 0\}} w_i \ln \left\{ \Phi(z_i' \gamma) + \left[ 1 - \Phi(z_i' \gamma) \right] \frac{1}{Z(\lambda_i, v_i)} \right\}
+ \sum_{\{i: y_i > 0\}} w_i \left\{ \ln \left[ (1 - \Phi(z_i' \gamma)) \right] - \ln(Z(\lambda_i, v_i)) + (y_i \ln(\lambda) - v \ln(y_i!)) \right\}
\]
Zero-Inflated Negative Binomial Regression

The zero-inflated negative binomial (ZINB) model in PROC CNTSELECT is based on the negative binomial model that has a quadratic variance function (when DIST=NEGBIN in the MODEL or PROC CNTSELECT statement). The ZINB model is obtained by specifying a negative binomial distribution for the data generation process referred to earlier as Process 2:

\[
g(y_i) = \frac{\Gamma(y_i + \alpha^{-1})}{y_i!\Gamma(\alpha^{-1})} \left( \frac{\alpha^{-1}}{\alpha^{-1} + \mu_i} \right)^{\alpha^{-1}} \left( \frac{\mu_i}{\alpha^{-1} + \mu_i} \right)^{y_i}
\]

Thus the ZINB model is defined to be

\[
P(y_i = 0|x_i, z_i) = F_i + (1 - F_i)(1 + \alpha \mu_i)^{-\alpha^{-1}}
\]

\[
P(y_i|x_i, z_i) = (1 - F_i) \frac{\Gamma(y_i + \alpha^{-1})}{y_i!\Gamma(\alpha^{-1})} \left( \frac{\alpha^{-1}}{\alpha^{-1} + \mu_i} \right)^{\alpha^{-1}}
\]

\[
\times \left( \frac{\mu_i}{\alpha^{-1} + \mu_i} \right)^{y_i}, \quad y_i > 0
\]

In this case, the conditional expectation (E) and conditional variance (V) of \( y_i \) are

\[
E(y_i|x_i, z_i) = \mu_i (1 - F_i)
\]

\[
V(y_i|x_i, z_i) = E(y_i|x_i, z_i) [1 + \mu_i (F_i + \alpha)]
\]

Like the ZIP model, the ZINB model exhibits overdispersion because the conditional variance exceeds the conditional mean.

**ZINB Model with Logistic Link Function**

In this model, the probability \( \phi_i \) is given by the logistic function, namely

\[
\phi_i = \frac{\exp(z_i' \gamma)}{1 + \exp(z_i' \gamma)}
\]

The log-likelihood function is

\[
\mathcal{L} = \sum_{\{i:y_i = 0\}} \ln \left[ \exp(z_i' \gamma) + (1 + \alpha \exp(z_i' \beta))^{-\alpha^{-1}} \right]
\]

\[
+ \sum_{\{i:y_i > 0\}} \sum_{j=0}^{y_i-1} \ln(j + \alpha^{-1})
\]

\[
+ \sum_{\{i:y_i > 0\}} \left\{ -\ln(y_i!) - (y_i + \alpha^{-1}) \ln(1 + \alpha \exp(z_i' \beta)) + y_i \ln(\alpha) + y_i z_i' \beta \right\}
\]

\[- \sum_{i=1}^{N} \ln \left[ 1 + \exp(z_i' \gamma) \right]\]
ZINB Model with Standard Normal Link Function

For this model, the probability \( \psi_i \) is expressed by the standard normal distribution function (probit function):

\[
\psi_i = \Phi(z_i^\prime \gamma).
\]

The log-likelihood function is

\[
\mathcal{L} = \sum_{i: y_i = 0} \ln \left\{ \Phi(z_i^\prime \gamma) + \left[ 1 - \Phi(z_i^\prime \gamma) \right] \left( 1 + \alpha \exp(x_i^\prime \beta) \right)^{-\alpha^{-1}} \right\} \\
+ \sum_{i: y_i > 0} \ln \left[ 1 - \Phi(z_i^\prime \gamma) \right] \\
+ \sum_{i: y_i > 0} \sum_{j=0}^{y_i-1} \{ \ln (j + \alpha^{-1}) \} \\
- \sum_{i: y_i > 0} \ln (y_i !) \\
- \sum_{i: y_i > 0} (y_i + \alpha^{-1}) \ln (1 + \alpha \exp(x_i^\prime \beta)) \\
+ \sum_{i: y_i > 0} y_i \ln (\alpha) \\
+ \sum_{i: y_i > 0} y_i x_i^\prime \beta
\]

For more information about the zero-inflated negative binomial regression model, see SAS/ETS User’s Guide.

Parameter Naming Conventions for the RESTRICT, TEST, BOUNDS, and INIT Statements

This section describes how you can refer to the parameters in the MODEL, ZEROMODEL, and DISPMODEL statements when you use the RESTRICT, TEST, BOUNDS, or INIT statement. The following examples use the RESTRICT statement, but the same remarks apply to naming parameters when you use the TEST, BOUNDS, or INIT statement. The names of the parameters can be seen in the Parameter Estimates table if you specify the option. If a parameter name contains a blank or some other special character (such as ‘*’, ‘-’, ‘(’, or ‘)’), then you must use the internal name of the parameter in order to refer to that parameter in the RESTRICT, TEST, BOUNDS, or INIT statement. You can specify the PRINTINTERNALNAMES option in the PROC CNTSELECT statement if you want to see the internal names of the parameters. When you specify the PRINTINTERNALNAMES option, an extra column is added to the “Parameter Estimates” table, which shows the internal name of each parameter.

To impose a restriction on a parameter that is related to a regressor in the MODEL statement, you simply use the name of the regressor itself to refer to its associated parameter. Suppose your model is

\[
\text{model } y = x_1 \ x_2 \ x_5;
\]

where \( x_1 \) through \( x_5 \) are continuous variables. If you want to restrict the parameter associated with the regressor \( x_5 \) to be greater than 1.7, then you should use the following statement:
RESTRICT x5 > 1.7;

To impose a restriction on a parameter associated with a regressor in the ZEROMODEL statement, you can form the name of the parameter by prefixing $\text{Inf}_\cdot$ to the name of the regressor. Suppose your MODEL and ZEROMODEL statements are as follows:

```plaintext
model y = x1 x2 x5;
zeromodel y ~ x3 x5;
```

If you want to restrict the parameter related to the $x5$ regressor in the ZEROMODEL statement to be less than 1.0, then you refer to the parameter as $\text{Inf}_x5$ and provide the following statement:

```
RESTRICT Inf_x5 < 1.0;
```

Even though the regressor $x5$ appears in both the MODEL and ZEROMODEL statements, the parameter associated with $x5$ in the MODEL statement is, of course, different from the parameter associated with $x5$ in the ZEROMODEL statement. Thus, when the name of a regressor is used in a RESTRICT statement without any prefix, it refers to the parameter associated with that regressor in the MODEL statement. Meanwhile, when the name of a regressor is used in a RESTRICT statement with the prefix $\text{Inf}_\cdot$, it refers to the parameter associated with that regressor in the ZEROMODEL statement. The parameter associated with the intercept in the ZEROMODEL is named $\text{Inf}_\text{Intercept}$.

In a similar way, you can form the name of a parameter associated with a regressor in the DISPMODEL statement by prefixing $\text{Dsp}_\cdot$ to the name of the regressor. The parameter associated with the intercept in the DISPMODEL is named $\text{Dsp}_\text{Intercept}$.

**Referring to Class-Level Parameters**

When your MODEL includes a classification variable, you can impose restrictions on the parameters associated with each of the levels that are related to the classification variable as follows.

Suppose your classification variable is named $C$ and it has three levels: 0, 1, 2. Suppose your model is the following:

```plaintext
class C;
model y = x1 x2 C;
```

Adding a classification variable as a regressor to your model introduces additional parameters into your model, each of which is associated with one of the levels of the classification variable. You can form the name of the parameter associated with a particular level of your class variable by inserting the underscore character between the name of the classification variable and the value of the level. Thus, to restrict the parameter associated with level 0 of the classification variable $C$ to always be greater than 0.7, you refer to the parameter as $C_0$ and provide the following statement:

```
RESTRICT C_0 > 0.7;
```

**Referring to Parameters Associated with Interactions between Regressors**

When a regressor in your model involves an interaction between other regressors, you can impose restrictions on the parameters associated with the interaction.

Suppose you have the following model:
You can form the name of the parameter associated with the interaction regressor \( x_3 \times x_4 \) by replacing the multiplication sign with an underscore. Thus, \( x_3 \_x_4 \) refers to the parameter that is associated with the interaction regressor \( x_3 \times x_4 \).

Referring to interactions between regressors and classification variables is handled in the same way. Suppose you have a classification variable that is named \( C \) and has three levels: 0, 1, 2. Suppose that your model is the following:

```plaintext
class C;
model y = x1 x2 C\_x_3;
```

The interaction between the continuous variable \( x_3 \) and the classification variable \( C \) introduces three additional parameters, which are named \( x_3 \_C\_0 \), \( x_3 \_C\_1 \), and \( x_3 \_C\_2 \). Note how, although the order of the terms in the interaction is \( C \) followed by \( x_3 \), the name of the parameter associated with the interaction is formed by placing the name of the continuous variable \( x_3 \) first, followed by an underscore, followed by the name of the classification variable \( C \), followed by an underscore, and then followed by the level value. Once again, depending on the parameterization you specify in your CLASS statement, for each interaction in your model that involves a classification variable, one of the parameters associated with that interaction might be dropped from your model prior to optimization.

The name of a parameter associated with a nested interaction is formed in a slightly different way. Suppose you have a classification variable that is named \( C \) and has three levels: 0, 1, 2. Suppose that your model is the following:

```plaintext
class C;
model y = x1 x2 x3(C);
```

The nested interaction between the continuous variable \( x_3 \) and the classification variable \( C \) introduces three additional parameters, which are named \( x_3 \_C\_0 \), \( x_3 \_C\_1 \), and \( x_3 \_C\_2 \). Note how the name in each case is formed from the name of the regressor by replacing the left and right parentheses with underscores and then appending another underscore followed by the level value.

**Referring to Class Level Parameters with Negative Values**

When the value of a level is a negative number, you must replace the minus sign with an underscore when you form the name of the parameter that is associated with that particular level of the classification variable. For example, suppose your classification variable is named \( D \) and has four levels: –1, 0, 1, 2. Suppose your model is the following:

```plaintext
class D;
model y = x1 x2 D;
```

To restrict the parameter that is associated with level –1 of the classification variable \( D \) to always be less than 0.4, you refer to the parameter as \( D\_\_1 \) (note that there are two underscores in this parameter name: one to connect the name of the classification variable to its value and the other to replace the minus sign in the value itself) and provide the following statement:
RESTRICT D__1 < 0.4;

**Dropping a Class Level Parameter to Avoid Collinearity**

Depending on the parameterization you impose on your classification variable, one of the parameters associated with its levels might be dropped from your model prior to optimization in order to avoid collinearity. For example, when the default parameterization GLM is imposed, the parameter that is associated with the last level of your classification variable is dropped prior to optimization. If you attempt to impose a restriction on a dropped parameter by using the RESTRICT statement, PROC COUNTREG issues an error message in the log.

For example, suppose again that your classification variable is named C and that it has three levels: 0, 1, 2. Suppose your model is the following:

```plaintext
class C;
model y = x1 x2 C;
```

Because no additional options are specified in the CLASS statement, GLM parameterization is assumed. This means that the parameter named C_2 (which is the parameter associated with the last level of your classification variable) will be dropped from your model before the optimizer is invoked. Therefore, an error will be issued if you attempt to restrict the C_2 parameter in any way by referring to it in a RESTRICT statement. For example, the following RESTRICT statement will generate an error:

```plaintext
RESTRICT C_2 < 0.3;
```

**Referring to Implicit Parameters**

For certain model types, one or more implicit parameters will be added to your model prior to optimization. You can impose restrictions on these implicit parameters.

For the Poisson model for which ERRORCOMP=RANDOM is specified, PROC COUNTREG automatically adds the _Alpha parameter to your model.

If no ERRORCOMP= option is specified, for zero-inflated binomial and negative binomial models, PROC COUNTREG adds the _Alpha parameter to the model. If ERRORCOMP=RANDOM is specified for the zero-inflated binomial and negative binomial models, then PROC COUNTREG adds two implicit parameters to the model: _Alpha and _Beta.

For Conway-Maxwell Poisson models that do not include a DISPMODEL statement, the _lnNu parameter is added to the model.

Whenever your model type dictates the addition of one or more of these implicit parameters, you can impose restrictions on the implicit parameters by referring to them by name in a RESTRICT statement. For example, if your model type implies the existence of the _Alpha parameter, you can restrict _Alpha to be greater than 0.2 as follows:

```plaintext
RESTRICT _Alpha > 0.2;
```
Computational Resources

The time and memory that PROC CNTSELECT requires are proportional to the number of parameters in the model and the number of observations in the data set being analyzed. Less time and memory are required for smaller models and fewer observations. When PROC CNTSELECT is run in the high-performance distributed environment, the amount of time required is also affected by the number of nodes and the number of threads per node as specified in the PERFORMANCE statement.

The method that is chosen to calculate the variance-covariance matrix and the optimization method also affect the time and memory resources. All optimization methods available through the METHOD= option have similar memory use requirements. The processing time might differ for each method, depending on the number of iterations and functional calls needed. The data set is read into memory to save processing time. If not enough memory is available to hold the data, the CNTSELECT procedure stores the data in a utility file on disk and rereads the data as needed from this file, substantially increasing the execution time of the procedure. The gradient and the variance-covariance matrix must be held in memory. If the model has \( p \) parameters including the intercept, then at least \( 8 \times (p + p \times (p + 1)/2) \) bytes of memory are needed. The processing time is also a function of the number of iterations needed to converge to a solution for the model parameters. The number of iterations that are needed cannot be known in advance. You can use the MAXITER= option to limit the number of iterations that PROC CNTSELECT executes. You can alter the convergence criteria by using the nonlinear optimization options available in the PROC CNTSELECT statement. For a list of all the nonlinear optimization options, see “Optimization Control Options” on page 78.

Covariance Matrix Types

The COVEST= option in the PROC CNTSELECT statement enables you to specify the estimation method for the covariance matrix. COVEST=HESSIAN estimates the covariance matrix that is based on the inverse of the Hessian matrix; COVEST=OP uses the outer product of gradients; and COVEST=QML produces the covariance matrix that is based on both the Hessian and outer product matrices. Although all three methods produce asymptotically equivalent results, they differ in computational intensity and produce results that might differ in finite samples. The COVEST=OP option provides the covariance matrix that is typically the easiest to compute. In some cases, the OP approximation is considered more efficient than the Hessian or QML approximation because it contains fewer random elements. The QML approximation is computationally the most complex because it requires both the outer product of gradients and the Hessian matrix. In most cases, the OP or Hessian approximation is preferred to QML. The need for QML approximation arises in cases where the model is misspecified and the information matrix equality does not hold. The default is COVEST=HESSIAN.

Displayed Output

PROC CNTSELECT produces the following displayed output.

Model Fit Summary

The “Model Fit Summary” table contains the following information:
A line below the “Model Fit Summary” table indicates whether the algorithm successfully converged.

**Parameter Estimates**

The “Parameter Estimates” table gives the estimates of the model parameters. In zero-inflated (ZI) models, estimates are also given for the ZI intercept and ZI regressor parameters, which are labeled with the prefix “Inf_”. For example, the ZI intercept is labeled “Inf_intercept”. If you specify “Age” as a ZI regressor, then the “Parameter Estimates” table labels the corresponding parameter estimate “Inf_Age”. If you do not list any ZI regressors, then only the ZI intercept term is estimated.

If the DISPMODEL statement is specified for the Conway-Maxwell-Poisson model, the estimates are given for the dispersion intercept, and parameters are labeled with the prefix “Dsp_”. For example, the dispersion model intercept is labeled “Dsp_Intercept”. If you specify “Education” as a dispersion model regressor, then
the “Parameter Estimates” table labels the corresponding parameter estimate “Dsp_Education”. If you do not list any dispersion regressors, then only the dispersion intercept is estimated.

“_Alpha” is the negative binomial dispersion parameter. The $t$ statistic that is given for “_Alpha” is a test of overdispersion.

**Covariance of Parameter Estimates**

If you specify the COVB option in the PROC CNTSELECT or MODEL statement, the CNTSELECT procedure displays the estimated covariance matrix, which is defined as the inverse of the information matrix at the final iteration.

**Correlation of Parameter Estimates**

If you specify the CORRB option in the PROC CNTSELECT or MODEL statement, the CNTSELECT procedure displays the estimated correlation matrix, which is based on the Hessian matrix used at the final iteration.

**OUTPUT OUT= Data Set**

The OUTPUT statement creates a new SAS data set that contains various estimates that you specify. You can request that the output data set contain the estimates of $\hat{x}^T \beta$, the expected value of the response variable, and the probability that the response variable will take the current value. In a zero-inflated model, you can also request that the output data set contain the estimates of $z_i^T \gamma$, and the probability that the response is zero as a result of the zero-generating process. In a Conway-Maxwell-Poisson model, you can also request that the output data set contains estimates of $g_i^T \delta$, $\lambda$, $v$, $\mu$, mode, variance and dispersion.

Except for the probability of the current value, these statistics can be computed for all observations in which the regressors are not missing, even if the response is missing. By adding observations with missing response values to the input data set, you can compute these statistics for new observations or for settings of the regressors that are not present in the data without affecting the model fit. Because of potential space limitations on the client workstation, the data set that is created by the OUTPUT statement does not contain the variables in the input data set.

**ODS Table Names**

PROC CNTSELECT assigns a name to each table that it creates. You can use these names to denote the table when you use the Output Delivery System (ODS) to select tables and create output data sets. These table names are listed in Table 3.2.

<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>
</tbody>
</table>
Examples: The CNTSELECT Procedure

Example 3.1: Zero-Inflated Poisson Model with CLASS Statement

This example shows the use of the CNTSELECT procedure to estimate a zero-inflated Poisson model with a classification variable Group that has two levels. The following DATA step generates 10,000 replicates from the zero-inflated Poisson (ZIP) model. The first 5,000 replicates belong to the first group, and the second 5,000 replicates belong to the second group. The model contains seven variables and three variables that correspond to the zero-inflated process.

```sas
data simulate;
  call streaminit(12345);
  array vars x1-x7;
  array zero_vars z1-z3;
  array parms(7) (.3 .4 .2 .4 -.3 -.5 -.3);
  array zero_parms(3) (-.6 .3 .2);
  intercept=0.5;
  group=1;
  z_intercept=-1;
  theta=0.5;
  do i=1 to 10000;
    sum_xb=0;
    sum_gz=0;
    if i>5000 then do;
      intercept=2;
      group=2;
    end;
    /* Additional code for zero-inflated Poisson model */
  end;
```

---

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Statement</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConvergenceStatus</td>
<td>Convergence status</td>
<td>MODEL</td>
<td>Default</td>
</tr>
<tr>
<td>Correlation</td>
<td>Correlation of parameter estimates</td>
<td>MODEL</td>
<td>CORRB</td>
</tr>
<tr>
<td>Covariance</td>
<td>Covariance of parameter estimates</td>
<td>MODEL</td>
<td>COVB</td>
</tr>
<tr>
<td>DepVarSummary</td>
<td>Summary of dependent variable statistics</td>
<td>PROC</td>
<td>DEPVARSUMMARY</td>
</tr>
<tr>
<td>FitModelSummary</td>
<td>Summary of nonlinear estimation</td>
<td>MODEL</td>
<td>Default</td>
</tr>
<tr>
<td>ParameterEstimates</td>
<td>Parameter estimates</td>
<td>MODEL</td>
<td>Default</td>
</tr>
<tr>
<td>TestResults</td>
<td>Hypothesis test statistics</td>
<td>TEST</td>
<td>Default</td>
</tr>
<tr>
<td>TimingDetails</td>
<td>Detailed summary of time taken</td>
<td>PROC</td>
<td>PRINTTIMING=(DETAILS)</td>
</tr>
<tr>
<td>TimingSummary</td>
<td>Summary of time taken for main phases of execution</td>
<td>PROC</td>
<td>PRINTTIMING</td>
</tr>
</tbody>
</table>
end;
  do j=1 to 7;
    vars[j]=rand('NORMAL',0,1);
    sum_xb=sum_xb+parms[j]*vars[j];
  end;
  mu=exp(intercept+sum_xb);
  y_p=rand('POISSON', mu);

  do j=1 to 3;
    zero_vars[j]=rand('NORMAL',0,1);
    sum_gz = sum_gz+zero_parms[j]*zero_vars[j];
  end;
  z_gamma = z_intercept+sum_gz;
  pzero = cdf('LOGISTIC',z_gamma);
  cut=rand('UNIFORM');
  if cut<pzero then y_p=0;
  output;
end;
keep y_p group x1-x7 z1-z3;
run;

The following statements estimate a zero-inflated Poisson model with the classification variable Group:

```plaintext
proc cntselect data=mycas.simulate dist=zip;
  class group;
  model y_p=group x1-x7;
  zeromodel y_p ~ z1-z3;
run;
```

Output 3.1.1 shows the results for the zero-inflated Poisson model.

**Output 3.1.1**  Zero-Inflated Poisson Model with CLASS Statement

![](image)
The “Class Level Information” table shows that the classification variable Group has two levels. The “Model Fit Summary” table shows detailed information about the model and indicates that all 10,000 observations were used to fit the model. All parameter estimates in the “Parameter Estimates” table are highly significant and correspond to their theoretical values set during the data generating process.
References


Chapter 4
The CPANEL Procedure

Contents

Overview: CPANEL Procedure ................................................................. 112
PROC CPANEL Features ................................................................. 113
PROC CPANEL Compared with Other SAS Procedures ......................... 114
Using CAS Sessions and CAS Engine Librefs .................................... 115
Getting Started: CPANEL Procedure .................................................. 116
Syntax: CPANEL Procedure ............................................................... 118
  Functional Summary ................................................................. 118
  PROC CPANEL Statement ........................................................... 119
  BY Statement ................................................................. 120
  CLASS Statement ............................................................. 120
  COMPARE Statement ........................................................... 121
  CORRELATED Statement ......................................................... 123
  ENDOGENOUS Statement ......................................................... 123
  ID Statement ................................................................. 124
  INSTRUMENTS Statement ......................................................... 124
  MODEL Statement ............................................................. 125
  OUTPUT Statement ............................................................ 128
  RESTRICT Statement ........................................................... 129
  TEST Statement ............................................................. 129
Details: CPANEL Procedure ............................................................. 130
  Specifying the Input Data ......................................................... 130
  Specifying the Regression Model .............................................. 131
  Missing Values ................................................................. 132
  Unbalanced Data ............................................................... 132
  Common Notation ............................................................... 132
  Pooled Regression (POOLED) ..................................................... 133
  Between-Groups Regression (BTWNG) ......................................... 133
  One-Way Fixed-Effects Model (FIXONE) ....................................... 133
  One-Way Fixed-Effects Model, First Differencing (FDONE) ............... 134
  Two-Way Fixed-Effects Model (FIXTWO) ..................................... 134
  One-Way Random-Effects Model (RANONE) ................................... 135
  Two-Way Random-Effects Model (RANTWO) .................................. 138
  Hausman-Taylor Estimation (HTAYLOR) ....................................... 142
  Amemiya-MaCurdy Estimation (AMACURDY) .................................. 143
  Instrumental Variables Regression ............................................ 144
  Poolability Test for Fixed Effects .............................................. 147
Overview: CPANEL Procedure

The CPANEL procedure analyzes a class of linear econometric models that arise when time series and cross-sectional data are combined. This type of data is called panel data. Typical examples of panel data include observations over time on people, households, countries, or firms. For example, in the case of survey data on household income, the panel is created by repeatedly surveying the same households over many years. The individual entities that are followed over time are called cross sections.

Regression models for panel data are characterized by an error structure that can be divided into a cross-sectional component, a time component, and an observation-level component. Panel data models can be categorized by the exact structure of the error term and how it interacts with the regression equation. The CPANEL procedure supports the following models:

- one-way and two-way models
- fixed-effects, random-effects, and hybrid models
- instrumental variables (IV) models

A one-way model depends only on the cross section to which the observation belongs. A two-way model depends on both the cross section and the time period to which the observation belongs. The models are called fixed-effects models if the effects are nonrandom and are called random-effects models otherwise.

If the effects are fixed, the models are essentially regression models with dummy variables that correspond to the effects. For fixed-effects models, ordinary least squares (OLS) estimation, with some facility for absorbing the dummy variables, is the best linear unbiased estimator. Random-effects models use a two-stage approach. In the first stage, variance components are estimated by your choice of the methods described by Swamy and Arora (1972); Fuller and Battese (1974); Wansbeek and Kapteyn (1989); Wallace and Hussain (1969); Nerlove (1971). In the second stage, variance components are used to standardize the data, and then OLS regression is performed.

Random-effects models are more efficient than fixed-effects models, and they can estimate effects for variables that do not vary within cross sections. The cost of these added features is that random-effects
models carry much more stringent assumptions than their fixed-effects counterparts. The CPANEL procedure also supports models that blend the desirable features of random and fixed effects. These hybrid models are described by Hausman and Taylor (1981) and Amemiya and MacCurdy (1986).

Instrumental variables (IV) models are models that introduce auxiliary regression variables to deal with endogenous regressors—regressors that are correlated with the residual errors. The CPANEL procedure supports IV generalizations of the pooled regression model, the between-groups regression model, the one-way fixed-effects model, and the one-way random-effects model. For the one-way random-effects model, PROC CPANEL uses methods that were developed by Baltagi (1981).

**PROC CPANEL Features**

The CPANEL procedure fits panel-data regression models by using various techniques, and does the following:

- provides an ID statement for specifying the cross section and time variables
- provides flexible model-building syntax that supports continuous regressors, classification (CLASS) variables, interactions, and nested effects
- fits one-way and two-way fixed-effects models
- fits one-way and two-way random-effects models
- fits Hausman and Taylor (1981) models
- fits Amemiya and MacCurdy (1986) models
- fits instrumental variables regression models
- enables you to fit multiple models by using multiple estimators, all within one procedure call
- provides a COMPARE statement for producing customizable comparisons of various models and estimators
- provides a TEST statement for performing linear tests of hypotheses
- provides a RESTRICT statement for imposing linear parameter constraints
- provides an OUTPUT statement for producing a data table that contains predictions and residuals

Because the CPANEL 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 on CAS
- exploits all the available cores and concurrent threads. For information about how PROC CPANEL uses threads, see the section “Multithreading” on page 49 in Chapter 1, “Shared Concepts.”
PROC CPANEL Compared with Other SAS Procedures

The CPANEL procedure models panel data in a way that is comparable to how the HPPANEL and PANEL procedures in SAS/ETS software do this.

PROC CPANEL Compared with the HPPANEL Procedure

The functionality of the CPANEL procedure closely resembles that of the HPPANEL procedure, which is a high-performance procedure. The CPANEL procedure is the next generation of the HPPANEL procedure, and it was developed specifically for SAS Viya. Both procedures are designed to run on a cluster of machines that distribute the data and the computations. Both procedures perform computations in multiple threads.

The CPANEL provides all the functionality of the HPPANEL procedure, and also provides the following:

- instrumental variable (IV) regression estimators
- Hausman and Taylor (1981) and Amemiya and MaCurdy (1986) estimators, which are hybrids of fixed-effects and random-effects estimators
- estimation of the fixed-effects model by first differencing (FD)
- Swamy and Arora (1972) estimates of variance components
- tables that compare parameters from multiple models and multiple estimation methods
- the option to specify a character variable as the cross section ID variable
- the option to not specify a time series ID variable when time is not relevant to the estimation

PROC CPANEL Compared with the PANEL Procedure

The CPANEL procedure provides the following functionality not found in PROC PANEL:

- instrumental variables (IV) regression estimators
- Swamy and Arora (1972) estimates of variance components
- support for estimation when the data contain cross sections that consist of only one observation
- the option to specify a character variable as the cross section ID variable
- the option to not specify a time-series ID variable when time is not relevant to the estimation

The PANEL procedure parameterizes the fixed-effects model by using the last ordered cross section and the latest time point as reference categories. The CPANEL procedure uses a mean-centered parameterization.

The PANEL procedure supports estimation of dynamic panel models by the generalized method of moments (GMM), the Da Silva (1975) method for serial correlation, and the Parks (1967) method for first-order autocorrelation. The PANEL procedure also provides alternative estimators of variance such as heteroscedastic and autocorrelation-consistent (HAC). Such functionality is not provided in this release of the CPANEL procedure.
The PANEL procedure offers a wide variety of postfitting analyses, such as model specification tests and tests for unit root. The PANEL procedure is also able to construct utility data sets that contain lagged variables and data reformatted from wide form to long form. This release of the CPANEL procedure is limited in postfitting and data-generating functionality, because the focus for large data sets is primarily on model fitting and comparison.

The CPANEL procedure is specifically designed to operate in SAS Viya and performs computations in multiple threads. The PANEL procedure executes in a single thread on a single machine.

**Modeling Only Time Effects**

The PANEL procedure and the HPPANEL procedure provide the following MODEL statement options for estimating models that contain time effects and no cross-sectional effects:

- **FIXONETIME** for one-way fixed time effects
- **BTWNT** for between-groups regression, grouped by time value

You can obtain the same functionality with the CPANEL procedure by setting your time variable as the cross section ID variable, and by leaving the time series ID variable unspecified. The following statements perform estimation equivalent to the FIXONETIME option:

```sas
proc cpanel data = mycas.a;
id time;
model y = x1 x2 x3 / fixone;
run;
```

**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 2 in Chapter 1, “Shared Concepts.”

---

**Getting Started: CPANEL 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 2 and “Loading a SAS Data Set onto a CAS Server” on page 3 in Chapter 1, “Shared Concepts.”

The following DATA step creates the data table `Electricity` from the cost function data in Greene (1990). This DATA step assumes that your CAS engine libref is named `mycas`, but you can substitute any appropriately defined CAS engine libref. The variable `Production` is the log of output in millions of kilowatt-hours, and the variable `Cost` is the log of cost in millions of dollars.

```sas
   data mycas.Electricity;
      input firm year production cost @@;
   datalines;
   1 1955 5.36598 1.14867 1 1960 6.03787 1.45185
   1 1965 6.37673 1.52257 1 1970 6.93245 1.76627
   2 1965 7.40245 2.09519 2 1970 7.82644 2.39480
   3 1955 8.07153 2.94628 3 1960 8.47679 3.25967
   ;
```

Consider the model

$$C_{it} = \beta_0 + \beta_1 P_{it} + v_i + e_{it} \quad \text{for } i = 1, \ldots, N \text{ and } t = 1, \ldots, T$$

where $C_{it}$ represents cost, $P_{it}$ represents production, $v_i$ is the cross-sectional error component, and $e_{it}$ is the error variance component.

If you assume that the cross-sectional effects are random, five possible estimators are available for the variance components. The VCOMP=FB option in the following statements uses the Fuller and Battese (1974) estimator to fit the model:
The code snippet is:

```sas
proc cpanel data = mycas.Electricity;
    id firm year;
    model cost = production / ranone vcomp = fb;
run;
```

The output of these statements is shown in **Output 4.1**.

**Figure 4.1** One-Way Random-Effects Estimation Results

<table>
<thead>
<tr>
<th>Model Description</th>
<th>RanOne</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Set</td>
<td>ELECTRICITY</td>
</tr>
<tr>
<td>Number of Observations</td>
<td>24</td>
</tr>
<tr>
<td>Number of Cross Sections</td>
<td>6</td>
</tr>
<tr>
<td>Time Series Length</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fit Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSE</td>
</tr>
<tr>
<td>MSE</td>
</tr>
<tr>
<td>Root MSE</td>
</tr>
<tr>
<td>R-Square</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variance Component Estimates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
</tr>
<tr>
<td>Cross Sections</td>
</tr>
<tr>
<td>Error</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hausman Test For Random Effects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coefficients</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter Estimates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
</tr>
<tr>
<td>Intercept</td>
</tr>
<tr>
<td>production</td>
</tr>
</tbody>
</table>

Printed first is a report that provides the estimation method and various data counts. Fit statistics and variance components estimates are printed next. A Hausman specification test compares this model to its fixed-effects counterpart. Finally, the table of regression parameter estimates shows the estimates, standard errors, and t tests.
**Syntax: CPANEL Procedure**

The following statements are available in the CPANEL procedure:

```plaintext
PROC CPANEL <options>;
  BY variables;
  CLASS variable <(options)>...<variable <(options)>> </global-options> ;
  COMPARE <model-list> </options> ;
  CORRELATED effects;
  ENDOGENOUS effects;
  ID cross-section-id < time-series-id> ;
  INSTRUMENTS effects;
  MODEL response = <effects> </options> ;
  OUTPUT OUT=CAS-libref.data-table <options> ;
  RESTRICT equation1 < , equation2... > ;
  TEST equation1 < , equation2... > ;
```

**Functional Summary**

Table 4.1 summarizes the statements and options that the CPANEL procedure supports.

<table>
<thead>
<tr>
<th>Description</th>
<th>Statement</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data Set Options</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specifies the input data set</td>
<td>CPANEL</td>
<td>DATA=</td>
</tr>
<tr>
<td>Prevents partitioning of data by cross sections</td>
<td>CPANEL</td>
<td>NOPART</td>
</tr>
<tr>
<td>Specifies the name of an output SAS data table</td>
<td>OUTPUT</td>
<td>OUT=</td>
</tr>
<tr>
<td><strong>Variable Role Statements</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specifies BY-group processing</td>
<td>BY</td>
<td></td>
</tr>
<tr>
<td>Specifies the classification variables</td>
<td>CLASS</td>
<td></td>
</tr>
<tr>
<td>Specifies the cross section and time ID variables</td>
<td>ID</td>
<td></td>
</tr>
<tr>
<td>Declares correlated variables or effects</td>
<td>CORRELATED</td>
<td></td>
</tr>
<tr>
<td>Declares endogenous variables or effects</td>
<td>ENDOGENOUS</td>
<td></td>
</tr>
<tr>
<td>Declares instrumental variables or effects</td>
<td>INSTRUMENTS</td>
<td></td>
</tr>
<tr>
<td><strong>Printing Control Options</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prints correlations of the estimates</td>
<td>MODEL</td>
<td>CORRB</td>
</tr>
<tr>
<td>Prints covariances of the estimates</td>
<td>MODEL</td>
<td>COVB</td>
</tr>
<tr>
<td>Suppress parameter labels from output tables</td>
<td>MODEL</td>
<td>NOLABEL</td>
</tr>
<tr>
<td>Suppresses printed output</td>
<td>CPANEL</td>
<td>NOPRINT</td>
</tr>
<tr>
<td>Prints all available output</td>
<td>CPANEL</td>
<td>PRINTALL</td>
</tr>
<tr>
<td>Prints fixed effects</td>
<td>MODEL</td>
<td>PRINTFIXED</td>
</tr>
</tbody>
</table>
Table 4.1 continued

<table>
<thead>
<tr>
<th>Description</th>
<th>Statement</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performs tests of linear hypotheses</td>
<td>TEST</td>
<td>WALD, LM, LR</td>
</tr>
</tbody>
</table>

**Model Estimation Options**

- Specifies the Amemiya-MacCurdy model: MODEL AMACURDY
- Specifies the between-groups model: MODEL BTWNG
- Specifies the one-way fixed-effects model via first-differencing: MODEL FDONE
- Specifies the one-way fixed-effects model: MODEL FIXONE
- Specifies the two-way fixed-effects model: MODEL FIXTWO
- Specifies the Hausman-Taylor model: MODEL HTAYLOR
- Specifies the between-groups model with instrumental variables: MODEL IVBTWNG
- Specifies the one-way fixed-effects model with instrumental variables: MODEL IVFIXONE
- Specifies the pooled regression model with instrumental variables: MODEL IVPOOLED
- Specifies the one-way random-effects model with instrumental variables: MODEL IVRANONE
- Suppresses the intercept term: MODEL NOINT
- Specifies the pooled regression model: MODEL POOLED
- Specifies the one-way random-effects model: MODEL RANONE
- Specifies the two-way random-effects model: MODEL RANTWO
- Specifies the method for the variance components estimator: MODEL VCOMP=
- Specifies linear equality restrictions on the parameters: RESTRICT

**Model Comparison Statements**

Create tables with side-by-side model comparisons: COMPARE

---

**PROC CPANEL Statement**

`PROC CPANEL <options> ;`

The PROC CPANEL statement invokes the procedure. You can specify the following `options`:

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

names the input data table for PROC CPANEL to use. The default is the most recently created data table. *CAS-libref.data-table* is a two-level name, where
Chapter 4: The CPANEL Procedure

**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 115.

**data-table** specifies the name of the input data table.

**NOPART** prevents repartitioning of the data. Normally, the CPANEL procedure redistributes the data table so that observations for each cross section are not divided among multiple machines. Repartitioning the data makes model computations more efficient but has a cost in memory and computing cycles that are associated with redistribution. Specifying the NOPART option leaves the data distributed as they are, at the expense of less efficient model computations.

**NOPRINT** suppresses all printed output.

**PRINTALL** prints all available output.

In addition, any of the following MODEL statement options can be specified in the PROC CPANEL statement: **CORRB**, **COVB**, **NOLABEL**, and **PRINTFIXED**. When specified in the PROC CPANEL statement, these options apply globally to every MODEL statement. For a complete description of each of these options, see the section “MODEL Statement” on page 125.

---

**BY Statement**

BY *variables*;

You can specify a BY statement in PROC CPANEL 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. Table 4.2 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 4 in Chapter 1, “Shared Concepts.”
Table 4.2  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>
</tbody>
</table>

COMPARE Statement

COMPARE <model-list> <options>;

The COMPARE statement creates tables of side-by-side comparisons of parameter estimates and other model statistics. You can fit multiple models simultaneously by specifying multiple MODEL statements, and you can specify a COMPARE statement to create tables that compare the models.

The COMPARE statement creates two tables: the first table compares model fit statistics such as R-square and mean square error; the second table compares regression coefficients, their standard errors, and (optionally) t-tests.

By default, comparison tables are created for all fitted models, but you can use the optional model-list to limit the comparison to a subset of the fitted models. The model-list consists of a set of model labels, as specified in the MODEL statement; for more information, see the section “MODEL Statement” on page 125. If a model does not have a label, you refer to it generically as “Model i,” where the corresponding model is the ith MODEL statement specified.

You can specify one or more COMPARE statements. The following code illustrates the use of the COMPARE statement:

```
proc cpanel data=mycas.a;
  id csid tsid;
  mod_one: model y = x1 x2 x3 / fixone;
  second: y = x1 x2 / fixone;
  model y = x1 x2 x3 x4 / ranone;
  compare;
  compare "second" "Model 3";
run;
```

The first COMPARE statement compares all three fitted models. The second COMPARE statement uses the generic “Model 3” to identify the third model and compares it to the second model.

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

MSTAT (mstat-list)

specifies a list of model fit statistics to be displayed. A set of statistics is displayed by default, but you can use this option to specify a custom set of model statistics.

The mstat-list can contain one or more of the following keywords:
**ALL** displays all model fit statistics. Not all statistics are appropriate for all models, and thus not always calculated. A blank cell in the table indicates that the statistic is not appropriate for that model.

**DFE** displays the error degrees of freedom. This statistic is displayed by default.

**F** displays the $F$ statistic of the overall test for no fixed effects.

**FNUMDF** displays the numerator degrees of freedom of the overall test for no fixed effects.

**FDENDF** displays the denominator degrees of freedom of the overall test for no fixed effects.

**M** displays the Hausman test $m$ statistic.

**MDF** displays the Hausman test degrees of freedom.

**MSE** displays the model mean square error. This statistic is displayed by default.

**NCS** displays the number of cross sections. This statistic is displayed by default.

**NOBS** displays the number of observations. This statistic is displayed by default.

**NONE** suppresses the table of model fit statistics when specified alone, and is ignored when specified with other options.

**NTS** displays the maximum length of the time series. This statistic is displayed by default.

**PROBF** displays the significance level of the overall test for no fixed effects.

**PROBM** displays the significance level of the Hausman test.

**RMSE** displays the model root mean square error.

**RSQUARE** displays the model R-square fit statistic. This statistic is displayed by default.

**SSE** displays the model sum of squares.

**VARCS** displays the variance component that corresponds to cross sections in random-effects models.

**VARERR** displays the variance component that corresponds to error in random-effects models.

**VARTS** displays the variance component that corresponds to time series in random-effects models.

**OUTPARM=** _CAS-libref.data-table_

names an output data set to contain the data from the comparison table for parameter estimates, standard errors, and $t$ tests. _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 115.

**OUTSTAT=** _CAS-libref.data-table_

names an output data set to contain the data from the comparison table for model fit statistics, such as R-square and mean square error. _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 115.
**PSTAT** *(pstat-list)*
specifies a list of parameter statistics to be displayed. By default, estimated regression coefficients and their standard errors are displayed. Use this option to specify a custom set of parameter statistics.

*pstat-list* can contain one or more of the following keywords:

- **ALL** displays all parameter statistics.
- **ESTIMATE** displays the estimated regression coefficient. This statistic is displayed by default.
- **NONE** suppresses the table of parameter statistics when specified alone, and is ignored when specified with other options.
- **STDERR** displays the standard error. This statistic is displayed by default.
- **PROBT** displays the significance level of the *t* test.
- **T** displays the *t* statistic.

For a demonstration of the COMPARE statement, see Example 4.2.

**CORRELATED Statement**

```
CORRELATED effects;
```

The CORRELATED statement selects the subset of effects from the MODEL statement that are correlated with the individual (or cross-sectional) error term. This statement is valid only when you specify the AMACURDY or HTAYLOR model estimation method in the MODEL statement.

If you specify an *effect* in the CORRELATED statement but not in the MODEL statement, then the *effect* is added to the list of model effects and treated as correlated with the cross-sectional error term. If you have multiple MODEL statements, then each CORRELATED statement is associated with the MODEL statement that immediately precedes it.

The following statements fit a Hausman and Taylor (1981) model with model effects *X1*, *X2*, *Z1*, and *Z2*. The effects *X2* and *Z2* are correlated with the cross-sectional (firm-level) error term.

```plaintext
proc cpanel data=mycas.a;
  id firm year;
  model Y = X1 X2 Z1 Z2 / htaylor;
  correlated X2 Z2;
run;
```

You can also specify classification variables, interactions, and nested effects. For more information about constructing model effects, see the section “Specification and Parameterization of Model Effects” on page 33 in Chapter 1, “Shared Concepts.”

**ENDOGENOUS Statement**

```
ENDOGENOUS effects;
```

The ENDOGENOUS statement selects the subset of effects from the MODEL statement that are endogenous—that is, the effects that are correlated with the observation-level error term. The ENDOGENOUS statement is
valid only when instrumental variables (IV) regression is performed because one or more of the IVBTWNG, IVFIXONE, IVPOOLED, and IVRANONE options are specified in the MODEL statement.

If you specify an effect in the ENDOGENOUS statement but not in the MODEL statement, then the effect is added to the list of model effects and treated as endogenous. If you have multiple MODEL statements, then each ENDOGENOUS statement is associated with the MODEL statement that immediately precedes it.

The ENDOGENOUS statement is used in conjunction with the INSTRUMENTS statement to perform IV regression. The following statements perform IV fixed-effects regression with model effects X1, X2, Z1, and Z2. The effects X2 and Z2 are endogenous, and G1 and G2 are external instruments.

```
proc cpanel data=mycas.a;
  id firm year;
  model Y = X1 X2 Z1 Z2 / ivfixone;
  endogenous X2 Z2;
  instruments G1 G2;
run;
```

You can also specify classification variables, interactions, and nested effects. For more information about constructing model effects, see the section “Specification and Parameterization of Model Effects” on page 33 in Chapter 1, “Shared Concepts.”

### ID Statement

**ID cross-section-id < time-series-id> ;**

The ID statement specifies variables in the input data set that identify the cross section and time period for each observation. Sometimes estimation does not require a time variable, in which case specifying the time-series-id variable is optional.

### INSTRUMENTS Statement

**INSTRUMENTS effects ;**

The INSTRUMENTS statement specifies instrumental variables and effects, and is valid only when instrumental variables (IV) regression is performed because one or more of the IVBTWNG, IVFIXONE, IVPOOLED, and IVRANONE options are specified in the MODEL statement.

The effects you specify in the INSTRUMENTS statement are auxiliary effects that are external to those you specify in the MODEL statement. If you have multiple MODEL statements, then each INSTRUMENTS statement is associated with the MODEL statement that immediately precedes it.

The INSTRUMENTS statement is used in conjunction with the ENDOGENOUS statement to perform IV regression. The following statements perform IV fixed-effects regression with model effects X1, X2, Z1, and Z2. The effects X2 and Z2 are endogenous, and G1 and G2 are external instruments.

```
proc cpanel data=mycas.a;
  id firm year;
  model Y = X1 X2 Z1 Z2 / ivfixone;
  endogenous X2 Z2;
```
You can also specify classification variables, interactions, and nested effects. For more information about constructing model effects, see the section “Specification and Parameterization of Model Effects” on page 33 in Chapter 1, “Shared Concepts.”

MODEL Statement

MODEL response = <effects> < / options > ;

The MODEL statement specifies the regression model, the error structure that is assumed for the regression residuals, and the estimation technique to be used. The response variable (response) on the left side of the equal sign is regressed on the independent variables (effects), which are listed after the equal sign. You can specify any number of MODEL statements. For each MODEL statement, you can specify only one response.

For information about constructing the model effects, see the section “Specification and Parameterization of Model Effects” on page 33 in Chapter 1, “Shared Concepts.”

You can label models. Model labels are used in the printed output to identify the results for different models. If you do not specify a label, the model is referred to by numerical order wherever necessary. You can label models by prefixing the MODEL statement by a label followed by a colon as follows:

label: MODEL ... ;

The MODEL statement supports many options, some more specific than others. Table 4.3 summarizes the options available in the MODEL statement. These are subsequently discussed in detail in the order in which they are presented in the table.

Table 4.3 Summary of MODEL Statement Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimation Technique Options</td>
<td>Fits a one-way model by using the Amemiya-MaCurdy estimator</td>
</tr>
<tr>
<td>AMACURDY</td>
<td>Fits the between-groups model</td>
</tr>
<tr>
<td>BTWNG</td>
<td>Fits a one-way model by using first differences</td>
</tr>
<tr>
<td>FDONE</td>
<td>Fits a one-way fixed-effects model</td>
</tr>
<tr>
<td>FIXONE</td>
<td>Fits a two-way fixed-effects model</td>
</tr>
<tr>
<td>FIXTWO</td>
<td>Fits a one-way model by using the Hausman-Taylor estimator</td>
</tr>
<tr>
<td>HTAYLOR</td>
<td>Fits a between-groups regression model by using instrumental variables</td>
</tr>
<tr>
<td>IVBTWNG</td>
<td>Fits a one-way random-effects model by using instrumental variables</td>
</tr>
<tr>
<td>IVFIXONE</td>
<td>Fits the pooled regression model by using instrumental variables</td>
</tr>
<tr>
<td>IVPOOLED</td>
<td>Fits the pooled regression model by using instrumental variables</td>
</tr>
<tr>
<td>IVRANONE</td>
<td>Fits the pooled regression model by using instrumental variables</td>
</tr>
<tr>
<td>POOLED</td>
<td>Fits the pooled regression model</td>
</tr>
</tbody>
</table>


Table 4.3  continued

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RANONE</td>
<td>Fits a one-way random-effects model</td>
</tr>
<tr>
<td>RANTWO</td>
<td>Fits a two-way random-effects model</td>
</tr>
</tbody>
</table>

**Estimation Control Options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOINT</td>
<td>Suppresses the intercept</td>
</tr>
<tr>
<td>VCOMP=</td>
<td>Specifies the type of variance component estimation for random-effects estimation</td>
</tr>
</tbody>
</table>

**Printed Output Options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CORRB</td>
<td>Prints the parameter correlation matrix</td>
</tr>
<tr>
<td>COVB</td>
<td>Prints the parameter covariance matrix</td>
</tr>
<tr>
<td>NOLABEL</td>
<td>Suppresses variable labels</td>
</tr>
<tr>
<td>PRINTFIXED</td>
<td>Estimates and prints the fixed effects</td>
</tr>
</tbody>
</table>

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

**Estimation Technique Options**

These options specify the assumed error structure and estimation method. You can specify more than one option, in which case the analysis is repeated for each. The default is FIXONE (one-way fixed effects).

All estimation methods are described in detail in the section “Details: CPANEL Procedure” on page 130.

**AMACURDY**
requests Amemiya-MaCurdy estimation for a model that has correlated individual (cross-sectional) effects. You specify the correlated effects by using the CORRELATED statement.

**BTWNG**
estimates a between-groups model.

**FDONE**
estimates a one-way model by using first-differenced methods.

**FIXONE**
estimates a one-way fixed-effects model that corresponds to cross-sectional effects only.

**FIXTWO**
estimates a two-way fixed-effects model.

**HTAYLOR**
requests Hausman-Taylor estimation for a model that has correlated individual (cross-sectional) effects. You specify the correlated effects by using the CORRELATED statement.
**IVBTWNG**
requests instrumental variables between-groups estimation for a model that has endogenous effects. You specify the endogenous effects by using the ENDOGENOUS statement, and you specify external instruments by using the INSTRUMENTS statement.

**IVFIXONE**
requests instrumental variables one-way fixed-effects estimation for a model that has endogenous effects. You specify the endogenous effects by using the ENDOGENOUS statement, and you specify external instruments by using the INSTRUMENTS statement.

**IVPOOLED**
requests instrumental variables pooled regression. You specify the endogenous effects by using the ENDOGENOUS statement, and you specify external instruments by using the INSTRUMENTS statement.

**IVRANONE**
requests instrumental variables one-way random-effects estimation for a model that has endogenous effects. You specify the endogenous effects by using the ENDOGENOUS statement, and you specify external instruments by using the INSTRUMENTS statement.

**POOLED**
estimates a pooled (OLS) model.

**RANONE**
estimates a one-way random-effects model.

**RANTWO**
estimates a two-way random-effects model.

**Estimation Control Options**
These options define parameters that control the estimation and can be specific to the chosen technique (for example, how to estimate variance components in a random-effects model).

**NOINT**
suppresses the intercept parameter from the model.

**VCOMP=FB | NL | SA | WH | WK**
specifies the type of variance component estimate to use. You can specify the following values:

- **FB** uses the Fuller-Battese method.
- **NL** uses the Nerlove method.
- **SA** uses the Swamy-Arora method.
- **WH** uses the Wallace-Hussain method.
- **WK** uses the Wansbeek-Kapteyn method.

By default, VCOMP=SA.
Printed Output Options

These options alter how results are presented.

**CORRB**
prints the matrix of estimated correlations between the parameter estimates.

**COVB**
prints the matrix of estimated covariances between the parameter estimates.

**NOLABEL**
suppresses variable labels from the printed output.

**PRINTFIXED**
estimates and prints the fixed effects in models where they would normally be absorbed within the estimation.

OUTPUT Statement

```
OUTPUT OUT=\textit{CAS-libref.data-table} < \textit{options} > ;
```

The OUTPUT statement creates a data table that contains observationwise statistics that PROC CPANEL computes after fitting the model. In order to avoid data duplication for large data tables, only the identification and dependent variables are included in the output data table unless you specify additional variables in the COPYVAR= option. To avoid further data duplication, the OUTPUT statement is allowed only with MODEL statements that request a single estimation. If you have multiple MODEL statements, then each OUTPUT statement is associated with the MODEL statement that immediately precedes it.

You must specify the following option:

**OUT=** \textit{CAS-libref.data-table}

names the output data table for PROC CPANEL 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 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 115.

- \textit{data-table} specifies the name of the output data table.

You can also specify the following options:

**COPYVAR=** \textit{variable}

**COPYVARS=** \textit{variables}

transfers one or more \textit{variables} from the input data table to the output data table.
RESTRICT Statement

RESTRICT equation <, equation2...> ;

The RESTRICT statement specifies linear equality restrictions on the parameters in the previous MODEL statement. There can be as many unique restrictions as the number of parameters in the preceding MODEL statement. Multiple RESTRICT statements are understood as joint restrictions on a model’s parameters. Restrictions on the intercept are obtained by the use of the INTERCEPT keyword. RESTRICT statements before the first MODEL statement are automatically associated with the first MODEL statement, in addition to any RESTRICT statements that follow it but precede subsequent MODEL statements.

Currently, only linear equality restrictions are permitted in PROC CPANEL. Tests and restriction expressions can be composed of only algebraic operations that involve the addition symbol (+), subtraction symbol (–), and multiplication symbol (*).

The following statements illustrate the use of the RESTRICT statement:

```plaintext
proc cpanel data = mycas.a;
  id csid tsid;
  model y = x1 x2 x3;
  restrict x1 = 0, x2 * .5 + 2 * x3 = 0;
  restrict intercept = 1;
run;
```

Note that a RESTRICT statement cannot include a division sign in its formulation.

TEST Statement

TEST equation <, equation2...> < /options> ;

The TEST statement performs Wald, Lagrange multiplier, and likelihood ratio tests of linear hypotheses about the regression parameters in the preceding MODEL statement. Like RESTRICT statements, TEST statements before the first MODEL statement are automatically associated with the first MODEL statement, in addition to any TEST statements that follow it but precede subsequent MODEL statements. Each equation specifies a linear hypothesis to be tested. All hypotheses in one TEST statement are tested jointly. Variable names in the equations must correspond to regressors in the preceding MODEL statement, and each name represents the coefficient of the corresponding regressor. The INTERCEPT keyword refers to the coefficient of the intercept.

You can specify the following options after a slash (/):
ALL
  specifies Wald, Lagrange multiplier, and likelihood ratio tests.

WALD
  specifies the Wald test.

LM
  specifies the Lagrange multiplier test.

LR
  specifies the likelihood ratio test.

The Wald test is performed by default.

The following statements illustrate the use of the TEST statement:

```
proc cpanel data = mycas.a;
  id csid tsid;
  model y = x1 x2 x3;
  test x1 = 0, x2 * .5 + 2 * x3 = 0 / all;
  test intercept = 0, x3 = 0;
run;
```

The first test investigates the joint hypothesis that $\beta_1 = 0$ and $0.5\beta_2 + 2\beta_3 = 0$.

Currently, only linear equality restrictions and tests are permitted in PROC CPANEL. Tests and restriction expressions can be composed only of algebraic operations that involve the addition symbol (+), subtraction symbol (–), and multiplication symbol (*).

When the TEST statement is used with MODEL statement options AMACURDY, HTAYLOR, IVBTWNG, IVFIXONE, IVPOOLED, and IVRANONE, only the Wald test is available.

---

**Details: CPANEL Procedure**

**Specifying the Input Data**

Panel data are identified by both a cross section identification (ID) variable and a time variable. Suppose that cross sections are identified by the variable State and time periods are identified by the variable Date. You specify the cross section and time series variables in an ID statement. The following statements show the appropriate syntax:

```
proc panel data = mycas.a;
  id State Date;
  model ...;
run;
```

For some estimation strategies (such as one-way fixed effects), time is not relevant to the estimation. In these cases, specifying the time variable is optional. The CPANEL procedure alerts you if do not specify a time variable when one is required.
PROC CPANEL performs its estimations regardless of how the data are partitioned logically across machines, but it might temporarily repartition the data to perform computations more efficiently. You can specify the NOPART option in the PROC CPANEL statement to bypass this repartitioning, at the expense of slower execution.

---

**Specifying the Regression Model**

The CPANEL procedure is similar to other regression procedures in SAS software. Suppose you want to regress the variable $Y$ on regressors $X_1$ and $X_2$. You specify the dependent variable first, followed by an equal sign, followed by the list of regression variables, as shown in the following statements:

```sas
proc panel data mycas.a;
  id state date;
  model y = x1 x2;
run;
```

One advantage of using PROC CPANEL is that you can incorporate a model for the structure of the error terms. It is important to consider what type of model is appropriate for your data and to specify the corresponding option in the MODEL statement. The following model estimation options are supported: POOLED, BTWNG, FIXONE, FDONE, FIXTWO, RANONE, RANTWO, HTAYLOR, AMACURDY, IVPOOLED, IVBTWNG, IVFIXONE, and IVRANONE. The methods that underlie these estimation options are described, in order, beginning with the section “Pooled Regression (POOLED)” on page 133.

The following statements fit a one-way random-effects model with variance components estimated by the Fuller-Battese (FB) method:

```sas
proc panel data mycas.a;
  id State Date;
  model Y = X1 X2 / ranone vcomp = fb;
run;
```

You can specify more than one estimation option in the MODEL statement, and the analysis is repeated for each specified method. You can use multiple MODEL statements to estimate different regression models or to estimate the same model by different methods.

You can also specify classification variables, interactions, and nested effects, as demonstrated by the following statements:

```sas
proc panel data mycas.a;
  id State Date;
  class C1 C2 C3;
  model Y = X1 X2 C1 C2*C1 C3 C1*C3 / ranone vcomp = fb;
run;
```

For more information about constructing model effects, see the section “Specification and Parameterization of Model Effects” on page 33 in Chapter 1, “Shared Concepts.”
Missing Values

Any observation in the input data set that has a missing value for the cross-section ID, time series ID, dependent variable, or any model effect is ignored by PROC CPANEL when it fits the model.

If there are observations in your data where only the dependent variable is missing, you can still compute predicted values for these observations and store them in an output data set by using the OUTPUT statement.

Unbalanced Data

Unbalanced data occur when not all time values are observed for all cross sections or, when time is not part of the estimation, the cross sections are not all the same size.

Whether the data are unbalanced by design or because of missing values, almost all the methods supported by the CPANEL procedure take proper account of the unbalanced data. The one exception is the AMACURDY method, which is suitable only for balanced data.

Common Notation

This section presents notation that is common to all subsequent sections. Consider the panel regression:

\[ y_{it} = \alpha + \sum_{k=1}^{K} x_{itk} \beta_k + u_{it} \quad i = 1, \ldots, N; \quad t = 1, \ldots, T_i \]

The total number of observations is \( M = \sum_{i=1}^{N} T_i \). For balanced data, \( T_i = T \) for all \( i \). For unbalanced data, define \( T \) to be the number of unique time periods.

The exact representation of \( u_{it} \) and the underlying assumptions depend on the estimation method.

In matrix notation the model is

\[ y_{it} = \alpha + x_{it} \beta + u_{it} \]

where \( x_{it} \) is a \( 1 \times K \) row vector of independent variables and \( \beta \) is the \( K \times 1 \) vector of coefficients. Let \( y \) and \( X \) be matrices that are formed by arranging the dependent and independent variables by cross section, and by time within each cross section. Let \( X_\alpha \) be the \( X \) matrix augmented by a first column of ones, which corresponds to the intercept term \( \alpha \).

Define the following utility matrices:

- \( I_p \) is an identity matrix of dimension \( p \).
- \( j_p \) is a \( p \times 1 \) column vector of ones.
- \( J_p = j_p j_p' \) is a matrix of ones of dimension \( p \).
- \( \bar{J}_p = p^{-1} J_p \).
Pooled Regression (POOLED)

You perform pooled regression by specifying the POOLED option in the MODEL statement. Pooled regression is standard ordinary least squares (OLS) regression without any cross-sectional or time effects. The error structure is simply \( u_{it} = e_{it} \), where the \( e_{it} \) are independently and identically distributed (iid) with zero mean and variance \( \sigma_e^2 \).

In the following sections, the panel data are assumed to be unbalanced unless otherwise indicated. If the data are balanced, the formulas reduce appropriately.

Between-Groups Regression (BTWNG)

You perform between-groups regression by specifying the BTWNG option in the MODEL statement. When the data are balanced, between-groups regression is ordinary least squares (OLS) regression performed on data that have been collapsed into cross-sectional means. When the data are unbalanced, the regression is weighted to assign more importance to larger cross sections. The cross-sectional means are multiplied by the weighting factor \( w_i = \sqrt{T_i/T} \), where \( T = M/N \) is the average cross section size.

One-Way Fixed-Effects Model (FIXONE)

You perform one-way fixed-effects estimation by specifying the FIXONE option in the MODEL statement. The error structure for the one-way fixed-effects model is

\[ u_{it} = v_i + e_{it} \]

where the \( v_i \) are nonrandom parameters that are restricted to sum to 0, and the \( e_{it} \) are iid with zero mean and variance \( \sigma_e^2 \).

The fixed-effects model can be estimated by ordinary least squared (OLS), treating the \( v_i \) as coefficients on dummy variables that identify the cross sections. However, when \( N \) is large, you might want to estimate only \( \beta \) and not \( v_i \).

Let \( Q_0 = \text{diag}(E_{T_i}) \). The matrix \( Q_0 \) represents the within transformation, the conversion of the raw data to deviations from a cross section’s mean. Let \( X_w = Q_0X \) and \( y_w = Q_0y \). The within estimator of \( \beta \) is

\[ \hat{\beta}_w = (X_w'X_w)^{-1}X_w'y_w \]

The previous estimation does not involve the intercept term because \( \hat{\beta}_w \) is the same whether or not the intercept \( \alpha \) is included in the model.

Standard errors, \( t \) statistics, and fit statistics such as mean square error (MSE) are all equivalent to those obtained from OLS regression of \( y_w \) on \( X_w \). The only exception is the error degrees of freedom, which equals \( M - N - K \) to account for the tacit estimation of the \( N \) fixed effects.
The estimate of the intercept is
\[ \hat{\alpha} = \bar{y}_{..} - \bar{x}_{..} \hat{\beta}_w \]
where \( \bar{y}_{..} \) is the overall mean of \( y_{it} \) and \( \bar{x}_{..} \) is the overall mean of \( x_{it} \).

Fixed effects are not estimated as part of the regression, but they can be obtained by specifying the PRINTFIXED option in the MODEL statement. Each fixed effect is estimated as
\[ \hat{\nu}_i = \bar{y}_i - \bar{x}_i \hat{\beta}_w \]
where \( \bar{y}_i \) and \( \bar{x}_i \) are cross-sectional means.

Variance estimates of \( \hat{\alpha} \) and \( \hat{\nu}_i \) are obtained by the delta method. If you specify the NOINT option, then the \( \hat{\nu}_i \) are shifted to absorb \( \hat{\alpha} \).

One-Way Fixed-Effects Model, First Differencing (FDONE)

You perform one-way fixed-effects estimation via first differencing by specifying the FDONE option in the MODEL statement. The method of first differencing offers an alternative to the within estimator \( \hat{\beta}_w \).

Consider the following one-way fixed-effects model:
\[ y_{it} = \alpha + x_{it} \beta + v_i + e_{it} \]

For this model, the fixed effects are removed by subtracting first-order lags from both sides of the equation:
\[ y_{it} - y_{i, t-1} = (x_{it} - x_{i, t-1}) \beta + (e_{it} - e_{i, t-1}) \]

Define \( \Delta y_{it} = y_{it} - y_{i, t-1} \) and \( \Delta x_{it} = x_{it} - x_{i, t-1} \), for \( i = 1, \ldots, N \) and \( t = 2, \ldots, T_i \). You obtain the first-differenced estimator, \( \hat{\beta}_d \), and its variance by performing OLS regression of \( \Delta y_{it} \) on \( \Delta x_{it} \).

You cannot obtain the error variance directly from the OLS regression because that regression is based on the differenced residual \( e_{it} - e_{i, t-1} \). Instead, you can estimate \( \sigma^2_e \) by \( \hat{\sigma}^2_e = \text{SSE}/(M - N - K) \), where
\[ \text{SSE} = \sum_{i=1}^{N} \sum_{t=1}^{T_i} (y_{it} - x_{it} \hat{\beta}_d - \hat{v}_i)^2 \]

and \( \hat{\nu}_i = \bar{y}_i - \bar{x}_i \hat{\beta}_d \).

Estimation and inference for \( \alpha \) and \( v_i \) are identical to that described in the section “One-Way Fixed-Effects Model (FIXONE)” on page 133, with \( \hat{\beta}_w \) replaced by \( \hat{\beta}_d \).

Two-Way Fixed-Effects Model (FIXTWO)

You perform two-way fixed-effects estimation by specifying the FIXTWO option in the MODEL statement. The error specification for the two-way fixed-effects model is
\[ u_{it} = v_i + \lambda_t + e_{it} \]
where the \( v_i \) and \( \lambda_t \) are nonrandom parameters to be estimated. Assume the identifiability restriction \( \sum_t \lambda_t = 0 \).

Estimation is similar to that for one-way fixed effects, for which a within transformation is used to convert the problem to OLS regression. For two-way models under the general case of unbalanced data, the within transformation is more complex.

Following Wansbeek and Kapteyn (1989) and Baltagi (2013, sec. 9.4), let \( X^* \) and \( y^* \) be versions of \( X \) and \( y \) whose rows are sorted by time period, and by cross section within each time period. With the data sorted in this manner, define \( D_N \) to be the \( M \times N \) design matrix for cross sections. Each row of \( D_N \) contains a 1 in the column that corresponds to that observation’s cross section, and 0s in the remaining columns. Similarly, define \( D_T \) to be the \( M \times T \) design matrix for time periods. In balanced data, \( D_N D_T = I_N \otimes I_T \).

Define the following:

\[
\begin{align*}
\Delta_N &= \mathbf{D}_N \mathbf{D}_N \quad (N \times N) \\
\Delta_T &= \mathbf{D}_T \mathbf{D}_T \quad (T \times T) \\
\mathbf{A} &= \mathbf{D}_T \mathbf{D}_N \quad (T \times N) \\
\bar{\mathbf{D}} &= \mathbf{D}_T - \mathbf{D}_N \Delta_N^{-1} \mathbf{A} \quad (M \times T) \\
\mathbf{Q} &= \Delta_T - \mathbf{A} \Delta_N^{-1} \mathbf{A} \quad (T \times T) \\
\mathbf{P} &= \mathbf{I}_M - \mathbf{D}_N \Delta_N^{-1} \mathbf{D}_N - \bar{\mathbf{D}} \mathbf{Q}^{-1} \bar{\mathbf{D}} \quad (M \times M)
\end{align*}
\]

The matrix \( \mathbf{P} \) provides the two-way within transformation. If the data are balanced, this amounts to transforming any data value \( z_{it} \) to \( z_{it} - \bar{z}_i - \bar{z}_t + \bar{z}_{..} \).

Applying the two-way within transformation means that you can use OLS regression of \( \mathbf{P} y^* \) on \( \mathbf{P} X^* \) to obtain \( \hat{\beta}_f \), \( \text{Var}(\hat{\beta}_f) \), and fit statistics such as mean-square error (MSE), provided that you adjust the error degrees of freedom to equal \( M - N - T - K + 1 \).

Define the residual vector \( r^* = y^* - X^* \hat{\beta}_f \). Estimates of the time effects are \( \hat{\lambda} = Q^{-1} \bar{D} r^* \), and estimates of the cross-sectional effects are \( \hat{\nu} = (\Theta_1 + \Theta_2 - \Theta_3) r^* \), where

\[
\begin{align*}
\Theta_1 &= \Delta_N^{-1} \bar{\mathbf{D}}_N \\
\Theta_2 &= \Delta_N^{-1} \mathbf{A} \mathbf{Q}^{-1} \bar{\mathbf{D}} \mathbf{T} \\
\Theta_3 &= \Delta_N^{-1} \mathbf{A} \mathbf{Q}^{-1} \mathbf{A} \Delta_N^{-1} \mathbf{D}_N
\end{align*}
\]

If the model includes an intercept, then \( \hat{\alpha} = \bar{y} - \bar{x} \cdot \hat{\beta}_f \), and the \( \hat{v}_i \) are shifted to exclude \( \hat{\alpha} \).

Variance and covariance estimates for \( \hat{\alpha}, \hat{\nu}, \) and \( \hat{\lambda} \) are obtained by the delta method, because each of these quantities is a linear transformation of \( y^* \) and \( \hat{\beta}_f \).

---

**One-Way Random-Effects Model (RANONE)**

You perform one-way random-effects estimation by specifying the RANONE option in the MODEL statement. The specification for the one-way random-effects model is

\[ u_{it} = v_i + e_{it} \]
where the $v_{it}$ are iid with zero mean and variance $\sigma_v^2$, and the $e_{it}$ are iid with zero mean and variance $\sigma_e^2$. Furthermore, a random-effects specification assumes that the error terms are mutually uncorrelated and that each error term is uncorrelated with $X$.

Estimation proceeds in two steps. First, you obtain estimates of the variance components $\sigma_v^2$ and $\sigma_e^2$. The CPANEL procedure provides five methods for estimating variance components; these methods are described in the following subsections.

Second, with the variance components in hand, you form a weight for each cross section,

$$\hat{\theta}_i = 1 - \hat{\sigma}_e / \hat{w}_i$$

where $\hat{w}_i = T_i \hat{\sigma}_v^2 + \hat{\sigma}_e^2$. Taking $\hat{\theta}_i$, you form the partial deviations:

$$\tilde{y}_{it} = y_{it} - \hat{\theta}_i \tilde{y}_i,$$

$$\tilde{x}_{\alpha, it} = x_{\alpha, it} - \hat{\theta}_i \tilde{x}_{\alpha, i}.$$

The random-effects estimation is then the result of OLS regression on the transformed data.

The CPANEL procedure provides five options for estimating variance components, as described in the following subsections.

**Swamy-Arora Method**

The Swamy and Arora (1972) method is the default method for estimating variance components; it can also be specified as the VCOMP=SA option in the MODEL statement. The Swamy-Arora method is part of a class of methods known as analysis of variance (ANOVA) estimators.

ANOVA estimators obtain variance components by solving a system of equations that is based on expected sums of squares. The following quadratic forms correspond to the within and between sums of squares, respectively:

$$q_e = u' Q_0 u$$

$$q_v = u' P_0 u$$

In these equations, $Q_0 = \text{diag}(E_{T_i} T_i)$, $P_0 = \text{diag}(\bar{J}_{T_i} T_i)$, and $u$ is the vector of true residuals.

The ANOVA methods differ only in how they estimate $u$. The Swamy-Arora method substitutes into $q_e$ the within residuals from one-way fixed-effects estimation and substitutes into $q_v$ the residuals from between-groups regression.

The expected values of the quadratic forms are

$$E \left( \hat{u}_w' Q_0 \hat{u}_w \right) = (M - N - K) \sigma_v^2$$

$$E \left( \hat{u}_b' P_0 \hat{u}_b \right) = \left[ M - \text{tr} \left\{ (X'_{\alpha} P_0 X_{\alpha})^{-1} X'_{\alpha} Z_0 Z_o X_{\alpha} \right\} \right] \sigma_v^2 + (N - K - 1) \sigma_e^2$$

where $Z_0 = \text{diag}(J_{T_i})$.

Estimates of $\sigma_v^2$ and $\sigma_e^2$ are obtained by first setting the expected sums of squares to their observed values, and then solving.
Wallace-Hussain Method

You can use the Wallace and Hussain (1969) method for estimating variance components by specifying the VCOMP=WH option in the MODEL statement. The Wallace-Hussain method is an ANOVA method that uses the residuals from pooled (OLS) regression, \( \hat{u}_p \), in both quadratic forms.

The expected values of the quadratic forms are

\[
E \left( \hat{u}_p' Q_0 \hat{u}_p \right) = (d_1 - d_3) \sigma^2_v + (M - N - K - 1 + d_2) \sigma^2_e
\]

\[
E \left( \hat{u}_p' P_0 \hat{u}_p \right) = (M - 2d_1 + d_3) \sigma^2_v + (N - d_2) \sigma^2_e
\]

where

\[
d_1 = \text{tr} \left\{ (X'_\alpha X_\alpha)^{-1} X'_\alpha Z_0 Z'_0 X_\alpha \right\}
\]

\[
d_2 = \text{tr} \left\{ (X'_\alpha X_\alpha)^{-1} X'_\alpha P_0 X_\alpha \right\}
\]

\[
d_3 = \text{tr} \left\{ (X'_\alpha X_\alpha)^{-1} X'_\alpha P_0 X_\alpha (X'_\alpha X_\alpha)^{-1} X'_\alpha Z_0 Z'_0 X_\alpha \right\}
\]

Wansbeek-Kapteyn Method

You can use the Wansbeek-Kapteyn method for estimating variance components by specifying the VCOMP=WK option in the MODEL statement. The method is a specialization (Baltagi and Chang 1994) of the approach used by Wansbeek and Kapteyn (1989) for unbalanced two-way models. The method was also suggested by Amemiya (1971) for balanced data.

The Wansbeek and Kapteyn method is an ANOVA method that uses the within residuals from one-way fixed effects, \( \hat{u}_w \), in both quadratic forms.

The expected values of the quadratic forms are

\[
E \left( \hat{u}_w' Q_0 \hat{u}_w \right) = (M - N - K) \sigma^2_v
\]

\[
E \left( \hat{u}_w' P_0 \hat{u}_w \right) = (N - 1 + d) \sigma^2_v + \left( M - M^{-1} \sum_{i=1}^{N} T_i^2 \right) \sigma^2_e
\]

where

\[
d = \text{tr} \left\{ (X'_Q X)^{-1} X'_P X \right\} - \text{tr} \left\{ (X'_Q X)^{-1} X' \tilde{J}_M X \right\}
\]

Fuller-Battese Method

You can use the Fuller and Battese (1974) method for estimating variance components by specifying the VCOMP=FB option in the MODEL statement. Following Baltagi (2013, sec. 9.2), obtain \( \hat{\sigma}_e^2 \) as the mean square error (MSE) from one-way fixed effects. The cross-sectional variance is

\[
\hat{\sigma}_v^2 = \frac{R(\nu|\beta) - (N - 1) \hat{\sigma}_e^2}{M - \text{tr} \{ Z_0 X_\alpha (X'_\alpha X_\alpha)^{-1} X'_\alpha Z_0 \}}
\]
where
\[ R(v|\beta) = R(\beta|v) + R(v) - R(\beta) \]
for
\[ R(v) = y^T Z_0 (Z_0^T Z_0)^{-1} Z_0^T y \]
\[ R(\beta|v) = y_w^T X_w' (X_w' X_w)^{-1} X_w y_w \]
\[ R(\beta) = y^T X^T (X^T X)^{-1} X y \]

**Nerlove Method**

You can use the Nerlove (1971) method for estimating variance components by specifying the VCOMP=NL option in the MODEL statement. Nerlove’s method provides a simple alternative to previous estimation strategies. You estimate \( \sigma^2_v \) as the sample variance of the cross-sectional effects, estimated from a one-way fixed-effects regression. Specifically, \( \hat{\sigma}^2_v = \frac{1}{N-1} \sum_{i=1}^{N} (\hat{v}_i - \bar{v})^2 \), where \( \bar{v} \) is the mean of the estimated fixed effects. You estimate \( \sigma^2_e \) by taking the error sum of squares from one-way fixed-effects regression and then dividing by \( M \).

**Selecting the Appropriate Variance Component Method**

Baltagi and Chang (1994) conducted an extensive simulation study regarding the finite-sample properties of the variance estimators that are supported by the CPANEL procedure. The choice of method has little bearing on estimates of regression coefficients, their standard errors, and estimation of the error variance \( \sigma^2_e \). If your goal is inference on \( \hat{\beta} \), then the variance-component method will matter little.

The methods have varying performance in how they estimate \( \sigma^2_v \), the cross-sectional variance. All five methods tend to perform poorly if either the data are severely unbalanced or the ratio \( \sigma^2_v / \sigma^2_e \) is much greater than 1.

The Nerlove method is the only method that guarantees a nonnegative estimate of \( \sigma^2_v \); the other four methods reset a negative estimate to 0. However, the Nerlove method is particularly unsuitable for unbalanced data because the sample variance it computes is not weighted by \( T_i \).

**Two-Way Random-Effects Model (RANTWO)**

You perform two-way random-effects estimation by specifying the RANTWO option in the MODEL statement. The specification for the two-way random-effects model is
\[ u_{it} = v_i + \lambda_t + e_{it} \]
where the \( v_i \) are iid with zero mean and variance \( \sigma^2_v \), the \( \lambda_t \) are iid with zero mean and variance \( \sigma^2_\lambda \), and the \( e_{it} \) are iid with zero mean and variance \( \sigma^2_e \). Furthermore, a random-effects specification assumes that the error terms are mutually uncorrelated and that each error term is uncorrelated with \( X \).

Estimation proceeds in two steps. First, you obtain estimates of the variance components \( \sigma^2_v, \sigma^2_\lambda, \) and \( \sigma^2_e \). The CPANEL procedure provides five methods for estimating variance components; these methods are described in the following subsections.
Second, with the variance-component estimates in hand, you transform the data in such a way that estimation can take place using ordinary least squares (OLS). In two-way models with unbalanced data, the transformation is quite complex. Throughout this section, \(y\) and \(X\) are treated as being sorted first by time, and then by cross-section within time. For the definitions of the design matrices \(D_N\) and \(D_T\), see the section “Two-Way Fixed-Effects Model (FIXTWO)” on page 134. The variance of \(y\) is

\[
\Omega = \sigma^2_y I_M + \sigma^2_v D_N D_N' + \sigma^2_{\lambda} D_T D_T'
\]

and estimation proceeds as OLS regression of \(\hat{\sigma}_e \hat{\Omega}^{-1/2} y\) on \(\hat{\sigma}_e \hat{\Omega}^{-1/2} X\).

Rather than invert the \(M \times M\) matrix \(\hat{\Omega}\) directly, Wansbeek and Kapteyn (1989) provide the more convenient form

\[
\hat{\sigma}_e^2 \hat{\Omega}^{-1} = V - VD_T \tilde{P}^{-1} D_T' V
\]

where

\[
V = I_M - D_N \tilde{\Delta}^{-1} D_N', \\
\tilde{P} = \tilde{\Delta}_T - D_T' D_N \tilde{\Delta}_N^{-1} D_N D_T
\]

with \(\tilde{\Delta}_N = D_N' D_N + (\hat{\sigma}_v^2 / \hat{\sigma}_e^2) I_N\) and \(\tilde{\Delta}_T = D_T' D_T + (\hat{\sigma}_e^2 / \hat{\sigma}_\lambda^2) I_T\).

If the data are balanced, then the calculations are simplified considerably—the data are transformed from \(z_{it} - \hat{\theta}_1 \hat{z}_{i} - \hat{\theta}_2 \hat{z}_{,i} + \hat{\theta}_3 \hat{z}_{,,}\), where

\[
\hat{\theta}_1 = 1 - \hat{\sigma}_e (T \hat{\sigma}_v^2 + \hat{\sigma}_e^2)^{-1/2} \\
\hat{\theta}_2 = 1 - \hat{\sigma}_e (N \hat{\sigma}_\lambda^2 + \hat{\sigma}_e^2)^{-1/2} \\
\hat{\theta}_3 = \hat{\theta}_1 + \hat{\theta}_2 + \hat{\sigma}_e (T \hat{\sigma}_v^2 + N \hat{\sigma}_\lambda^2 + \hat{\sigma}_e^2)^{-1/2} - 1
\]

The CPANEL procedure provides five options for estimating variance components, as described in the following subsections.

**Swamy-Arora Method**

The Swamy and Arora (1972) method is the default method for estimating variance components; it can also be specified as the VCOMP=SA option in the MODEL statement. The Swamy-Arora method is part of a class of methods known as analysis of variance (ANOVA) estimators.

ANOVA estimators obtain variance components by solving a system of equations that is based on expected sums of squares. The following quadratic forms correspond to the two-way within sum of squares, the sum of squares between time periods, and the sum of squares between cross sections, respectively:

\[
q_e = u' P u \\
q_\lambda = u' D_T \Delta_T^{-1} D_T' u \\
q_v = u' D_N \Delta_N^{-1} D_N' u
\]

The matrix \(P\) is the two-way within transformation defined in the section “Two-Way Fixed-Effects Model (FIXTWO)” on page 134, \(\Delta_T = D_T' D_T\), \(\Delta_N = D_N' D_N\), and \(u\) is the vector of true residuals.
The ANOVA methods differ only in how they estimate \( u \). The Swamy-Arora method substitutes into \( q_e \) the within residuals from two-way fixed-effects estimation, substitutes into \( q_\lambda \) the residuals from between-time-periods regression, and substitutes into \( q_v \) the residuals from between-cross-sections regression.

The expected values of the quadratic forms are

\[
E \left( \hat{u}_j \hat{P} \hat{u}_f \right) = (M - N - T - K + 1)\sigma_e^2
\]

\[
E \left( \hat{u}_\lambda \hat{P} \lambda \hat{u}_\lambda \right) = \left[ M - \text{tr} \left( (X'_\alpha P_\lambda X_\alpha)^{-1} X'_\alpha D_T D'_T X_\alpha \right) \right] \sigma_\lambda^2 + (T - K - 1)\sigma_e^2
\]

\[
E \left( \hat{u}_v \hat{P} v \hat{u}_v \right) = \left[ M - \text{tr} \left( (X'_\alpha P_v X_\alpha)^{-1} X'_\alpha D_N D'_N X_\alpha \right) \right] \sigma_v^2 + (N - K - 1)\sigma_e^2
\]

where \( P_\lambda = D_T \Delta_T^{-1} D'_T \) and \( P_v = D_N \Delta_N^{-1} D'_N \).

Estimates of \( \sigma_v^2, \sigma_\lambda^2 \), and \( \sigma_e^2 \) are obtained by setting the expected sums of squares to their observed values and solving.

**Wallace-Hussain Method**

You can use the Wallace and Hussain (1969) method for estimating variance components by specifying the VCOMP=WH option in the MODEL statement. The Wallace-Hussain method is an ANOVA method that uses the residuals from pooled (OLS) regression, \( \hat{u}_p \), in all three quadratic forms.

The expected values of the quadratic forms are

\[
E \left( \hat{u}_p \hat{P} p \hat{u}_p \right) = d_{11} \sigma_e^2 + d_{12} \sigma_v^2 + d_{13} \sigma_\lambda^2
\]

\[
E \left( \hat{u}_p \hat{P} \lambda \hat{u}_p \right) = d_{21} \sigma_e^2 + d_{22} \sigma_v^2 + d_{23} \sigma_\lambda^2
\]

\[
E \left( \hat{u}_p \hat{P} v \hat{u}_p \right) = d_{31} \sigma_e^2 + d_{32} \sigma_v^2 + d_{33} \sigma_\lambda^2
\]

Define \( \Sigma = (X'_\alpha X_\alpha)^{-1} \), which is the inverse crossproducts matrix from pooled regression. Also define \( S_v = X'_\alpha D_N D'_N X_\alpha \) and \( S_\lambda = X'_\alpha D_T D'_T X_\alpha \), which are the individual-level sum of squares and the time-level sum of squares, respectively. The coefficients are

\[
d_{11} = M - N - T + 1 - \text{tr} \left( X'_\alpha P X_\alpha \Sigma \right)
\]

\[
d_{12} = \text{tr} \left( S_v \Sigma X'_\alpha P X_\alpha \Sigma \right)
\]

\[
d_{13} = \text{tr} \left( S_\lambda \Sigma X'_\alpha P X_\alpha \Sigma \right)
\]

\[
d_{21} = T - \text{tr} \left( X'_\alpha P_\lambda X_\alpha \Sigma \right)
\]

\[
d_{22} = T - 2\text{tr} \left( X'_\alpha P_\lambda D_N D'_N X_\alpha \Sigma \right) + \text{tr} \left( X'_\alpha P_\lambda X_\alpha \Sigma S_v \Sigma \right)
\]

\[
d_{23} = M - 2\text{tr} \left( S_\lambda \Sigma \right) + \text{tr} \left( X'_\alpha P_\lambda X_\alpha \Sigma S_\lambda \Sigma \right)
\]

\[
d_{31} = N - \text{tr} \left( X'_\alpha P_v X_\alpha \Sigma \right)
\]

\[
d_{32} = M - 2\text{tr} \left( S_v \Sigma \right) + \text{tr} \left( X'_\alpha P_v X_\alpha \Sigma S_v \Sigma \right)
\]

\[
d_{33} = N - 2\text{tr} \left( X'_\alpha P_v D_T D'_T X_\alpha \Sigma \right) + \text{tr} \left( X'_\alpha P_v X_\alpha \Sigma S_\lambda \Sigma \right)
\]
Wansbeek-Kapteyn Method

You can use the Wansbeek-Kapteyn method for estimating variance components by specifying the VCOMP=WK option in the MODEL statement. The method is a specialization (Baltagi and Chang 1994) of the approach used by Wansbeek and Kapteyn (1989) for unbalanced two-way models.

The Wansbeek and Kapteyn method is an ANOVA method that uses the within residuals from two-way fixed effects, $\hat{u}_f$, in all three quadratic forms.

The expected values of the quadratic forms are

$$E \left( \hat{u}_f' P_{\hat{u}_f} \right) = (M - N - T - K + 1)\sigma_e^2$$
$$E \left( \hat{u}_f' P_{\hat{u}_f} \right) = (T + k_N - k_0) \sigma_e^2 + (T - \delta_N)\sigma_v^2 + (M - \delta_T)\sigma_\lambda^2$$
$$E \left( \hat{u}_f' P_v \hat{u}_f \right) = (N + k_T - k_0) \sigma_e^2 + (N - \delta_N)\sigma_v^2 + (N - \delta_T)\sigma_\lambda^2$$

where $\delta_N = M^{-1} \sum_{i=1}^{N} T_i^2$ and $\delta_T = M^{-1} \sum_{t=1}^{T} N_t^2$. The other constants are defined by

$$k_0 = 1 + M^{-1} j_M X (X' P X)^{-1} X' j_M$$
$$k_N = \text{tr} \{ (X' P X)^{-1} X' P_{\lambda} X \}$$
$$k_T = \text{tr} \{ (X' P X)^{-1} X' P_v X \}$$

When the NOINT option is specified, the variance-component equations change slightly: $k_0$, $\delta_N$, and $\delta_T$ are all replaced by 0.

Fuller-Battese Method

You can use the Fuller and Battese (1974) method for estimating variance components by specifying the VCOMP=FB option in the MODEL statement. Following the discussion in Baltagi, Song, and Jung (2002), the Fuller-Battese method is a variation of the three ANOVA methods discussed previously.

The quadratic form, $q_e$, is the same as in the previous methods, and $u$ is estimated by the two-way within residuals $\hat{u}_f$. The other two quadratic forms, $q_\lambda$ and $q_v$, are replaced by the error sums of squares from one-way fixed-effects estimations.

The resulting system of equations is

$$E \left( \hat{u}_f' P_{\hat{u}_f} \right) = (M - N - T - K + 1)\sigma_e^2$$
$$E \left( \hat{u}_f' P_{\hat{u}_f} \right) = (M - T - K)\sigma_e^2 + \left[ M - T - \text{tr} \left\{ X' W_{\lambda} D_N D' N W_{\lambda} X (X' W_{\lambda} X)^{-1} \right\} \right] \sigma_v^2$$
$$E \left( \hat{u}_f' P_v \hat{u}_f \right) = (M - N - K)\sigma_e^2 + \left[ M - N - \text{tr} \left\{ X' W_v D_T D' T W_v X (X' W_v X)^{-1} \right\} \right] \sigma_\lambda^2$$

where $W_{\lambda} = I_M - P_{\lambda}$, $W_v = I_M - P_v$, $\hat{u}_\lambda$ are the residuals from a one-way model with time fixed effects, and $\hat{u}_v$ are the residuals from a one-way model with individual fixed effects.
Nerlove Method

You can use the Nerlove (1971) method for estimating variance components by specifying the VCOMP=NL option in the MODEL statement.

Begin by fitting a two-way fixed-effects model. The estimator of the error variance is

\[ \hat{\sigma}_e^2 = M^{-1} \hat{u}_f' \hat{P} \hat{u}_f \]

You obtain \( \hat{\sigma}_e^2 \) as the sample variance of the \( N \) estimated individual effects, and \( \hat{\sigma}_\lambda^2 \) as the sample variance of the \( T \) estimated time effects.

Hausman-Taylor Estimation (HTAYLOR)

You perform Hausman-Taylor estimation by specifying the HTAYLOR option in the MODEL statement. The Hausman and Taylor (1981) model is a hybrid that combines the consistency of a fixed-effects model with the efficiency and applicability of a random-effects model. One-way random-effects models assume exogeneity of the regressors, namely that they are independent of both the cross-sectional and observation-level errors. When some regressors are correlated with the cross-sectional errors, the random effects model can be adjusted to deal with this form of endogeneity.

Consider the one-way model:

\[ y_{it} = x_{1it} \beta_1 + x_{2it} \beta_2 + z_{1i} \gamma_1 + z_{2i} \gamma_2 + v_i + e_{it} \]

The regressors are subdivided so that \( x_{1it} \) and \( x_{2it} \) vary within cross sections whereas \( z_{1i} \) and \( z_{2i} \) do not and would otherwise be dropped from a fixed-effects model. The subscript 1 denotes variables that are independent of both error terms (exogenous variables), and the subscript 2 denotes variables that are independent of the observation-level errors \( e_{it} \) but correlated with cross-sectional errors \( v_i \). The intercept term (if your model has one) is included as part of \( z_{1i} \).

The Hausman-Taylor estimator is a two-stage least squares (2SLS) regression on data that are weighted similarly to data for random-effects estimation. The weights are functions of the estimated variance components.

The observation-level variance is estimated from a one-way fixed-effects model fit. Obtain \( y_w, X_w, \) and \( \hat{\beta}_w \) from the section “One-Way Fixed-Effects Model (FIXONE)” on page 133. Then

\[ \hat{\sigma}_e^2 = \frac{\text{SSE}}{(M - N)} \]

where

\[ \text{SSE} = (y_w - X_w \hat{\beta}_w)'(y_w - X_w \hat{\beta}_w) \]

To estimate the cross-sectional error variance, form the mean-residual vector \( r = P_0'(y - X_w \hat{\beta}_w) \), where \( P_0 = \text{diag}(J_{T_i}) \). You can use the mean residuals to obtain intermediate estimates of the coefficients for \( z_1 \) and \( z_2 \) via two-stage least squares (2SLS) estimation. At the first stage, use \( x_1 \) and \( z_1 \) as instrumental variables to predict \( z_2 \). At the second stage, regress \( r \) on both \( z_1 \) and the predicted \( z_2 \) to obtain \( \hat{y}_1^m \) and \( \hat{y}_2^m \).

To estimate the cross-sectional variance, compute

\[ \hat{\sigma}_\lambda^2 = \{R(v)/N - \hat{\sigma}_e^2\}/\hat{T}, \quad \text{where} \quad \hat{T} = N/(\sum_{i=1}^N T_i^{-1}) \]

and

\[ R(v) = (r - Z_1 \hat{y}_1^m - Z_2 \hat{y}_2^m)'(r - Z_1 \hat{y}_1^m - Z_2 \hat{y}_2^m) \]

The design matrices \( Z_1 \) and \( Z_2 \) are formed by stacking the data observations of \( z_{1i} \) and \( z_{2i} \), respectively.
After variance-component estimation, transform the dependent variable into partial deviations: \( y_{it}^* = y_{it} - \hat{\theta}_i \tilde{y}_i \). Likewise, transform the regressors to form \( x_{1it}^*, x_{2it}^*, z_{1i}^*, \) and \( z_{2i}^* \). The partial weights \( \hat{\theta}_i \) are determined by \( \hat{\theta}_i = 1 - \hat{\delta}_e / \hat{w}_i \), with \( \hat{w}_i^2 = T_i \hat{\sigma}_e^2 + \hat{\sigma}_c^2 \).

Finally, you obtain the Hausman-Taylor estimates by performing 2SLS regression of \( y_{it}^* \) on \( x_{1it}^*, x_{2it}^*, z_{1i}^* \), and \( z_{2i}^* \). For the first-stage regression, use the following instruments:

- \( \tilde{x}_{it} \), the deviations from cross-sectional means for all time-varying variables (correlated and uncorrelated) for the \( i \)th cross section during time period \( t \)
- \( (1 - \hat{\theta}_i) \bar{x}_{1i} \), where \( \bar{x}_{1i} \) are the means of the time-varying exogenous variables for the \( i \)th cross section
- \( (1 - \hat{\theta}_i) z_{1i} \)

Multiplication by the factor \( (1 - \hat{\theta}_i) \) is redundant in balanced data, but necessary in the unbalanced case to produce accurate instrumentation; see Gardner (1998).

Let \( k_1 \) equal the number of regressors in \( x_1 \), and let \( g_2 \) equal the number of regressors in \( z_2 \). Then the Hausman-Taylor model is identified only if \( k_1 \geq g_2 \); otherwise, no estimation takes place.

Hausman and Taylor (1981) describe a specification test that compares their model to fixed effects. For a null hypothesis of fixed effects, Hausman’s \( m \) statistic is calculated by comparing the parameter estimates and variance matrices for both models identically to how it is calculated for one-way random effects models; for more information, see the section “Hausman Specification Tests” on page 147. However, the degrees of freedom of the test are not based on matrix rank but instead are equal to \( k_1 - g_2 \).

**Amemiya-MaCurdy Estimation (AMACURDY)**

You perform Amemiya-MaCurdy estimation by specifying the AMACURDY option in the MODEL statement. The Amemiya and MaCurdy (1986) model is similar to the Hausman-Taylor model. Following the development in the section “Hausman-Taylor Estimation (HTAYLOR)” on page 142, estimation is identical up to the final 2SLS instrumental variables regression. In addition to the set of instruments used by the Hausman-Taylor estimator, use the following:

- \( x_{1i1}, x_{1i2}, \ldots, x_{1iT} \)

For each observation in the \( i \)th cross section, you use the data on the time-varying exogenous regressors for the entire cross section. Because of the structure of the added instruments, the Amemiya-MaCurdy estimator can be applied only to balanced data.

The Amemiya-MaCurdy model attempts to gain efficiency over Hausman-Taylor by adding instruments. This comes at a price of a more stringent assumption on the exogeneity of the \( x_1 \) variables. Although the Hausman-Taylor model requires only that the cross-sectional means of \( x_1 \) be orthogonal to \( v_i \), the Amemiya-MaCurdy estimation requires orthogonality at every point in time; see Baltagi (2013, sec. 7.4).

A Hausman specification test is provided to test the validity of the added assumption. Define \( \alpha' = (\beta_1', \beta_2', \gamma_1', \gamma_2') \), its Hausman-Taylor estimate as \( \hat{\alpha}_{HT} \), and its Amemiya-MaCurdy estimate as \( \hat{\alpha}_{AM} \). Under the null hypothesis, both estimators are consistent and \( \hat{\alpha}_{AM} \) is efficient. The Hausman test statistic is

\[
m = (\hat{\alpha}_{HT} - \hat{\alpha}_{AM})' (\hat{\Sigma}_{HT} - \hat{\Sigma}_{AM})^{-1} (\hat{\alpha}_{HT} - \hat{\alpha}_{AM})
\]
where $\hat{\Sigma}_{HT}$ and $\hat{\Sigma}_{AM}$ are variance-covariance estimates of $\hat{\alpha}_{HT}$ and $\hat{\alpha}_{AM}$, respectively. Under the null hypothesis, $m$ follows a $\chi^2$ distributed with degrees of freedom equal to the rank of $(\hat{\Sigma}_{HT} - \hat{\Sigma}_{AM})^{-1}$.

---

**Instrumental Variables Regression**

Recall the original specification of the regression model

$$y_{it} = \alpha + \sum_{k=1}^{K} x_{ik} \beta_k + u_{it} \quad i = 1, \ldots, N; \ t = 1, \ldots, T_i$$

Consider that a subset of the regressors are *endogenous*—that is, these regressors are correlated with the observation-level errors $u_{it}$.

Formally, divide $\mathbf{x}$ and $\mathbf{\beta}$ so that

$$y_{it} = x_{1it} \beta_1 + x_{2it} \beta_2 + u_{it}$$

where $x_{1it}$ are $k_1$ uncorrelated (exogenous) regressors, $x_{2it}$ are $k_2$ endogenous regressors, and the intercept (if your model has one) is contained in $x_{1it}$.

In matrix notation, the model is

$$\mathbf{y} = \mathbf{X}_1 \mathbf{\beta}_1 + \mathbf{X}_2 \mathbf{\beta}_2 + \mathbf{u}$$

where the matrix rows are arranged by cross section, and by time within cross section.

Instrumental variables (IV) regression is the process of using external regressors, called *instruments*, to deal with the endogeneity. Let $\mathbf{Z}$ represent a set of $p$ instruments such that $p \geq k_2$. The set of endogenous regressors, $\mathbf{X}_2$, are specified as effects in the ENDOGENOUS statement. The set of external instruments, $\mathbf{Z}$, are specified as effects in the INSTRUMENTS statement.

The CPANEL procedure performs instrumental variables regression by the method of two-stage least squares (2SLS). Depending on the error structure of $u_{it}$, there are four 2SLS methods available, as described in the following subsections.

**Instrumental Variables Pooled Regression (IVPOOLED)**

You perform IV pooled regression by specifying the IVPOOLED option in the MODEL statement. Pooled regression is standard ordinary least squares (OLS) regression without any cross-sectional or time effects. The error structure is simply $u_{it} = e_{it}$, where the $e_{it}$ are iid with zero mean and variance $\sigma_e^2$. The errors $e_{it}$ are uncorrelated with $x_{1it}$, but correlated with $x_{2it}$.

Estimation is performed by two-stage least squares (2SLS), where you form the full instrument set by combining the external instruments $\mathbf{Z}$ and the exogenous regression variables $\mathbf{X}_1$. At the first stage, the predictions $\hat{x}_{2it}$ are obtained from regressing $x_{2it}$ on $x_{1it}$ and $z_{it}$. At the second stage, $\hat{\beta} = (\hat{\beta}_1', \hat{\beta}_2')'$ is estimated by regressing $y$ on $x_{1it}$ and $\hat{x}_{2it}$.

Form $\mathbf{X}_*$ by stacking the data observations of $(x_{1it}, \hat{x}_{2it})$. The variance of $\hat{\beta}$ is

$$\text{Var} \left( \hat{\beta} \right) = \sigma_e^2 \left( \mathbf{X}_* \mathbf{X}_* \right)^{-1}$$
The error variance $\sigma_e^2$ is estimated by using residuals that are based on the original data:

$$\hat{\sigma}_e^2 = (M - K - 1)^{-1} \sum_{i=1}^{N} \sum_{t=1}^{T_i} \left( y_{it} - x_{1it} \hat{\beta}_1 - x_{2it} \hat{\beta}_2 \right)^2$$

**Instrumental Variables Between-Groups Regression (IVBTWNG)**

You perform IV between-groups regression by specifying the IVBTWNG option in the MODEL statement. When the data are balanced, between-groups IV regression is two-stage least squares (2SLS) regression performed on data that have been collapsed into cross-sectional means. When the data are unbalanced, the regression is weighted to assign more importance to larger cross sections. The cross-sectional means are multiplied by the weighting factor $w_i = \sqrt{T_i/T}$, where $T = M/N$ is the average cross section size.

**Instrumental Variables One-Way Fixed Effects (IVFIXONE)**

You perform IV one-way fixed-effects estimation by specifying the IVFIXONE option in the MODEL statement. The error structure for the one-way fixed-effects model is

$$u_{it} = v_i + e_{it}$$

where the $v_i$ are nonrandom parameters that are restricted to sum to 0, and the $e_{it}$ are iid with zero mean and variance $\sigma_e^2$. The errors $e_{it}$ are uncorrelated with $x_{1it}$, but correlated with $x_{2it}$.

The IV fixed-effects model can be estimated as an IV pooled regression, treating the $v_i$ as coefficients on dummy variables that identify the cross sections. However, when $N$ is large, you might want to estimate only $\beta$ and not $v_i$.

Let $Q_0 = \text{diag}(E_{T_i})$. The matrix $Q_0$ is the within transformation that removes means (and $v_i$) from the estimation. Define $y_w = Q_0 y$, and use the same transformation to form $X_{1w}$, $X_{2w}$ and $Z_w$. Because the within transformation also removes the intercept term, the intercept is not included as part of $X_1$.

The estimates $\hat{\beta}_{W2SLS}$ and $\text{Var}(\hat{\beta}_{W2SLS})$ are obtained by performing 2SLS regression of $y_w$ on $X_{1w}$ and $X_{2w}$, using $Z_w$ as external instruments. When forming the error variance $\hat{\sigma}_e^2$, the error degrees of freedom are adjusted to equal $M - N - K$ to account for the $N$ fixed effects.

The estimate of the intercept is

$$\hat{\alpha} = \bar{y}_. - \bar{x}_. \hat{\beta}_{W2SLS}$$

where $\bar{y}_.$ is the overall mean of $y_{it}$ and $\bar{x}_.$ is the overall mean of $x_{it}$.

Fixed effects are not estimated as part of the regression but can be obtained by specifying the PRINTFIXED option in the MODEL statement. Each fixed effect is estimated as

$$\hat{v}_i = \bar{y}_{it} - \bar{x}_{it} \hat{\beta}_{W2SLS}$$

where $\bar{y}_{it}$ and $\bar{x}_{it}$ are cross-sectional means.

Variance estimates of $\hat{\alpha}$ and $\hat{v}_i$ are obtained by the delta method. If you specify the NOINT option, then the $\hat{v}_i$ are shifted to absorb $\hat{\alpha}$.
A poolability test is an $F$ test of the null hypothesis that all fixed effects are jointly 0. Following the methodology of Wooldridge (2002, pp. 98–99) for restricted 2SLS, the test statistic is

$$F = \frac{(SSE_{2u} - SSE_{2w})/(N - 1)}{SSE_u/(M - N - K)} \sim F(N - 1, M - N - K)$$

$SSE_u$ is the final unrestricted error sum of squares

$$SSE_u = \left( y_w - X_{1w} \hat{\beta}_{1,W2SLS} - X_{2w} \hat{\beta}_{2,W2SLS} \right)' \left( y_w - X_{1w} \hat{\beta}_{1,W2SLS} - X_{2w} \hat{\beta}_{2,W2SLS} \right)$$

$SSE_{2u}$ is the second-stage unrestricted error sum of squares

$$SSE_{2u} = \left( y_w - X_{1w} \hat{\beta}_{1,W2SLS} - \hat{X}_{2w} \hat{\beta}_{2,W2SLS} \right)' \left( y_w - X_{1w} \hat{\beta}_{1,W2SLS} - \hat{X}_{2w} \hat{\beta}_{2,W2SLS} \right)$$

and $SSE_{2r}$ is the second-stage restricted error sum of squares, which is equal to the residual sum of squares from a pooled OLS regression of $y$ on $X_1$ and $X_2^* = \hat{X}_{2w} + P_0X_2$.

**Instrumental Variables One-Way Random Effects (IVRANONE)**

You perform IV one-way random-effects estimation by specifying the IVRANONE option in the MODEL statement. The specification for the one-way random-effects model is

$$u_{1t} = v_i + \epsilon_{1t}$$

where the $v_i$ are iid with zero mean and variance $\sigma_v^2$, and the $\epsilon_{1t}$ are iid with zero mean and variance $\sigma_e^2$. The regressors in $x_{1it}$ are orthogonal to $\epsilon_{1t}$, but the regressors in $x_{2it}$ are correlated with $\epsilon_{1t}$. This form of endogeneity is different from that assumed by Hausman and Taylor (1981), where $x_{2it}$ is correlated with $v_i$ but not with $\epsilon_{1t}$.

The estimation performed by the CPANEL procedure is based on the error-component two-stage least squares (EC2SLS) method of Baltagi (1981). Estimation proceeds in two steps.

First, estimates of the variance components are obtained by using a variant of the Swamy-Arora method. Consider the system of equations

$$E \left( \hat{u}_F Q_0 \hat{u}_F \right) = (M - N - K)\sigma_v^2$$

$$E \left( \hat{u}_B P_0 \hat{u}_B \right) = \left[ M - \text{tr} \left( (X_\alpha' P_0 X_\alpha)^{-1} X_\alpha' Z_0 Z_0' X_\alpha \right) \right] \sigma_v^2 + (N - K - 1)\sigma_e^2$$

where $\hat{u}_F$ are the residuals from IV one-way fixed effects, $\hat{u}_B$ are the residuals from IV between-groups regression, and the other quantities are as defined in the section “Swamy-Arora Method” on page 136. Estimates of $\sigma_v^2$ and $\sigma_e^2$ are obtained by setting the expected sums of squares to their observed values and solving.

Second, with the variance components in hand, you form a weight for each cross section:

$$\hat{\theta}_l = 1 - \hat{\sigma}_e/\hat{w}_i$$

where $\hat{w}_i^2 = T_i \hat{\sigma}_v^2 + \hat{\sigma}_e^2$. Taking $\hat{\theta}_l$, you form the partial deviations $y_{it}^* = y_{it} - \hat{\theta}_l \tilde{y}_{it}$. Estimation then proceeds as a 2SLS regression of $y_{it}^*$ on $x_{1it}^*$ and $x_{2it}^*$. For the first-stage regression, use the following instruments: $x_{1it}^*$, $x_{1i:}$, $z_{i:}$, and $z_{i:}$.

A Hausman specification test that compares IV random effects to IV fixed effects is formed by using the method described in the section “Hausman Specification Tests” on page 147. Failure to reject the null hypothesis favors the random-effects specification.
Poolability Test for Fixed Effects

A poolability test is an $F$ test of the null hypothesis that all fixed effects are jointly 0; it is obtained by comparing fixed-effects estimates to those from pooled regression. The $F$ statistic is

$$F = \frac{(SSE_r - SSE_u) / df_1}{SSE_u / df_2} \sim F(df_1, df_2)$$

where $SSE_r$ is the error sum of squares from the restricted model (pooled regression) and $SSE_u$ is the error sum of squares from the unrestricted fixed-effects model.

The numerator degrees of freedom, $df_1$, equals $N - 1$ for one-way models and $(N - 1) + (T - 1)$ for two-way models. The denominator degrees of freedom, $df_2$, is equal to the error degrees of freedom from the fixed-effects estimation. If you specify the NOINT option, add 1 to $df_1$ to account for the added restriction to the pooled regression.

Hausman Specification Tests

For models that include random effects, the CPANEL procedure outputs the results of the Hausman (1978) specification test. This test was also proposed by Wu (1973) and further extended in Hausman and Taylor (1982).

Consider two estimators, $\hat{\beta}_e$ and $\hat{\beta}_c$, which under the null hypothesis are both consistent but only $\hat{\beta}_e$ is asymptotically efficient. Under the alternative hypothesis, only $\hat{\beta}_c$ is consistent. The $m$ statistic is

$$m = (\hat{\beta}_c - \hat{\beta}_e)'(\hat{\Sigma}_c - \hat{\Sigma}_e)^{-1}(\hat{\beta}_c - \hat{\beta}_e)$$

where $\hat{\Sigma}_c$ and $\hat{\Sigma}_e$ are estimates of the asymptotic covariance matrices of $\hat{\beta}_c$ and $\hat{\beta}_e$. The statistic $m$ follows a $\chi^2$ distribution with $k$ degrees of freedom, where $k$ is the rank of $(\hat{\Sigma}_c - \hat{\Sigma}_e)^{-1}$. This rank is normally equal to the dimension of $\hat{\beta}_c - \hat{\beta}_e$, but is reduced when regressors that are constant within cross sections are dropped from the fixed-effects model.

The null hypothesis is that the effects are independent of the regressors. Under the null hypothesis, the fixed-effects estimator is consistent yet inefficient, whereas the random-effects estimator is both consistent and efficient. Failure to reject the null hypothesis favors the random-effects specification.

Restricted Estimation

The CPANEL procedure can fit models that have linear restrictions, producing a Lagrange multiplier (LM) test for each restriction. Consider a set of $J$ linear restrictions $R\beta = q$, where $R$ is $J \times K$ and $q$ is $J \times 1$.

The restricted regression is performed by minimizing the error sum of squares subject to the restrictions. In matrix terms, the Lagrangian for this problem is

$$L = (y - X\hat{\beta})'(y - X\hat{\beta}) + 2\lambda(R\hat{\beta} - q)$$

The Lagrangian is minimized by the restricted estimator $\hat{\beta}^*$, and it can be shown that

$$\hat{\beta}^* = \hat{\beta} - (X'X)^{-1}R'\lambda$$

where $\hat{\beta}$ is the unrestricted estimator.
Because $R\beta^* = q$, you can solve for $\lambda$ to obtain the Lagrange multipliers

$$\lambda^* = \left[ R(X'X)^{-1}R' \right]^{-1} (R\hat{\beta} - q)$$

The standard errors of the Lagrange multipliers are the square roots of the diagonal elements of the variance matrix

$$\text{Var}(\lambda^*) = \hat{\sigma}^2 \left[ R(X'X)^{-1}R' \right]^{-1}$$

where $\hat{\sigma}^2$ is the mean square error (MSE) under the null hypothesis. A significant Lagrange multiplier indicates a restriction that is not binding.

**Linear Hypothesis Testing**

Consider a linear hypothesis of the form $R\beta = q$, where $R$ is $J \times K$ and $q$ is $J \times 1$. The Wald test statistic is

$$\chi^2_W = (R\hat{\beta} - q)' \left( R\hat{V}R' \right)^{-1} (R\hat{\beta} - q)$$

where $\hat{V}$ is the estimated variance of $\hat{\beta}$.

In simple linear models, the Wald test statistic is equal to the $F$ test statistic

$$F = \frac{(\text{SSE}_r - \text{SSE}_u)/J}{\text{SSE}_u/df_e}$$

where $\text{SSE}_r$ is the restricted error sum of squares, $\text{SSE}_u$ is the unrestricted error sum of squares, and $df_e$ is the unrestricted error degrees of freedom.

The $F$ statistic represents a more direct comparison of the restricted model to the unrestricted model. Comparing error sums of squares is appealing in complex models for which restrictions are applied not only during the final regression but also during intermediate calculations.

The likelihood ratio (LR) test and the Lagrange multiplier (LM) test are derived from the $F$ statistic. The LR test statistic is

$$\chi^2_{LR} = M \ln \left[ 1 + \frac{JF}{M - K} \right]$$

The LM test statistic is

$$\chi^2_{LM} = M \left[ \frac{JF}{M - K + JF} \right]$$

The distribution of these test statistics is $\chi^2$ with $J$ degrees of freedom. The three tests are asymptotically equivalent, but they possess different small-sample properties. For more information, see Greene (2000, p. 392) and Davidson and MacKinnon (1993, pp. 456–458).
R-Square

The R-square statistic is the proportion of variability in the dependent variable that is attributed to the independent variables. Because of the transformations used prior to fitting the final regression model, the conventional R-square measure is not appropriate for most of the models supported by the CPANEL procedure. In random-effects models that use a GLS transform, PROC CPANEL calculates the modified R-square statistic proposed by Buse (1973),

\[ R^2 = 1 - \frac{SSE}{y'D\hat{\Omega}^{-1}Dy} \]

where SSE is the error sum of squares from the final model fit, \( \hat{\Omega}^{-1/2} \) represents the GLS transform, and \( D = I_M - a^{-1}J_M\hat{\Omega}^{-1}, \) for \( a = j_M\hat{\Omega}^{-1}j_M. \)

In GLS models that do not have an intercept, the alternate R-square measure, which is attributed to Theil (1961), is calculated as follows:

\[ R^2 = 1 - \frac{SSE}{y'\hat{\Omega}^{-1}y} \]

In fixed-effects models, the R-square measure is

\[ R^2 = 1 - \frac{SSE}{y'_w y_w} \]

where \( y_w \) is the within-transformed dependent variable.

In the case of pooled OLS estimation, all three of the R-square formulas reduce to the usual R-square statistic for linear models.

The R-square statistic is not appropriate for models that contain external instruments, and thus is not reported for IV regressions.

Displayed Output

The following sections describe the output that PROC CPANEL produces. The output is organized into various tables, which are discussed in their order of appearance.

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.

Model Description

The “Model Information” table displays basic information about the model: the response variable, the name of the data table, the estimation method, the number of observations, the number of cross sections, and the length of the time series. The reported numbers are useful for verifying that observations were not dropped unexpectedly and for verifying that the data are balanced when they should be.
Chapter 4: The CPANEL Procedure

**Fit Statistics**

The “Fit Statistics” table displays model fit statistics that are widely used in linear models. This table reports the error sum of squares, mean square error, root mean square error, R-square statistic, and the error degrees of freedom.

**Variance Component Estimates**

If you fit a model that contains random effects, then the “Variance Component Estimates” table reports the variances and standard deviations of the observation-level errors, the cross-sectional effects, and the time effects (in two-way models).

**Test for Poolability**

If you fit a fixed-effects model, then an $F$ test of poolability is produced. The null hypothesis is that all fixed effects are jointly 0, making pooled OLS regression a viable alternative. The “F Test for No Fixed Effects” table reports the numerator degrees of freedom, denominator degrees of freedom, test statistic, and $p$-value.

**Hausman Specification Tests**

If you fit a model that contains random effects, then a table that summarizes the Hausman test is produced. The table presents the number of coefficients included in the test, the degrees of freedom, the $m$-statistic, and the $p$-value.

If you fit a one-way random-effects model (RANONE), then the fitted model is compared to the alternative model, a one-way fixed-effects (FIXONE) model.

If you fit a two-way random-effects model (RANTWO), then the fitted model is compared to the alternative model, a two-way fixed-effects (FIXTWO) model.

If you fit a Hausman-Taylor model (HTAYLOR), then the fitted model is compared to the alternative model, a one-way fixed-effects (FIXONE) model.

If you fit an Amemiya-MaCurdy model (AMACURDY), then the fitted model is compared to the alternative model, a Hausman-Taylor (HTAYLOR) model.

If you fit an instrumental variables random-effects model (IVRANONE), then the fitted model is compared to the alternative model, an IV fixed-effects (IVFIXONE) model.

**Structural Summary**

If you fit an instrumental variables (IV) model, then the “Structural Summary” table lists the endogenous effects and the instrumental variables.

**Parameter Estimates**

The parameter estimates, degrees of freedom, standard errors, $t$ statistics and $p$-values for the hypothesis that the parameter is 0 are presented in the “Parameter Estimates” table. If one or more of your regression variables are labeled, then a column that contains variable labels is added to this table. Use the NOLABEL option in the MODEL statement to suppress this column.
If you specify a RESTRICT statement, then the table includes a row for each linear restriction. This row reports the restriction label, Lagrange multiplier (LM) statistic, standard error, \( t \)-statistic, \( p \)-value, and a label that describes the restriction.

If you fit an instrumental variables (IV) model, then the table includes a column that identifies the endogenous variables. If you fit a Hausman-Taylor or Amemiya-MaCurdy model, then both the correlated variables and the time-invariant variables are identified in the table.

**Parameter Estimates Covariance Matrix**

When you specify the **COVB** option in the **PROC CPANEL** statement, the CPANEL procedure displays the covariance matrix of the parameter estimates.

**Parameter Estimates Correlation Matrix**

When you specify the **CORRB** option in the **PROC CPANEL** statement, the CPANEL procedure displays the correlation matrix of the parameter estimates.

**Tests of Linear Hypotheses**

If you specify a TEST statement, a table that describes the tests is provided. For each test of hypothesis, the table reports the test label, the type of test (Wald, LR, or LM), the test statistic, the \( p \)-value, and a label that states the null hypothesis.

**Model Comparison Tables**

If you specify a COMPARE statement, two comparison tables are produced.

The first table is the “Comparison of Model Statistics” table, which lists the number of observations, number of cross sections, time series length, mean square error (MSE), root MSE, and R-square for each fitted model.

The second is table is the “Comparison of Model Parameter Estimates” table, which presents the parameter estimate and standard error of each regressor for each fitted model.

Both tables can be customized; for more information, see the section “COMPARE Statement” on page 121.

---

**ODS Table Names**

Each table that the CPANEL 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 4.4.

<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 <strong>CLASS</strong> statement</td>
<td>CLASS</td>
<td>Default</td>
</tr>
<tr>
<td>CorrB</td>
<td>Correlation matrix of parameter estimates</td>
<td>MODEL</td>
<td>CORRB</td>
</tr>
</tbody>
</table>
Table 4.4  continued

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
<th>Statement</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>CovB</td>
<td>Covariance matrix of parameter estimates</td>
<td>MODEL</td>
<td>COVB</td>
</tr>
<tr>
<td>FitStatistics</td>
<td>Fit statistics</td>
<td>Default</td>
<td></td>
</tr>
<tr>
<td>FixedEffectsTest</td>
<td>$F$ test for poolability</td>
<td>MODEL</td>
<td>FIXONE, FIXTWO, or IVFIXONE RANONE, RANTWO, or IVRANONE</td>
</tr>
<tr>
<td>HausmanTestForRandom</td>
<td>Hausman test for random effects</td>
<td>MODEL</td>
<td></td>
</tr>
<tr>
<td>HausmanTestVsFixed</td>
<td>Hausman test for Hausman-Taylor models</td>
<td>MODEL</td>
<td>HTAYLOR</td>
</tr>
<tr>
<td>HausmanTestVsHausmanTaylor</td>
<td>Hausman test for Amemiya-MaCurdy models</td>
<td>MODEL</td>
<td>AMACURDY</td>
</tr>
<tr>
<td>ModelDescription</td>
<td>Model summary and data counts</td>
<td>Default</td>
<td></td>
</tr>
<tr>
<td>ParameterComparisonTable</td>
<td>Comparison of model parameter estimates, standard errors, and $t$ tests</td>
<td>COMPARE</td>
<td></td>
</tr>
<tr>
<td>ParameterEstimates</td>
<td>Solutions for the parameter estimates associated with effects in MODEL statements</td>
<td>Default</td>
<td></td>
</tr>
<tr>
<td>StatComparisonTable</td>
<td>Comparison of model summary statistics</td>
<td>COMPARE</td>
<td></td>
</tr>
<tr>
<td>StructuralSummary</td>
<td>Endogenous and instrumental variable lists</td>
<td>MODEL</td>
<td>IVPOOLED, IVBTWNG, IVFIXONE, or IVRANONE</td>
</tr>
<tr>
<td>TestResults</td>
<td>Hypothesis tests of linear restrictions</td>
<td>TEST</td>
<td></td>
</tr>
<tr>
<td>VarianceComponents</td>
<td>Variance component estimates</td>
<td>MODEL</td>
<td>RANONE, RANTWO, HTAYLOR, AMACURDY, or IVRANONE</td>
</tr>
</tbody>
</table>
**Example 4.1: The Airline Cost Data: Fixed Effects**

The Christenson Associates airline data are a frequently cited data set (Greene 2000). The data measure the costs, prices of inputs, and utilization rates for six airlines from 1970 to 1984. This example analyzes the log transformations of cost (variable \( C \)), quantity (variable \( Q \)), and price (variable \( PF \)), and the untransformed load factor (variable \( LF \)). You speculate the following model,

\[
\log(C_{it}) = \alpha + \beta_1 \log(Q_{it}) + \beta_2 \log(PF_{it}) + \beta_3 LF_{it} + v_i + \epsilon_{it}
\]

where the \( v_i \) are airline effects. The actual model in the original, untransformed variables is highly nonlinear:

\[
C_{it} = \exp(\alpha + \beta_3 LF_{it} + v_i + \epsilon_{it}) Q_{it}^{\beta_1} PF_{it}^{\beta_2}
\]

The following statements create the data and perform the necessary log transformations. These statements assume that your CAS engine libref is named mycas, but you can substitute any appropriately defined CAS engine libref.

```plaintext
data mycas.airline;
  input Obs Airline T C Q PF LF;
  Year = T + 1969;
  lC = log(C);
  lQ = log(Q);
  lPF = log(PF);
  label lC = "Log Transformation of Costs";
  label lQ = "Log Transformation of Quantity";
  label lPF = "Log Transformation of Price of Fuel";
  label LF = "Load Factor (utilization index)";
datalines;
1 1 1 1140640 0.95276 106650 0.53449
2 1 2 1215690 0.98676 110307 0.53233
3 1 3 1309570 1.09198 110574 0.54774
4 1 4 1511530 1.17578 121974 0.54085
5 1 5 1676730 1.16017 196606 0.59117
... more lines ...
```

The following statements fit a one-way fixed-effects model:

```plaintext
proc cpanel data = mycas.airline;
  id Airline Year;
  model lC = lQ lPF LF / fixone printfixed;
run;
```

Output 4.1.1 provides a model and data description. There are six cross sections and 15 time points.
Output 4.1.1  Airline Cost Data, Model Description

The CPANEL Procedure
One-Way Fixed Effects (FixOne)
Dependent Variable: IC (Log Transformation of Costs)

<table>
<thead>
<tr>
<th>Model Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimation Method</td>
</tr>
<tr>
<td>Data Set</td>
</tr>
<tr>
<td>Number of Observations</td>
</tr>
<tr>
<td>Number of Cross Sections</td>
</tr>
<tr>
<td>Time Series Length</td>
</tr>
</tbody>
</table>

The R-square and degrees of freedom can be seen in Output 4.1.2. The R-square statistic nearly 1, indicating a reasonable fit. The error degrees of freedom is derived from 90 observations minus 5 cross sections, minus 4 regressors.

Output 4.1.2  Airline Cost Data, Fit Statistics

<table>
<thead>
<tr>
<th>Fit Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSE</td>
</tr>
<tr>
<td>DFE</td>
</tr>
<tr>
<td>MSE</td>
</tr>
<tr>
<td>Root MSE</td>
</tr>
<tr>
<td>R-Square</td>
</tr>
</tbody>
</table>

The $F$ test for fixed effects is shown in Output 4.1.3. You easily reject the null hypothesis of poolability. There are significant effects due to airlines, and it would be unreasonable to perform pooled OLS regression that ignores these effects.

Output 4.1.3  Airline Cost Data, Test for Fixed Effects

<table>
<thead>
<tr>
<th>F Test for No Fixed Effects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Num DF</td>
</tr>
<tr>
<td>--------</td>
</tr>
<tr>
<td>5</td>
</tr>
</tbody>
</table>

The PRINTFIXED option in the MODEL statement provides estimates of the airline effects (which are not displayed by default). Looking at the parameters, only two airlines (3 and 4) are significantly different from average. Quantity and fuel price have positive effects on cost, but load factors negatively affect costs. Because cost, quantity, and fuel price are log-transformed, the coefficients for quantity and price are interpreted as elasticities of cost. The coefficient for (log) fuel price is 0.417, meaning that you would associate a 10% increase in fuel price with a 4.17% increase in costs.
Example 4.1: The Airline Cost Data: Fixed Effects

You suspect that there might be other factors at play, and so you augment your model to include time effects. The following statements fit a two-way model, a model with both airline and time effects:

```sas
proc cpanel data = mycas.airline;
  id Airline Year;
  model lC = lQ lPF LF / fixtwo printfixed;
run;
```

The $F$ test and parameter estimates for the two-way model are provided in Output 4.1.5.

### Output 4.1.5  Airline Cost Data, Two-Way Fixed Effects

#### The CPANEL Procedure
**Two-Way Fixed Effects (FixTwo)**
**Dependent Variable: IC (Log Transformation of Costs)**

<table>
<thead>
<tr>
<th>F Test for No Fixed Effects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Num DF</td>
</tr>
<tr>
<td>--------</td>
</tr>
<tr>
<td>19</td>
</tr>
</tbody>
</table>
Output 4.1.5 continued

| Variable | DF | Estimate | Standard Error | t Value | Pr > |t| | Label |
|----------|----|----------|----------------|---------|-------|-----|--------|
| Intercept | 1 | 12.66527 | 2.0810 | 6.09 | <.0001 | Intercept |
| IQ       | 1 | 0.817264 | 0.0318 | 25.66 | <.0001 | Log Transformation of Quantity |
| IPF      | 1 | 0.168732 | 0.1635 | 1.03 | 0.3057 | Log Transformation of Price of Fuel |
| LF       | 1 | -0.88267 | 0.2617 | -3.37 | 0.0012 | Load Factor (utilization index) |
| CS 1     | 1 | 0.128306 | 0.0460 | 2.79 | 0.0069 | Cross-Sectional Effect: 1 |
| CS 2     | 1 | 0.065481 | 0.0390 | 1.68 | 0.0975 | Cross-Sectional Effect: 2 |
| CS 3     | 1 | -0.18948 | 0.0156 | -12.14 | <.0001 | Cross-Sectional Effect: 3 |
| CS 4     | 1 | 0.134259 | 0.0183 | 7.33 | <.0001 | Cross-Sectional Effect: 4 |
| CS 5     | 1 | -0.09264 | 0.0373 | -2.48 | 0.0155 | Cross-Sectional Effect: 5 |
| CS 6     | 1 | -0.04593 | 0.0416 | -1.10 | 0.2736 | Cross-Sectional Effect: 6 |
| Time 1970 | 1 | -0.37386 | 0.1919 | -1.95 | 0.0555 | Time Effect: 1970 |
| Time 1971 | 1 | -0.31916 | 0.1861 | -1.72 | 0.0909 | Time Effect: 1971 |
| Time 1972 | 1 | -0.27654 | 0.1833 | -1.51 | 0.1362 | Time Effect: 1972 |
| Time 1973 | 1 | -0.22292 | 0.1730 | -1.29 | 0.2019 | Time Effect: 1973 |
| Time 1974 | 1 | -0.15388 | 0.0864 | -1.78 | 0.0796 | Time Effect: 1974 |
| Time 1975 | 1 | -0.10805 | 0.0449 | -2.41 | 0.0188 | Time Effect: 1975 |
| Time 1976 | 1 | -0.07686 | 0.0319 | -2.41 | 0.0188 | Time Effect: 1976 |
| Time 1977 | 1 | -0.02072 | 0.0204 | -1.01 | 0.3145 | Time Effect: 1977 |
| Time 1978 | 1 | 0.047197 | 0.0291 | 1.62 | 0.1093 | Time Effect: 1978 |
| Time 1979 | 1 | 0.091657 | 0.0811 | 1.13 | 0.2627 | Time Effect: 1979 |
| Time 1980 | 1 | 0.207197 | 0.1491 | 1.39 | 0.1693 | Time Effect: 1980 |
| Time 1981 | 1 | 0.28534 | 0.1756 | 1.62 | 0.1089 | Time Effect: 1981 |
| Time 1982 | 1 | 0.301255 | 0.1660 | 1.81 | 0.0741 | Time Effect: 1982 |
| Time 1983 | 1 | 0.300347 | 0.1536 | 1.96 | 0.0547 | Time Effect: 1983 |
| Time 1984 | 1 | 0.319001 | 0.1475 | 2.16 | 0.0341 | Time Effect: 1984 |

Only a few of the time effects are significant, but there is an overall time trend of increasing costs. The time period of the data spans the OPEC oil embargoes and the dissolution of the Civil Aeronautics Board (CAB). These are two possible explanations for the rising costs.

A surprising result is that the fuel cost is not significant in the two-way model. If the time effects are proxies for the effect of the oil embargoes, then the effect of fuel price might be subsumed by the time effects. If the time dummy variables are proxies for the dissolution of the CAB, then the effect of load factors is not precisely estimated.
Example 4.2: Analyzing Demand for Liquid Assets: Random Effects

Feige (1964) provides data on the demand for liquid assets. The data are for six states and the District of Columbia (CA, DC, FL, IL, NY, TX, and WA) and were collected each year from 1949 to 1959. All variables are log-transformed.

The following statements create the Assets data table. These statements assume that your CAS engine libref is named mycas, but you can substitute any appropriately defined CAS engine libref.

```plaintext
data mycas.Assets;
  length state $ 2;
  input state $ year d t s y rd rt rs;
  label d = 'Per Capita Demand Deposits'
    t = 'Per Capita Time Deposits'
    s = 'Per Capita S & L Association Shares'
    y = 'Permanent Per Capita Personal Income'
    rd = 'Service Charge on Demand Deposits'
    rt = 'Interest on Time Deposits'
    rs = 'Interest on S & L Association Shares';
datalines;
CA 1949 6.2785 6.1924 4.4998 7.2056 -1.0700 0.1080 1.0664
CA 1950 6.4019 6.2106 4.6821 7.2889 -1.0106 0.1501 1.0767
CA 1951 6.5058 6.2729 4.8598 7.3827 -1.0024 0.4008 1.1291
CA 1952 6.4785 6.2729 5.0039 7.4000 -0.9970 0.4492 1.1227
CA 1953 6.4118 6.2538 5.1761 7.4200 -0.8916 0.4662 1.2110
CA 1954 6.4520 6.2971 5.3613 7.4478 -0.6951 0.4756 1.1924
... more lines ...
```

The data contain per capita consumptions for three liquid assets: demand deposits such as checking, time deposits, and savings and loan (S&L) shares. You posit a linear model for per capita demand deposits, with random effects for states.

The following statements fit a one-way random-effects model:

```plaintext
proc cpanel data = mycas.Assets;
  id state year;
  model d = y rd rt rs / ranone;
run;
```

The regression results are provided in Output 4.2.1.

The “Variance Component Estimates” table provides the estimated variances and standard deviations for the cross-sectional (state) effects in addition to the overall errors. A majority of the overall error variance can be attributed to differences between states, not differences within states.

The “Hausman Test for Random Effects” table shows the result of a Hausman specification test. The null hypothesis is that state effects can be treated as random (random-effects model) and that they do not need to be estimated directly (fixed-effects model). The test results favor the random-effects specification that is used to generate this output.
Output 4.2.1 Demand for Demand Deposits, One-Way Random-Effects Model

The parameter estimate for the variable $Y$ is greater than 1, indicating that demand is elastic to income—income has a more than proportional positive association with the demand for demand deposits. The coefficient on the variable $RD$ indicates that demand deposits increase significantly as the service charge is reduced.

The variables $RT$ and $RS$ represent positive aspects of competing products, and you would expect these variables to affect demand negatively. The coefficient for $RS$ meets that expectation, but the coefficient for $RT$ is not significant.

The previous analysis used the default Swamy-Arora method to estimate the variance components. The CPANEL procedure supports four other methods, and you might be interested in how the different methods affect the analysis.
The following statements fit the model by using all five methods and include a COMPARE statement to compare the results:

```sas
proc cpanel data = mycas.Assets;
  id state year;
  sa: model d = y rd rt rs / ranone vcomp = sa;
  wh: model d = y rd rt rs / ranone vcomp = wh;
  wk: model d = y rd rt rs / ranone vcomp = wk;
  fb: model d = y rd rt rs / ranone vcomp = fb;
  nl: model d = y rd rt rs / ranone vcomp = nl;
  compare / mstat(varcs varerr);
run;
```

The tables produced by the COMPARE statement are provided in Output 4.2.2.

**Output 4.2.2** Comparison of Variance-Component Methods, Assets Data

### The CPANEL Procedure

**Model Comparison**

Dependent Variable: d (Per Capita Demand Deposits)

<table>
<thead>
<tr>
<th>Comparison of Model Statistics</th>
<th>SA RanOne</th>
<th>WH RanOne</th>
<th>WK RanOne</th>
<th>FB RanOne</th>
<th>NL RanOne</th>
</tr>
</thead>
<tbody>
<tr>
<td>Var due to Cross Sections</td>
<td>0.0464</td>
<td>0.0315</td>
<td>0.0315</td>
<td>0.0291</td>
<td>0.0327</td>
</tr>
<tr>
<td>Var due to Error</td>
<td>0.001340</td>
<td>0.000107</td>
<td>0.001340</td>
<td>0.001340</td>
<td>0.001149</td>
</tr>
</tbody>
</table>

### Comparison of Model Parameter Estimates

<table>
<thead>
<tr>
<th>Variable</th>
<th>Estimate</th>
<th>Std Err</th>
<th>Estimate</th>
<th>Std Err</th>
<th>Estimate</th>
<th>Std Err</th>
<th>Estimate</th>
<th>Std Err</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intercept</td>
<td>-1.643246</td>
<td>0.684144</td>
<td>1.136776</td>
<td>0.099816</td>
<td>0.032076</td>
<td>0.029138</td>
<td>-0.412126</td>
<td>0.119181</td>
</tr>
<tr>
<td>y</td>
<td>-1.723092</td>
<td>0.681184</td>
<td>1.145844</td>
<td>0.099776</td>
<td>0.033397</td>
<td>0.028570</td>
<td>-0.414540</td>
<td>0.117486</td>
</tr>
<tr>
<td>rd</td>
<td>-1.742581</td>
<td>0.680541</td>
<td>1.148051</td>
<td>0.099761</td>
<td>0.033718</td>
<td>0.029416</td>
<td>-0.410731</td>
<td>0.119968</td>
</tr>
<tr>
<td>rt</td>
<td>-1.680406</td>
<td>0.682676</td>
<td>1.141001</td>
<td>0.099802</td>
<td>0.032692</td>
<td>0.029266</td>
<td>-0.411500</td>
<td>0.120160</td>
</tr>
<tr>
<td>rs</td>
<td>-1.742581</td>
<td>0.682676</td>
<td>1.141001</td>
<td>0.099802</td>
<td>0.032692</td>
<td>0.029266</td>
<td>-0.411500</td>
<td>0.120160</td>
</tr>
</tbody>
</table>

You conclude that how you estimate variance components has little bearing on the regression results.

---

**Example 4.3: Panel Study of Income Dynamics (PSID): Hausman-Taylor Models**

Cornwell and Rupert (1988) analyze data from the Panel Study of Income Dynamics (PSID), an income study of 595 individuals over the seven-year period, 1976–1982 inclusive. Of particular interest is the effect of additional schooling on wages. The analysis here replicates that of Baltagi (2013, sec. 7.5), where it is surmised that covariate correlation with individual effects makes a standard random-effects model inadequate.
The following statements create the PSID data. These statements assume that your CAS engine libref is named mycas, but you can substitute any appropriately defined CAS engine libref.

```
data mycas.psid;
  input id t lwage wks south smsa ms exp exp2 occ ind union fem blk ed;
  label id = 'Person ID'
  t = 'Time'
  lwage = 'Log of wages'
  wks = 'Weeks worked'
  south = '1 if resides in the South'
  smsa = '1 if resides in SMSA'
  ms = '1 if married'
  exp = 'Years full-time experience'
  exp2 = 'exp squared'
  occ = '1 if blue-collar occupation'
  ind = '1 if manufacturing'
  union = '1 if union contract'
  fem = '1 if female'
  blk = '1 if black'
  ed = 'Years of education';
datalines;
  1   1   5.5606799126 32 1 0 1 3 9 0 0 0 0 0 9
  1   2   5.7203102112 43 1 0 1 4 16 0 0 0 0 0 9
  1   3   5.9964499474 40 1 0 1 5 25 0 0 0 0 0 9
  1   4   5.9964499474 39 1 0 1 6 36 0 0 0 0 0 9
  1   5   6.0614600182 42 1 0 1 7 49 0 1 0 0 0 9
  1   6   6.1737899780 35 1 0 1 8 64 0 1 0 0 0 9
  1   7   6.2441701889 32 1 0 1 9 81 0 1 0 0 0 9
  2   1   6.1633100510 34 0 0 1 30 900 1 0 0 0 0 11
  2   2   6.2146100998 27 0 0 1 31 961 1 0 0 0 0 11
  2   3   6.2634000778 33 0 0 1 32 1024 1 1 1 0 0 11
  2   4   6.5439100266 30 0 0 1 33 1089 1 1 0 0 0 11
  2   5   6.6970300674 30 0 0 1 34 1156 1 1 0 0 0 11
  2   6   6.7912201881 37 0 0 1 35 1225 1 1 0 0 0 11
  2   7   6.8156399727 30 0 0 1 36 1296 1 1 0 0 0 11
... more lines ...
```

You begin by fitting a one-way random-effects model:

```
proc cpanel data = mycas.psid;
  id id t;
  model lwage = wks south smsa ms exp exp2 occ
                 ind union fem blk ed / ranone;
run;
```

The output is shown in Output 4.3.1. The coefficient on the variable ED (which represents years of education) estimates that an additional year of schooling is associated with about a 10% increase in wages. However, the results of the Hausman test for random effects show a serious violation of the random-effects assumptions, namely that the regressors are independent of the individual effects.
Output 4.3.1 One-Way Random Effects Estimation

The CPANEL Procedure
One-Way Random Effects (RanOne)
Swamy and Arora Variance Components
Dependent Variable: lwage (Log of wages)

<table>
<thead>
<tr>
<th>Model Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimation Method</td>
</tr>
<tr>
<td>Data Set</td>
</tr>
<tr>
<td>Number of Observations</td>
</tr>
<tr>
<td>Number of Cross Sections</td>
</tr>
<tr>
<td>Time Series Length</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variance Component Estimates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
</tr>
<tr>
<td>--------</td>
</tr>
<tr>
<td>Cross Sections</td>
</tr>
<tr>
<td>Error</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hausman Test For Random Effects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coefficients</td>
</tr>
<tr>
<td>----------------</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter Estimates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
</tr>
<tr>
<td>-----------</td>
</tr>
<tr>
<td>Intercept</td>
</tr>
<tr>
<td>wks</td>
</tr>
<tr>
<td>south</td>
</tr>
<tr>
<td>smsa</td>
</tr>
<tr>
<td>ms</td>
</tr>
<tr>
<td>exp</td>
</tr>
<tr>
<td>exp2</td>
</tr>
<tr>
<td>occ</td>
</tr>
<tr>
<td>ind</td>
</tr>
<tr>
<td>union</td>
</tr>
<tr>
<td>fem</td>
</tr>
<tr>
<td>blk</td>
</tr>
<tr>
<td>ed</td>
</tr>
</tbody>
</table>

An alternative could be a fixed-effects (FIXONE) model, but that model would not permit estimation of the coefficient for ED, which does not vary within individuals. A compromise is the Hausman-Taylor model, for which you stipulate a set of covariates that are correlated with the individual effects (but uncorrelated with the observation-level errors). You specify the correlated variables in the CORRELATED statement:

```sql
proc cpanel data = mycas.psid;
   id id t;
   model lwage = wks south smsa ms exp exp2 occ
      ind union fem blk ed / htaylor;
   correlated wks ms exp exp2 union ed;
run;
```
The results are shown in Output 4.3.2. The table of parameter estimates has an added column, Type, which identifies the regressors that are assumed to be correlated with individual effects (C) and the regressors that do not vary within cross sections (TI). It was stated previously that the Hausman-Taylor model is a compromise between fixed-effects and random-effects models, and you can think of the compromise this way: You want to fit a random-effects model, but the correlated (C) variables make that model invalid. So you fall back to the consistent fixed-effects model, but then the time-invariant (TI) variables are the problem because they would be dropped from that model. The solution is to use the Hausman-Taylor estimator.

The estimation results show that an additional year of schooling is now associated with a 13.8% increase in wages. Also presented is a Hausman test that compares this model to the fixed-effects model. As was the case previously when you fit the random-effects model, you can think of the Hausman test as a referendum on the assumptions you are making. For this estimation, it seems that your choice of variables to treat as correlated is adequate.

Output 4.3.2  Hausman-Taylor Estimation

The CPANEL Procedure
Hausman and Taylor Model for Correlated Individual Effects
Dependent Variable: lwage (Log of wages)

<table>
<thead>
<tr>
<th>Variance Component Estimates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
</tr>
<tr>
<td>Cross Sections</td>
</tr>
<tr>
<td>Error</td>
</tr>
</tbody>
</table>

Hausman Test vs. Fixed Effects

<table>
<thead>
<tr>
<th>Coefficients</th>
<th>DF</th>
<th>m Value</th>
<th>Pr &gt; m</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>9</td>
<td>3</td>
<td>5.26</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter Estimates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
</tr>
<tr>
<td>Intercept</td>
</tr>
<tr>
<td>wks</td>
</tr>
<tr>
<td>south</td>
</tr>
<tr>
<td>smsa</td>
</tr>
<tr>
<td>ms</td>
</tr>
<tr>
<td>exp</td>
</tr>
<tr>
<td>exp2</td>
</tr>
<tr>
<td>occ</td>
</tr>
<tr>
<td>ind</td>
</tr>
<tr>
<td>union</td>
</tr>
<tr>
<td>fem</td>
</tr>
<tr>
<td>blk</td>
</tr>
<tr>
<td>ed</td>
</tr>
</tbody>
</table>

C: correlated with the individual effects
TI: constant (time-invariant) within cross sections
At its core, the Hausman-Taylor estimator is an instrumental variables regression, where the instruments are derived from regressors that are assumed to be uncorrelated with the individual effects. Technically, it is the cross-sectional means of these variables that need to be uncorrelated, not the variables themselves.

The Amemiya-MaCurdy model is a close relative of the Hausman-Taylor model. The only difference between the two is that the Amemiya-MaCurdy model makes the added assumption that the regressors (and not just their means) are uncorrelated with the individual effects. By making that assumption, the Amemiya-MaCurdy model can take advantage of a more efficient set of instrumental variables.

The following statements fit the Amemiya-MaCurdy model:

```latex
proc cpanel data = mycas.psid;
  id id t;
  model lwage = wks south smsa ms exp exp2 occ
              ind union fem blk ed / amacurdy;
  correlated wks ms exp exp2 union ed;
run;
```

The results are shown in Output 4.3.3. Little is changed from the Hausman-Taylor model. The Hausman test compares the Amemiya-MaCurdy model to the Hausman-Taylor model (not the fixed-effects model as previously) and shows that the one additional assumption is acceptable. You even gained a bit of efficiency in the process; compare the standard deviations of the coefficient on the variable ED from both models.

### Output 4.3.3 Amemiya-MaCurdy Estimation

#### The CPANEL Procedure

Amemiya and MaCurdy Model for Correlated Individual Effects

Dependent Variable: lwage (Log of wages)

<table>
<thead>
<tr>
<th>Variance Component Estimates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
</tr>
<tr>
<td>Cross Sections</td>
</tr>
<tr>
<td>Error</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hausman Test vs. Hausman-Taylor Coefficients</th>
<th>DF</th>
<th>m Value</th>
<th>Pr &gt; m</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>13</td>
<td>14.67</td>
<td>0.3287</td>
</tr>
</tbody>
</table>
Finally, you should realize that the Hausman-Taylor and Amemiya-MaCurdy estimators are not cure-alls for correlated individual effects. Estimation tacitly relies on the uncorrelated regressors being sufficient to predict the correlated regressors. Otherwise, you run into the problem of weak instruments. If you have weak instruments, you will obtain biased estimates that have very large standard errors. However, that does not seem to be the case here.

Example 4.4: Crime Rates in North Carolina: Instrumental Variables Regression

Cornwell and Trumbull (1994) examined data on crime rates from 90 counties in North Carolina and collected data each year from 1981 to 1987. The following statements create the Crime data table. These statements assume that your CAS engine libref is named mycas, but you can substitute any appropriately defined CAS engine libref.

data mycas.Crime;
  input county year west central urban d82 d83 d84 d85 d86 d87
  lcrmrte lprbarr lprbconv lprbpris lavgsen lpolpc
  ldensity ltaxpc lwcon lwtrd lwfir lwser lwmfg
  lwfed lwsta lwloc lmix lpctymle lpctmin;
  label county = 'county identifier'
  year = '81 to 87'
  west = '1 if in western N.C.'
  central = '1 if in central N.C.'
  urban = '1 if in SMSA'

... more lines ...
The variables are too numerous to describe individually, but they are fully labeled in the regression output and are summarized as follows:

- The panel ID variables are County and Year.
- The dependent variable, LCrMrte, is the log of the crime rate (number of crimes divided by total population).
- The variables West, Central, and Urban provide geographical background for each county.
- The variables D82 through D87 are dummy variables for year effects.
- The variables with the LPrb prefix are log-probabilities of police or legal actions. For example, LPrbArr is the log-probability of an arrest given that a crime was committed.
- The variables with the LW prefix are log weekly wages for various occupations and industries.
- Other variables that begin with the letter L are log-transformed. For example, the variable LPolPC is the log of police per capita.

Cornwell and Trumbull (1994) fit a fixed-effects model for the log of crime rate, with the log-probability of arrest (LPrbArr), and the log of police per capita (LPolPC) among the regressors. They noted that these two regressors were endogenous because of the following causality loops:

- Although police presence might decrease the crime rate because of deterrence (or increase the crime rate because of better reporting), an increase in crime might cause a desire for more police presence.
- Although the crime rate might fall if more crimes resulted in arrest, a rising crime rate could cause the probability of arrest to decrease because police resources would be spread thin.

To deal with the endogeneity, Cornwell and Trumbull (1994) fit a two-stage least squares (2SLS) model with county-level fixed effects, with two external instruments: LTaxPC (log tax revenue per capita) and LMix (log ratio of crimes that involve face-to-face contact). Their rationale for using these instruments is that counties with higher tax revenues have more police presence and that face-to-face crimes have a higher probability of arrest because victims could identify their offenders.

Baltagi (2006) proposed a similar 2SLS model, but with random effects instead of fixed effects for counties. The following statements recreate that estimation:

```
proc cpanel data = mycas.crime;
   id county year;
   model lcrmrte = lprbarr lpolpc lprbconv lprbpris lavgsen ldensity
                   lwcon lwtuc lwtrd lwfir lwser lwmfg lwfed lwsta
                   lwloc lpctyme lpctmin west central urban
                   d82 d83 d84 d85 d86 d87 / ivranone;
   endogenous lprbarr lpolpc;
   instruments ltaxpc lmix;
   run;
```

The results are shown in Output 4.4.1. The output includes a “Structural Summary” table, which lists the endogenous variables and the instrumental variables. The list of instrumental variables includes both the specified external instruments (variables LTaxPC and LMix) and the exogenous regression variables.
# Output 4.4.1 Crime Rates in North Carolina, IV Random Effects

## The CPANEL Procedure

**EC2SLS One-Way Random Effects (IVRanOne)**  
Dependent Variable: lcrrtce (log crimes committed per person)

<table>
<thead>
<tr>
<th>Model Description</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimation Method</td>
<td>IVRanOne</td>
</tr>
<tr>
<td>Data Set</td>
<td>CRIME</td>
</tr>
<tr>
<td>Number of Observations</td>
<td>630</td>
</tr>
<tr>
<td>Number of Cross Sections</td>
<td>90</td>
</tr>
<tr>
<td>Time Series Length</td>
<td>7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fit Statistics</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SSE</td>
<td>12.1250</td>
</tr>
<tr>
<td>DFE</td>
<td>603</td>
</tr>
<tr>
<td>MSE</td>
<td>0.0201</td>
</tr>
<tr>
<td>Root MSE</td>
<td>0.1418</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variance Component Estimates</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>Variance</td>
</tr>
<tr>
<td>Cross Sections</td>
<td>0.046036</td>
</tr>
<tr>
<td>Error</td>
<td>0.022272</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hausman Test For Random Effects</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Coefficients</td>
<td>22</td>
</tr>
<tr>
<td>DF</td>
<td>22</td>
</tr>
<tr>
<td>m Value</td>
<td>19.50</td>
</tr>
<tr>
<td>Pr &gt; m</td>
<td>0.6140</td>
</tr>
</tbody>
</table>

## Structural Summary

<table>
<thead>
<tr>
<th>Endogenous Variables</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Iprbarr</td>
<td></td>
</tr>
<tr>
<td>Ipolpc</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Instruments</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Iprbconv</td>
<td></td>
</tr>
<tr>
<td>Iprbpris</td>
<td></td>
</tr>
<tr>
<td>lavsgen</td>
<td></td>
</tr>
<tr>
<td>Idensity</td>
<td></td>
</tr>
<tr>
<td>lwcon</td>
<td></td>
</tr>
<tr>
<td>lwroc</td>
<td></td>
</tr>
<tr>
<td>lwtrd</td>
<td></td>
</tr>
<tr>
<td>lwfrw</td>
<td></td>
</tr>
<tr>
<td>lwser</td>
<td></td>
</tr>
<tr>
<td>lwmgf</td>
<td></td>
</tr>
<tr>
<td>lwfed</td>
<td></td>
</tr>
<tr>
<td>lwsta</td>
<td></td>
</tr>
<tr>
<td>lwloc</td>
<td></td>
</tr>
<tr>
<td>lpctyme</td>
<td></td>
</tr>
<tr>
<td>lpctmin</td>
<td></td>
</tr>
<tr>
<td>west</td>
<td></td>
</tr>
<tr>
<td>central</td>
<td></td>
</tr>
<tr>
<td>urban</td>
<td></td>
</tr>
<tr>
<td>d82</td>
<td></td>
</tr>
<tr>
<td>d83</td>
<td></td>
</tr>
<tr>
<td>d84</td>
<td></td>
</tr>
<tr>
<td>d85</td>
<td></td>
</tr>
<tr>
<td>d86</td>
<td></td>
</tr>
<tr>
<td>d87</td>
<td></td>
</tr>
<tr>
<td>ilxpc</td>
<td></td>
</tr>
<tr>
<td>limix</td>
<td></td>
</tr>
</tbody>
</table>
Cornwell and Trumbull (1994) favored a fixed-effects approach because a Hausman specification test rejected the null hypothesis of random effects. Baltagi (2006) argued that the Hausman test used by Cornwell and Trumbull (1994) was inaccurate because it was based on standard methodology that did not account for endogeneity in the regressors.

Baltagi (2006) proposed a modified 2SLS Hausman test, and the results of that test are provided in the “Hausman Test For Random Effects” table in the output. The results favor the random-effects specification.

A higher probability of arrest is shown to diminish crime, and police presence increases the crime rate because more crimes are reported.
References


# Chapter 5
The CQLIM Procedure

## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview: CQLIM Procedure</td>
<td>172</td>
</tr>
<tr>
<td>PROC CQLIM Features</td>
<td>172</td>
</tr>
<tr>
<td>PROC CQLIM Compared with Other SAS Procedures</td>
<td>173</td>
</tr>
<tr>
<td>Getting Started: CQLIM Procedure</td>
<td>174</td>
</tr>
<tr>
<td>Syntax: CQLIM Procedure</td>
<td>176</td>
</tr>
<tr>
<td>Functional Summary</td>
<td>176</td>
</tr>
<tr>
<td>PROC CQLIM Statement</td>
<td>178</td>
</tr>
<tr>
<td>BOUNDS Statement</td>
<td>179</td>
</tr>
<tr>
<td>BY Statement</td>
<td>179</td>
</tr>
<tr>
<td>ENDOGENOUS Statement</td>
<td>179</td>
</tr>
<tr>
<td>FREQ Statement</td>
<td>181</td>
</tr>
<tr>
<td>HETERO Statement</td>
<td>181</td>
</tr>
<tr>
<td>INIT Statement</td>
<td>182</td>
</tr>
<tr>
<td>MODEL Statement</td>
<td>183</td>
</tr>
<tr>
<td>OUTPUT Statement</td>
<td>185</td>
</tr>
<tr>
<td>RESTRICT Statement</td>
<td>186</td>
</tr>
<tr>
<td>TEST Statement</td>
<td>186</td>
</tr>
<tr>
<td>WEIGHT Statement</td>
<td>187</td>
</tr>
<tr>
<td>Details: CQLIM Procedure</td>
<td>188</td>
</tr>
<tr>
<td>Ordinal Discrete Choice Modeling</td>
<td>188</td>
</tr>
<tr>
<td>Limited Dependent Variable Models</td>
<td>189</td>
</tr>
<tr>
<td>Stochastic Frontier Production and Cost Models</td>
<td>190</td>
</tr>
<tr>
<td>Heteroscedasticity</td>
<td>191</td>
</tr>
<tr>
<td>Tests on Parameters</td>
<td>192</td>
</tr>
<tr>
<td>Output to SAS Data Set</td>
<td>193</td>
</tr>
<tr>
<td>Naming</td>
<td>197</td>
</tr>
<tr>
<td>ODS Table Names</td>
<td>197</td>
</tr>
<tr>
<td>Examples: The CQLIM Procedure</td>
<td>198</td>
</tr>
<tr>
<td>Example 5.1: Model with Censoring</td>
<td>198</td>
</tr>
<tr>
<td>References</td>
<td>200</td>
</tr>
</tbody>
</table>
Overview: CQLIM Procedure

The CQLIM procedure is a version of the QLIM procedure in SAS/ETS software that requires SAS Cloud Analytic Services in order to run. Both procedures analyze univariate limited dependent variable models in which dependent variables take discrete values or are observed within only a limited range of values. Unlike the QLIM procedure, which can be run only on a single workstation, the CQLIM procedure takes advantage of a distributed computing environment that enables it to distribute the optimization task to one or more nodes. In addition, each node can use one or more threads to perform the optimization on its subset of the data. When several nodes are used and each node uses several threads to carry out its part of the work, the result is a highly parallel computation that can provide a dramatic gain in performance.

The CQLIM procedure uses the maximum likelihood method and, by default, uses multiple threads to perform computations.

The CQLIM procedure is similar to the QLIM procedure in SAS/ETS. For example, the standard model with censoring or truncation is estimated by specifying the endogenous variable to be truncated or censored. When the data are limited by specific values or variables, the limits of the dependent variable can be specified using two options in two statements.

Another class of models that belongs to the limited depended variable case is the Stochastic Frontier analysis. These models were first introduced by Aigner, Lovell, and Schmidt (1977); Meeusen and van den Broeck (1977) and are capable of analyzing technical inefficiencies within the production or cost inefficiencies within the costs.

The CQLIM procedure is also capable of modeling problems where the dependent variable take discrete values. For these cases you can either use the LOGIT model or the PROBIT model. In both cases the dependent variable is modeled with a binary distribution (typically $y = 0, 1$). The main difference is that the LOGIT is characterized by an underlying logistic distribution while the PROBIT consider a Gaussian distribution. When the dependent variable takes more than two levels, the binary assumption is no longer valid. In these cases you can use the ordered LOGIT regression and the ordered PROBIT regression.

PROC CQLIM Features

The CQLIM procedure supports the following models:

- linear regression models with heteroscedasticity
- probit models with heteroscedasticity
- logit models with heteroscedasticity
- Tobit models (censored and truncated) with heteroscedasticity
- stochastic frontier production and cost models

In linear regression models with heteroscedasticity, the assumption that error variance is constant across observations is relaxed. The CQLIM procedure allows for a number of different linear and nonlinear variance specifications.
PROC CQLIM also offers a class of models in which the dependent variable is censored or truncated from below, above, or both. When a continuous dependent variable is observed within only a certain range, and values outside this range are not available, PROC CQLIM offers a class of models that adjust for truncation. In some cases, the dependent variable is continuous within only a certain range, and all values outside this range are reported as being on its boundary. For example, if it is not possible to observe negative values, the value of the dependent variable is reported as equal to 0. Because the data are censored, ordinary least squares (OLS) results are inconsistent, and you cannot be certain that the predicted values from the model will fall in the appropriate region.

The stochastic frontier production and cost models available in PROC CQLIM allow for random shocks of the production or cost. They include a systematic positive component in the error term that adjusts for technical or cost inefficiency.

Initial starting values for the nonlinear optimizations are usually calculated by OLS.

---

**PROC CQLIM Compared with Other SAS Procedures**

The CQLIM procedure provides limited dependent data modeling functionality comparable to that of the HPQLIM and QLIM procedures in SAS/ETS software.

**PROC CQLIM Compared with the HPQLIM Procedure**

The functionality of the CQLIM procedure closely resembles that of the HPQLIM procedure, which is a high-performance procedure. The CQLIM procedure is the next generation of the HPQLIM 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 on multiple machines.

The CQLIM procedure support maximum likelihood estimation of all the models available in PROC HPQLIM.

**PROC CQLIM Compared with the QLIM Procedure**

The CQLIM procedure is specifically designed to operate in SAS Viya and performs computations in multiple threads on multiple machines. The capability of the QLIM procedure to perform computations in a distributed environment is limited to the parallel implementation of MCMC sampling algorithms for Bayesian analysis.

This release of the CQLIM procedure contains some, but not all, of the functionality present in the QLIM procedure. The QLIM procedure provides the following features, which are not present in the CQLIM procedure:

- multivariate modeling
- variable selection
- Bayesian analysis
Chapter 5: The CQLIM Procedure

Getting Started: CQLIM Procedure

This example illustrates the use of the CQLIM procedure. The data were originally published by Mroz (1987), and the following DATA steps load a subset of the data into CAS. The assumption here is that your CAS engine libref is named mycas, but you can substitute any appropriately defined CAS engine libref.

```
title1 'Estimating a Tobit Model';
data subset;
  input Hours Yrs_Ed Yrs_Exp @@;
  if Hours eq 0 then Lower=.;
  else Lower=Hours;
datalines;
0 8 9 0 8 12 0 9 10 0 10 15 0 11 4 0 11 6
1000 12 1 1960 12 29 0 13 3 2100 13 36
3686 14 11 1920 14 38 0 15 14 1728 16 3
1568 16 19 1316 17 7 0 17 15
;
%*; options cashost="rdcgrd051" casport=30002;
%*; options casuser=chmaca;
%*; libname mycas sasioca;

data mycas.subset(copies=1 promote=yes);
  set subset;
run;
```

In these data, `Hours` is the number of hours that a wife worked outside the household in a particular year, `Yrs_Ed` is years of education, and `Yrs_Exp` is years of work experience.

From the nature of the data, it is clear that there are a number of women who committed some positive number of hours to outside work ($y_i > 0$ is observed). There are also a number of women who did not work outside the household at all ($y_i = 0$ is observed). This yields the following model,

$$y_i^* = x_i^\prime \beta + \epsilon_i$$

$$y_i = \begin{cases} 
y_i^* & \text{if } y_i^* > 0 \\
0 & \text{if } y_i^* \leq 0
\end{cases}$$

where $\epsilon_i \overset{iid}{\sim} N(0, \sigma^2)$ and $x_i$ denotes the set of explanatory variables. The following statements fit a Tobit model to the number of hours worked, with years of education and years of work experience as covariates:

```
/*--- Tobit Model ---*/
proc cqlim data=mycas.subset;
  model hours = yrs_ed yrs_exp;
  endogenous hours ~ censored(lb=0);
run;
```

The output of the CQLIM procedure is shown in Output 5.1.
Figure 5.1 Tobit Analysis Results

Estimating a Tobit Model

The CQLIM Procedure

<table>
<thead>
<tr>
<th>Model Fit Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution Mode</td>
</tr>
<tr>
<td>Dependent Variable</td>
</tr>
<tr>
<td>Number of Observations</td>
</tr>
<tr>
<td>Data Set</td>
</tr>
<tr>
<td>Model</td>
</tr>
<tr>
<td>Optimizer</td>
</tr>
<tr>
<td>Log Likelihood</td>
</tr>
<tr>
<td>Maximum Absolute Gradient</td>
</tr>
<tr>
<td>Number of Iterations</td>
</tr>
<tr>
<td>Optimization Method</td>
</tr>
<tr>
<td>AIC</td>
</tr>
<tr>
<td>SBC</td>
</tr>
<tr>
<td>Covariance Estimation</td>
</tr>
</tbody>
</table>

Convergence criterion (ABSGCONV=0.00001) satisfied.

<table>
<thead>
<tr>
<th>Parameter Estimates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter</td>
</tr>
<tr>
<td>Intercept</td>
</tr>
<tr>
<td>Yrs_Ed</td>
</tr>
<tr>
<td>Yrs_Exp</td>
</tr>
<tr>
<td>_Sigma</td>
</tr>
</tbody>
</table>

The “Parameter Estimates” table contains four rows. The first three rows correspond to the vector estimate of the regression coefficients $\hat{\beta}$. The last row is called _Sigma, which corresponds to the estimate of the error variance $\sigma$. 
Syntax: CQLIM Procedure

The following statements are available in the CQLIM procedure:

```
PROC CQLIM options ;
   BOUNDS bound1 < , bound2 . . . > ;
   BY variables ;
   FREQ variable ;
   ENDOGENOUS variables ~ options ;
   HETERO dependent-variables ~ exogenous-variables / options ;
   INIT initvalue1 < , initvalue2 . . . > ;
   MODEL dependent-variables = regressors / options ;
   OUTPUT OUT=SAS-data-set < output-options > ;
   RESTRICT restriction1 < , restriction2 . . . > ;
   TEST options ;
   WEIGHT variable < / option > ;
```

One MODEL statement is required. If you specify a FREQ or WEIGHT statement more than once, only the variable that is specified in the first instance is used.

Functional Summary

Table 5.1 summarizes the statements and options available in the CQLIM procedure.

<table>
<thead>
<tr>
<th>Description</th>
<th>Statement</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data Set Options</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specifies the input data set</td>
<td>PROC CQLIM</td>
<td>DATA=</td>
</tr>
<tr>
<td>Writes predictions to an output data set</td>
<td>OUTPUT</td>
<td>OUT=</td>
</tr>
<tr>
<td><strong>Declaring the Role of Variables</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specifies BY-group processing</td>
<td>BY</td>
<td></td>
</tr>
<tr>
<td>Specifies a frequency variable</td>
<td>FREQ</td>
<td></td>
</tr>
<tr>
<td>Specifies a weight variable</td>
<td>WEIGHT</td>
<td>NONNORMALIZE</td>
</tr>
<tr>
<td><strong>Printing Control Options</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prints the correlation matrix of the estimates</td>
<td>PROC CQLIM</td>
<td>CORRB</td>
</tr>
<tr>
<td>Prints the covariance matrix of the estimates</td>
<td>PROC CQLIM</td>
<td>COVB</td>
</tr>
<tr>
<td>Prints a summary iteration listing</td>
<td>PROC CQLIM</td>
<td>ITPRINT</td>
</tr>
<tr>
<td><strong>Optimization Process Control Options</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sets boundary restrictions on parameters</td>
<td>BOUNDS</td>
<td></td>
</tr>
<tr>
<td>Sets initial values for parameters</td>
<td>INIT</td>
<td></td>
</tr>
<tr>
<td>Sets linear restrictions on parameters</td>
<td>RESTRICT</td>
<td></td>
</tr>
</tbody>
</table>
### Table 5.1 continued

<table>
<thead>
<tr>
<th>Description</th>
<th>Statement</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Model Estimation Options</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Suppresses the intercept parameter</td>
<td>MODEL</td>
<td>NOINT</td>
</tr>
<tr>
<td>Specifies the method to calculate parameter covariance</td>
<td>PROC CQLIM</td>
<td>COVEST=</td>
</tr>
<tr>
<td><strong>Endogenous Variable Options</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specifies a discrete variable</td>
<td>ENDOGENOUS</td>
<td>DISCRETE()</td>
</tr>
<tr>
<td>Specifies a censored variable</td>
<td>ENDOGENOUS</td>
<td>CENSORED()</td>
</tr>
<tr>
<td>Specifies a truncated variable</td>
<td>ENDOGENOUS</td>
<td>TRUNCATED()</td>
</tr>
<tr>
<td>Specifies a stochastic frontier variable</td>
<td>ENDOGENOUS</td>
<td>FRONTIER()</td>
</tr>
<tr>
<td><strong>Heteroscedasticity Model Options</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specifies the function for heteroscedasticity models</td>
<td>HETERO</td>
<td>LINK=</td>
</tr>
<tr>
<td>Squares the function for heteroscedasticity models</td>
<td>HETERO</td>
<td>SQUARE</td>
</tr>
<tr>
<td>Specifies no constant for heteroscedasticity models</td>
<td>HETERO</td>
<td>NOCONST</td>
</tr>
<tr>
<td><strong>Output Control Options</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Outputs the predicted values</td>
<td>OUTPUT</td>
<td>PREDICTED</td>
</tr>
<tr>
<td>Outputs the structured part</td>
<td>OUTPUT</td>
<td>XBETA</td>
</tr>
<tr>
<td>Outputs the residuals</td>
<td>OUTPUT</td>
<td>RESIDUAL</td>
</tr>
<tr>
<td>Outputs the error standard deviation</td>
<td>OUTPUT</td>
<td>ERRSTD</td>
</tr>
<tr>
<td>Outputs the marginal effects</td>
<td>OUTPUT</td>
<td>MARGINAL</td>
</tr>
<tr>
<td>Outputs the probability for the current response</td>
<td>OUTPUT</td>
<td>PROB</td>
</tr>
<tr>
<td>Outputs the probability for all responses</td>
<td>OUTPUT</td>
<td>PROBALL</td>
</tr>
<tr>
<td>Outputs the expected value</td>
<td>OUTPUT</td>
<td>EXPECTED</td>
</tr>
<tr>
<td>Outputs the conditional expected value</td>
<td>OUTPUT</td>
<td>CONDITIONAL</td>
</tr>
<tr>
<td>Outputs the inverse Mills ratio</td>
<td>OUTPUT</td>
<td>MILLS</td>
</tr>
<tr>
<td>Outputs the technical efficiency measures</td>
<td>OUTPUT</td>
<td>TE1</td>
</tr>
<tr>
<td></td>
<td>OUTPUT</td>
<td>TE2</td>
</tr>
<tr>
<td><strong>Test Request Options</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Requests Wald, Lagrange multiplier, and likelihood ratio tests</td>
<td>TEST</td>
<td>ALL</td>
</tr>
<tr>
<td>Requests the Wald test</td>
<td>TEST</td>
<td>WALD</td>
</tr>
<tr>
<td>Requests the Lagrange multiplier test</td>
<td>TEST</td>
<td>LM</td>
</tr>
<tr>
<td>Requests the likelihood ratio test</td>
<td>TEST</td>
<td>LR</td>
</tr>
</tbody>
</table>
**PROC CQLIM Statement**

```
PROC CQLIM options ;
```

The PROC CQLIM statement invokes the CQLIM procedure. You can specify the following *options*.

**Data Set Options**

```
DATA=SAS-data-set
```

specifies the input SAS data set. If you omit this option, PROC CQLIM uses the most recently created SAS data set.

**Printing Options**

```
CORRB
```

prints the correlation matrix of the parameter estimates.

```
COVB
```

prints the covariance matrix of the parameter estimates.

```
ITPRINT
```

prints the initial parameter estimates, convergence criteria, and all constraints of the optimization. At each iteration, the objective function value, step size, maximum gradient, and slope of search direction are also printed.

**Model Estimation Options**

```
COVEST=HESSIAN | OP | QML
```

specifies the method to use for calculating the covariance matrix of parameter estimates. You can specify the following covariance options:

```
HESSIAN
```

specifies the covariance from the inverse Hessian matrix.

```
OP
```

specifies the covariance from the outer product matrix.

```
QML
```

specifies the covariance from the outer product and Hessian matrices (the quasi-maximum likelihood estimates).

By default, COVEST=HESSIAN.
BOUNDSS Statement

BOUNDSS bound1 < , bound2 . . . > ;

The BOUNDS statement imposes simple boundary constraints on the parameter estimates. BOUNDS statement constraints refer to the parameters that are estimated by the CQLIM procedure. You can specify any number of BOUNDS statements.

Each bound is composed of parameters, constants, and inequality operators. Parameters that are associated with regressor variables are referred to by the names of the corresponding regressor variables. Specify each bound as follows:

item operator item < operator item < operator item . . . >

Each item is a constant, the name of a parameter, or a list of parameter names. For more information about how parameters are named in the CQLIM procedure, see the section “Naming of Parameters” on page 197. Each operator is <, >, <=, or >=.

You can use both the BOUNDS statement and the RESTRICT statement to impose boundary constraints; however, the BOUNDS statement provides a simpler syntax for specifying these types of constraints. For more information, see the section “RESTRICT Statement” on page 186.

The following BOUNDS statement constrains the estimates of the parameters that are associated with the variable ttime and the variables x1 through x10 to be between 0 and 1. The following example illustrates the use of parameter lists to specify boundary constraints:

bounds 0 < ttime x1-x10 < 1;

The following BOUNDS statement constrains the estimates of the correlation (_RHO) and sigma (_SIGMA) in the bivariate model:

bounds _rho >= 0, _sigma.y1 > 1, _sigma.y2 < 5;

BY Statement

BY variables ;

A BY statement can be used in PROC CQLIM to obtain separate analyses of observations in groups defined by the BY variables.

ENDOGENOUS Statement

ENDOGENOUS variables ~ options ;

The ENDOGENOUS statement specifies the type of dependent variables that appear on the left-hand side of the equation. Currently, PROC CQLIM does not handle right-hand-side endogeneity. All variables that appear on the right-hand side of the equation are treated as exogenous. You can specify the following options.
Discrete Variable Options

\texttt{DISCRETE \textless\texttt{ (discrete-options \textgreater)}}
specifies that the endogenous variables in this statement be discrete. You can specify the following \texttt{discrete-options}:

\texttt{DISTRIBUTION=LOGISTIC | NORMAL}
\texttt{DIST=LOGISTIC | NORMAL}
\texttt{D=LOGISTIC | NORMAL}

specifies the cumulative distribution function to use to model the response probabilities. You can specify the following distribution types:

\texttt{LOGISTIC} specifies the logistic distribution for the logit model.
\texttt{NORMAL} specifies the normal distribution for the probit model.

By default, \texttt{DISTRIBUTION=\texttt{NORMAL}}.

Censored Variable Options

\texttt{CENSORED \texttt{ (censored-options \textgreater)}}
censors the endogenous variables in this statement. You can specify the following \texttt{censored-options}:

\texttt{LB=value | variable}
\texttt{LOWERBOUND=value | variable}

specifies the lower bound of the censored variables. If \texttt{value} is missing or the value of \texttt{variable} is missing, no lower bound is set. By default, no lower bound is set.

\texttt{UB=value | variable}
\texttt{UPPERBOUND=value | variable}

specifies the upper bound of the censored variables. If \texttt{value} is missing or the value of \texttt{variable} is missing, no upper bound is set. By default, no upper bound is set.

Truncated Variable Options

\texttt{TRUNCATED \texttt{ (truncated-options \textgreater)}}
truncates the endogenous variables in the statement. You can specify the following \texttt{truncated-options}:

\texttt{LB=value | variable}
\texttt{LOWERBOUND=value | variable}

specifies the lower bound of the truncated variables. If \texttt{value} is missing or the value of \texttt{variable} is missing, no lower bound is set. By default, no lower bound is set.

\texttt{UB=value | variable}
\texttt{UPPERBOUND=value | variable}

specifies the upper bound of the truncated variables. If \texttt{value} is missing or the value of \texttt{variable} is missing, no upper bound is set. By default, no upper bound is set.
Stochastic Frontier Variable Options

FRONTIER \langle \text{frontier-options} \rangle

specifies a stochastic frontier model. You can specify the following frontier-options:

COST
specifies that the estimated model be a cost function.

PRODUCTION
specifies that the estimated model be a production function.

TYPE=EXPONENTIAL | HALF | TRUNCATED
specifies the model type. You can specify the following types:

EXPONENTIAL specifies an exponential model.
HALF specifies a half-normal model.
TRUNCATED specifies a truncated normal model.

If neither PRODUCTION nor COST is specified, a production function is estimated by default.

FREQ Statement

FREQ variable ;

The FREQ statement identifies a variable that contains the frequency of occurrence of each observation. PROC CQLIM treats each observation as if it appeared \( n \) times, where \( n \) is the value of the FREQ variable for the observation. If the frequency value is not an integer, it is truncated to an integer. If the frequency value is less than 1 or missing, the observation is not used in the model fitting. When the FREQ statement is not specified, each observation is assigned a frequency of 1. If you specify more than one FREQ statement, then only the first FREQ statement is used.

HETERO Statement

HETERO dependent-variables \sim \text{exogenous-variables} </options> ;

The HETERO statement specifies variables that are related to the heteroscedasticity of the residuals and the way that these variables are used to model the error variance. PROC CQLIM supports the following heteroscedastic regression model:

\[ y_i = x_i' \beta + \epsilon_i \]
\[ \epsilon_i \sim N(0, \sigma_i^2) \]

For more information about the specification of functional forms, see the section “Heteroscedasticity” on page 191. The following options specify the functional forms of heteroscedasticity:
LINK=EXP | LINEAR
specifies the functional form. You can specify the following values:

EXP specifies the exponential link function:
\[
\sigma_i^2 = \sigma^2 (1 + \exp(z_i' \gamma))
\]

LINEAR specifies the linear link function:
\[
\sigma_i^2 = \sigma^2 (1 + z_i' \gamma)
\]

By default, LINK=EXP.

NOCONST
specifies that no constant appear in the exponential heteroscedasticity model:
\[
\sigma_i^2 = \sigma^2 \exp(z_i' \gamma)
\]

SQUARE
estimates the model by using the square of the linear heteroscedasticity function. For example, you can specify the heteroscedasticity function
\[
\sigma_i^2 = \sigma^2 (1 + (z_i' \gamma)^2)
\]
by using the following code:

```plaintext
model y = x1 x2 / censored(lb=0);
  hetero y ~ z1 / link=linear square;
```

The SQUARE option does not apply to the exponential heteroscedasticity function, because the square of an exponential function of \(z_i' \gamma\) is the same as the exponential of \(2z_i' \gamma\). Hence, the only difference is that all \(\gamma\) estimates are divided by two.

This option is ignored if you do not specify the LINK= option.

---

**INIT Statement**

**INIT initvalue1 < , initvalue2 ... > ;**

The INIT statement sets initial values for parameters in the optimization. You can specify any number of INIT statements.

Each *initvalue* is written as a parameter or parameter list, followed by an optional equality operator (=), followed by a number:

*parameter <= number*
MODEL Statement

MODEL dependent-variables<(order-options)> = regressors< / options> ;

The MODEL statement specifies the dependent variable and independent regressor variables for the regression model. For binary LOGIT and binary PROBIT models you can specify the following order-options:

ORDER= FORMATTED | FREQ | INTERNAL

specifies the sort order for the levels of the discrete variables that you specify in the ENDOGENOUS statement. The sort order determines which parameters in the model correspond to each level in the data. You can specify the following sort orders:

- **FORMATTED** sorts levels by formatted value. The sort order is machine-dependent.
- **FREQ** sorts levels by descending frequency count; levels that have the most observations come first in the order.
- **INTERNAL** sorts levels by unformatted value. The sort order is machine-dependent.

By default, ORDER=FORMATTED. For more information about sort order, see the chapter on the SORT procedure in the Base SAS Procedures Guide.

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

- **NOINT** suppresses the intercept parameter.

You can also specify the following endogenous variable options, which are the same as the options in the ENDOGENOUS statement. If you specify an endogenous variable option in both the MODEL statement and the ENDOGENOUS statement, only the option in the ENDOGENOUS statement is used.

Discrete Variable Options

- **DISCRETE < (discrete-options) >** specifies that the endogenous variables in the MODEL statement be discrete. You can specify the following discrete-options:

  - DISTRIBUTION=LOGISTIC | NORMAL
  - DIST=LOGISTIC | NORMAL
  - D=LOGISTIC | NORMAL

specifies the cumulative distribution function to use to model the response probabilities. You can specify the following distribution types:

- **LOGISTIC** specifies the logistic distribution for the logit model.
- **NORMAL** specifies the normal distribution for the probit model.

By default, DISTRIBUTION=NORMAL.
Censored Variable Options

CENSORED < (censored-options ) >
censors the endogenous variables in this statement. You can specify the following censored-options:

LB=value | variable
LOWERBOUND=value | variable
specifies the lower bound of the censored variables. If value is missing or the value of variable is missing, no lower bound is set. By default, no lower bound is set.

UB=value | variable
UPPERBOUND=value | variable
specifies the upper bound of the censored variables. If value is missing or the value of variable is missing, no upper bound is set. By default, no upper bound is set.

Truncated Variable Options

TRUNCATED < (truncated-options ) >
truncates the endogenous variables in this statement. You can specify the following truncated-options:

LB=value | variable
LOWERBOUND=value | variable
specifies the lower bound of the truncated variables. If value is missing or the value of variable is missing, no lower bound is set. By default, no lower bound is set.

UB=value | variable
UPPERBOUND=value | variable
specifies the upper bound of the truncated variables. If value is missing or the value of variable is missing, no upper bound is set. By default, no upper bound is set.

Stochastic Frontier Variable Options

FRONTIER < (frontier-options ) >
specifies the stochastic frontier model. You can specify the following frontier-options:

COST
specifies that the estimated model be a cost function.

PRODUCTION
specifies that the estimated model be a production function.

TYPE=EXPONENTIAL | HALF | TRUNCATED
specifies the model type.

EXPONENTIAL
specifies an exponential model.

HALF
specifies a half-normal model.

TRUNCATED
specifies a truncated normal model.

If neither PRODUCTION nor COST is specified, a production function is estimated by default.
The OUTPUT statement creates a new SAS data set to contain variables that you specify using the COPYVAR option and the following quantities of interest if they are specified as output-options: estimates of $x'\beta$, predicted value, residual, marginal effects, probability, standard deviation of the error, expected value, conditional expected value, technical efficiency measures, and inverse Mills ratio. When the response values are missing for an observation, all output estimates except the residual for that observation are still computed as long as none of the explanatory variables are missing. This enables you to compute these statistics for prediction. You can specify only one OUTPUT statement.

You must specify the OUT= option:

```
OUT=SAS-data-set < output-options > ;
```

```
You can specify one or more of the following output-options:
```

**CONDITIONAL**
outputs estimates of conditional expected values of continuous endogenous variables.

**COPYVAR=SAS-variable-names**
**COPYVARS=(SAS-variable-names)**
adds any SAS variables that appear in the input data set to the output data set.

**ERRSTD**
outputs estimates of $\sigma_j$, the standard deviation of the error term.

**EXPECTED**
outputs estimates of expected values of continuous endogenous variables.

**MARGINAL**
outputs marginal effects.

**MILLS**
outputs estimates of inverse Mills ratios of censored or truncated continuous variables and binary discrete variables.

**PREDICTED**
outputs estimates of predicted endogenous variables.

**PROB**
outputs estimates of probability of discrete endogenous variables that take the current observed responses.

**PROBALL**
outputs estimates of probability of discrete endogenous variables for all possible responses.
RESIDUAL
outputs estimates of residuals of continuous endogenous variables.

XBETA
outputs estimates of $x'\beta$.

TE1
outputs estimates of technical efficiency for each producer in the stochastic frontier model that is suggested by Battese and Coelli (1988).

TE2
outputs estimates of technical efficiency for each producer in the stochastic frontier model that is suggested by Jondrow et al. (1982).

RESTRICT Statement

RESTRICT restriction1 < , restriction2 . . . > ;

The RESTRICT statement imposes linear restrictions on the parameter estimates. You can specify any number of RESTRICT statements, but the number of restrictions that are imposed is limited by the number of regressors.

Each restriction is written as an expression, 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.

Restriction expressions can be composed of parameter names; multiplication (*), addition (+), and subtraction (−) operators; and constants. Parameters that are named in restriction expressions must be among the parameters that are estimated by the model. Parameters that are associated with a regressor variable are referred to by the name of the corresponding regressor variable. The restriction expressions must be a linear function of the parameters.

The following statements illustrate the use of the RESTRICT statement:

```proc cqlim data=mycas.dataset;
  model y = x1-x10 / censored(lb=0);
  restrict x1*2 <= x2 + x3;
run;```

TEST Statement

<"label":> TEST <"string">: equation < , equation . . . > / options ;

The TEST statement performs Wald, Lagrange multiplier, and likelihood ratio tests of linear hypotheses about the regression parameters in the preceding MODEL statement. Each equation specifies a linear hypothesis to be tested. All hypotheses in one TEST statement are tested jointly. Variable names in the equations must correspond to regressors in the preceding MODEL statement, and each name represents the coefficient of the corresponding regressor. Use the keyword INTERCEPT for a test that includes a constant.
You can specify the following options after the slash (/):

- **ALL**
  requests Wald, Lagrange multiplier, and likelihood ratio tests.

- **LM**
  requests the Lagrange multiplier test.

- **LR**
  requests the likelihood ratio test.

- **WALD**
  requests the Wald test.

The following statements illustrate the use of the TEST statement (note the use of the INTERCEPT keyword in the second TEST statement):

```plaintext
proc cqlim;
  model y = x1 x2 x3;
  test x1 = 0, x2 * .5 + 2 * x3 = 0;
  test_int: test intercept = 0, x3 = 0;
run;
```

The first TEST statement investigates the joint hypothesis that

\[ \hat{\beta}_1 = 0 \]

and

\[ 0.5\hat{\beta}_2 + 2\hat{\beta}_3 = 0 \]

Only linear equality restrictions and tests are permitted in PROC CQLIM. Test expressions can be composed only of algebraic operations that involve the addition symbol (+), subtraction symbol (−), and multiplication symbol (*).

The TEST statement accepts labels that are reproduced in the printed output. You can label a TEST statement in two ways: you can specify a label followed by a colon before the TEST keyword, or you can specify a string in quotation marks after the TEST keyword. If you specify both a label before the TEST keyword and a quoted string after the keyword, PROC CQLIM uses the label that precedes the colon. If no label or quoted string is specified, PROC CQLIM labels the test automatically.

---

**WEIGHT Statement**

```
WEIGHT variable < / option> ;
```

The WEIGHT statement specifies a variable that supplies weighting values to use for each observation in estimating parameters. The log likelihood for each observation is multiplied by the corresponding weight variable value.

If the weight of an observation is nonpositive, that observation is not used in the estimation.

You can add the following option after a slash (/):
**NONNORMALIZE**
specifies that the weights must be used as is. When you omit this option, the weights are normalized so that they add up to the actual sample size. Weights \( w_i \) are normalized by multiplying them by \( \frac{n}{\sum_{i=1}^{n} w_i} \), where \( n \) is the sample size.

---

**Details: CQLIM Procedure**

**Ordinal Discrete Choice Modeling**

**Binary Probit and Logit Model**

The binary choice model is

\[
y_i^* = x_i' \beta + \epsilon_i
\]

where the value of the latent dependent variable, \( y_i^* \), is observed only as follows:

\[
y_i = 1 \quad \text{if} \quad y_i^* > 0
\]
\[
y_i = 0 \quad \text{otherwise}
\]

The disturbance, \( \epsilon_i \), of the probit model has a standard normal distribution with the cumulative distribution function (CDF)

\[
\Phi(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} \exp(-t^2/2)dt
\]

The disturbance of the logit model has a standard logistic distribution with the CDF

\[
\Lambda(x) = \frac{\exp(x)}{1 + \exp(x)} = \frac{1}{1 + \exp(-x)}
\]

The binary discrete choice model has the following probability that the event \( \{ y_i = 1 \} \) occurs:

\[
P(y_i = 1) = F(x_i' \beta) = \begin{cases} \Phi(x_i' \beta) & \text{(probit)} \\ \Lambda(x_i' \beta) & \text{(logit)} \end{cases}
\]

For more information, see *SAS/ETS User’s Guide*.

**Ordinal Probit/Logit**

When the dependent variable is observed in sequence with \( M \) categories, binary discrete choice modeling is not appropriate for data analysis. McKelvey and Zavoina (1975) propose the ordinal (or ordered) probit model.

Consider the regression equation

\[
y_i^* = x_i' \beta + \epsilon_i
\]
where error disturbances, $\epsilon_i$, have the distribution function $F$. The unobserved continuous random variable, $y_i^*$, is identified as $M$ categories. Suppose there are $M + 1$ real numbers, $\mu_0, \ldots, \mu_M$, where $\mu_0 = -\infty$, $\mu_1 = 0$, $\mu_M = \infty$, and $\mu_0 \leq \mu_1 \leq \cdots \leq \mu_M$. Define

$$R_{i,j} = \mu_j - x_i'\beta$$

The probability that the unobserved dependent variable is contained in the $j$th category can be written as

$$P[\mu_{j-1} < y_i^* \leq \mu_j] = F(R_{i,j}) - F(R_{i,j-1})$$

For more information, see SAS/ETS User’s Guide.

### Limited Dependent Variable Models

#### Censored Regression Models

When the dependent variable is censored, all values within a certain range are transformed to a single value. For example, the standard Tobit model can be defined as

$$y_i^* = x_i'\beta + \epsilon_i$$

$$y_i = \begin{cases} y_i^* & \text{if } y_i^* > 0 \\ 0 & \text{if } y_i^* \leq 0 \end{cases}$$

where $\epsilon_i \sim \text{iicl} N(0, \sigma^2)$.

The Tobit model can be generalized to handle observation-by-observation censoring. The model that is censored on both the lower and upper limits can be defined as

$$y_i = \begin{cases} R_i & \text{if } y_i^* \geq R_i \\ y_i^* & \text{if } L_i < y_i^* < R_i \\ L_i & \text{if } y_i^* \leq L_i \end{cases}$$

For more information, see SAS/ETS User’s Guide.

#### Truncated Regression Models

In a truncated model, the observed sample is a subset of the population in which the dependent variable falls within a certain range. For example, when neither a dependent variable nor exogenous variables are observed for $y_i^* \leq 0$, the truncated regression model can be specified as

$$\ell = \sum_{i \in \{y_i > 0\}} \left\{-\ln \Phi(x_i'\beta/\sigma) + \ln \left[\frac{\phi((y_i - x_i'\beta)/\sigma)}{\sigma}\right]\right\}$$

For more information, see SAS/ETS User’s Guide.
Stochastic Frontier Production and Cost Models

Stochastic frontier production models were first developed by Aigner, Lovell, and Schmidt (1977); Meeusen and van den Broeck (1977). Specification of these models allows for random shocks of the production or cost but also includes a term for technical or cost inefficiency. Assuming that the production function takes a log-linear Cobb-Douglas form, the stochastic frontier production model can be written as

$$\ln(y_i) = \beta_0 + \sum_n \beta_n \ln(x_{ni}) + \epsilon_i$$

where $$\epsilon_i = v_i - u_i$$. The $$v_i$$ term represents the stochastic error component, and the $$u_i$$ term represents the nonnegative, technical inefficiency error component. The $$v_i$$ error component is assumed to be distributed iid normal and independent from $$u_i$$. If $$u_i > 0$$, the error term $$\epsilon_i$$ is negatively skewed and represents technical inefficiency. If $$u_i < 0$$, the error term $$\epsilon_i$$ is positively skewed and represents cost inefficiency. PROC CQLIM models the $$u_i$$ error component as a half-normal, exponential, or truncated normal distribution.

The Normal-Half-Normal Model

When $$v_i$$ is iid $$N(0, \sigma_v^2)$$ in a normal-half-normal model, $$u_i$$ is iid $$N^+(0, \sigma_u^2)$$, with $$v_i$$ and $$u_i$$ independent of each other. Given the independence of error terms, the joint density of $$v$$ and $$u$$ can be written as

$$f(u, v) = \frac{2}{2\pi \sigma_u \sigma_v} \exp\left\{ -\frac{u^2}{2\sigma_u^2} - \frac{v^2}{2\sigma_v^2} \right\}$$

Substituting $$v = \epsilon + u$$ into the preceding equation and integrating $$u$$ out gives

$$f(\epsilon) = \frac{2}{\sigma} \phi\left( \frac{\epsilon}{\sigma} \right) \Phi\left( \frac{-\lambda}{\sigma} \right)$$

where $$\lambda = \sigma_u / \sigma_v$$ and $$\sigma = \sqrt{\sigma_u^2 + \sigma_v^2}$$. In the case of a stochastic frontier cost model, $$v = \epsilon - u$$ and

$$f(\epsilon) = \frac{2}{\sigma} \phi\left( \frac{\epsilon}{\sigma} \right) \Phi\left( \frac{\epsilon \lambda}{\sigma} \right)$$

For more information, see SAS/ETS User’s Guide.

The Normal-Exponential Model

Under the normal-exponential model, $$v_i$$ is iid $$N(0, \sigma_v^2)$$ and $$u_i$$ is iid exponential. Given the independence of the error term components $$u_i$$ and $$v_i$$, the joint density of $$v$$ and $$u$$ can be written as

$$f(u, v) = \frac{1}{\sqrt{2\pi} \sigma_u \sigma_v} \exp\left\{ -\frac{u}{\sigma_u} - \frac{v^2}{2\sigma_v^2} \right\}$$

The marginal density function of $$\epsilon$$ for the production function is

$$f(\epsilon) = \left( \frac{1}{\sigma_u} \right) \phi\left( \frac{\epsilon}{\sigma_u} - \frac{\sigma_v}{\sigma_u} \right) \exp\left\{ \frac{\epsilon}{\sigma_u} + \frac{\sigma_v^2}{2\sigma_u^2} \right\}$$
The marginal density function for the cost function is equal to

\[ f(\epsilon) = \left( \frac{1}{\sigma_u} \right) \Phi \left( \frac{\epsilon - \sigma_v}{\sigma_u} \right) \exp \left\{ -\frac{\epsilon^2}{2\sigma_v^2} - \frac{\sigma_v^2}{2\sigma_u^2} \right\} \]

For more information, see SAS/ETS User’s Guide.

**The Normal–Truncated Normal Model**

The normal–truncated normal model is a generalization of the normal-half-normal model that allows the mean of \( u_i \) to differ from 0. Under the normal–truncated normal model, the error term component \( v_i \) is iid \( N^+(0, \sigma_v^2) \), and the error term component \( u_i \) is iid \( N(\mu, \sigma_u^2) \). The joint density of \( v_i \) and \( u_i \) can be written as

\[ f(u, v) = \frac{1}{\sqrt{2\pi} \sigma_u \sigma_v} \Phi \left( \frac{\mu}{\sigma_u} \right) \exp \left\{ -\frac{(u - \mu)^2}{2\sigma_u^2} - \frac{v^2}{2\sigma_v^2} \right\} \]

The marginal density function of \( \epsilon \) for the production function is

\[ f(\epsilon) = \frac{1}{\sigma} \phi \left( \frac{\epsilon + \mu}{\sigma} \right) \Phi \left( \frac{\mu}{\sigma \lambda} - \frac{\epsilon \lambda}{\sigma} \right) \left[ \Phi \left( \frac{\mu}{\sigma_u} \right) \right]^{-1} \]

The marginal density function for the cost function is

\[ f(\epsilon) = \frac{1}{\sigma} \phi \left( \frac{\epsilon - \mu}{\sigma} \right) \Phi \left( \frac{\mu}{\sigma \lambda} + \frac{\epsilon \lambda}{\sigma} \right) \left[ \Phi \left( \frac{\mu}{\sigma_u} \right) \right]^{-1} \]

For more information, see SAS/ETS User’s Guide.

For more information about normal-half-normal, normal-exponential, and normal–truncated normal models, see Kumbhakar and Lovell (2000); Coelli, Prasada Rao, and Battese (1998).

**Heteroscedasticity**

If the variance of regression disturbance \( \epsilon_i \) is heteroscedastic, the variance can be specified as a function of variables

\[ E(\epsilon_i^2) = \sigma_i^2 = f(z_i^T y) \]

Table 5.2 shows various functional forms of heteroscedasticity and the corresponding options to request each model.

<table>
<thead>
<tr>
<th>Number</th>
<th>Model</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( f(z_i^T y) = \sigma^2(1 + \exp(z_i^T y)) )</td>
<td>LINK=EXP (default)</td>
</tr>
<tr>
<td>2</td>
<td>( f(z_i^T y) = \sigma^2 \exp(z_i^T y) )</td>
<td>LINK=EXP NOCONST</td>
</tr>
<tr>
<td>3</td>
<td>( f(z_i^T y) = \sigma^2(1 + \sum_{l=1}^L y_l z_{li}) )</td>
<td>LINK=LINEAR</td>
</tr>
<tr>
<td>4</td>
<td>( f(z_i^T y) = \sigma^2(1 + (\sum_{l=1}^L y_l z_{li})^2) )</td>
<td>LINK=LINEAR SQUARE</td>
</tr>
<tr>
<td>5</td>
<td>( f(z_i^T y) = \sigma^2(\sum_{l=1}^L y_l z_{li}) )</td>
<td>LINK=LINEAR NOCONST</td>
</tr>
<tr>
<td>6</td>
<td>( f(z_i^T y) = \sigma^2((\sum_{l=1}^L y_l z_{li})^2) )</td>
<td>LINK=LINEAR SQUARE NOCONST</td>
</tr>
</tbody>
</table>
In models 3 and 5, variances of some observations might be negative. Although the CQLIM procedure assigns a large penalty to move the optimization away from such a region, the optimization might not be able to improve the objective function value and might become locked in the region. Signs of such an outcome include extremely small likelihood values or missing standard errors in the estimates. In models 2 and 6, variances are guaranteed to be greater than or equal to 0, but variances of some observations might be very close to 0. In these scenarios, standard errors might be missing. Models 1 and 4 do not have such problems. Variances in these models are always positive and never close to 0.

### Tests on Parameters

In general, the tested hypothesis can be written as

\[ H_0 : h(\theta) = 0 \]

where \( h(\theta) \) is an \( r \times 1 \) vector-valued function of the parameters \( \theta \) given by the \( r \) expressions that are specified in the TEST statement.

Let \( \hat{V} \) be the estimate of the covariance matrix of \( \hat{\theta} \). Let \( \hat{\theta} \) be the unconstrained estimate of \( \theta \), and let \( \tilde{\theta} \) be the constrained estimate of \( \theta \) such that \( h(\tilde{\theta}) = 0 \). Let

\[ A(\theta) = \frac{\partial h(\theta)}{\partial \theta} |_{\tilde{\theta}} \]

Using this notation, the test statistics for the three types of tests are computed as follows:

- The Wald test statistic is defined as
  \[ W = h'(\hat{\theta}) \left( A(\hat{\theta}) \hat{V} A'(\hat{\theta}) \right)^{-1} h(\hat{\theta}) \]

- The Lagrange multiplier test statistic is
  \[ LM = \lambda' A(\hat{\theta}) \hat{V} A'(\hat{\theta}) \lambda \]
  where \( \lambda \) is the vector of Lagrange multipliers from the computation of the restricted estimate \( \tilde{\theta} \).

- The likelihood ratio test statistic is
  \[ LR = 2 \left( L(\hat{\theta}) - L(\tilde{\theta}) \right) \]
  where \( \tilde{\theta} \) represents the constrained estimate of \( \theta \) and \( L \) is the concentrated log-likelihood value.

The following statements use the TEST statement to perform a likelihood ratio test:

```sas
proc cqlim;
  model y = x1 x2 x3;
  test x1 = 0, x2 * .5 + 2 * x3 = 0 /lr;
run;
```

For more information, see *SAS/ETS User’s Guide*. 
Output to SAS Data Set

**XBeta, Predicted, and Residual**

XBeta is the structural part on the right-hand side of the model. The predicted value is the predicted dependent variable value. For censored variables, if the predicted value is outside the boundaries, it is reported as the closest boundary. For discrete variables, it is the level whose boundaries XBeta falls between. The residual is defined only for continuous variables and is defined as

\[
\text{Residual} = \text{Observed} - \text{Predicted}
\]

**Error Standard Deviation**

The error standard deviation is \( \sigma_i \) in the model. It varies only when the HETERO statement is used.

**Marginal Effects**

A marginal effect is defined as a contribution of one control variable to the response variable. For a binary choice model with two response categories, \( \mu_0 = -\infty, \mu_1 = 0 \), and \( \mu_2 = \infty \). For an ordinal response model with \( M \) response categories \( (\mu_0, \ldots, \mu_M) \), define

\[
R_{i,j} = \mu_j - x_i' \beta
\]

The probability that the unobserved dependent variable is contained in the \( j \)th category can be written as

\[
P[\mu_{j-1} < y_i^* \leq \mu_j] = F(R_{i,j}) - F(R_{i,j-1})
\]

The marginal effect of changes in the regressors on the probability of \( y_i = j \) is then

\[
\frac{\partial \text{Prob}[y_i = j]}{\partial x} = [f(\mu_{j-1} - x_i' \beta) - f(\mu_j - x_i' \beta)] \beta
\]

where \( f(x) = \frac{dF(x)}{dx} \). In particular,

\[
f(x) = \frac{dF(x)}{dx} = \begin{cases} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} & \text{(probit)} \\ \frac{e^{-x}}{[1+e^{(-x)}]^2} & \text{(logit)} \end{cases}
\]

The marginal effects in the truncated regression model are

\[
\frac{\partial E[y_i | L_i < y_i^* < R_i]}{\partial x} = \beta \left[ 1 - \frac{(\phi(a_i) - \phi(b_i))^2}{(\Phi(b_i) - \Phi(a_i))^2} + \frac{a_i \phi(a_i) - b_i \phi(b_i)}{\Phi(b_i) - \Phi(a_i)} \right]
\]

where \( a_i = \frac{L_i - x_i' \beta}{\sigma_i} \) and \( b_i = \frac{R_i - x_i' \beta}{\sigma_i} \).

The marginal effects in the censored regression model are

\[
\frac{\partial E[y | x_i]}{\partial x} = \beta \times \text{Prob}[L_i < y_i^* < R_i]
\]
Inverse Mills Ratio, Expected and Conditionally Expected Values

Expected and conditionally expected values are computed only for continuous variables. The inverse Mills ratio is computed for censored or truncated continuous, binary discrete, and selection endogenous variables.

Let \( L_i \) and \( R_i \) be the lower boundary and the upper boundary of \( y_i \). Define \( a_i = \frac{L_i - x_i' \beta}{\sigma_i} \) and \( b_i = \frac{R_i - x_i' \beta}{\sigma_i} \).

Then the inverse Mills ratio is defined as

\[
\lambda = \frac{\phi(a_i) - \phi(b_i)}{\Phi(b_i) - \Phi(a_i)}
\]

for a continuous variable and defined as

\[
\lambda = \frac{\phi(x_i' \beta)}{\Phi(x_i' \beta)}
\]

for a binary discrete variable.

The expected value is the unconditional expectation of the dependent variable. For a censored variable, it is

\[
E[y_i] = \Phi(a_i) L_i + (x_i' \beta + \lambda \sigma_i)(\Phi(b_i) - \Phi(a_i)) + (1 - \Phi(b_i)) R_i
\]

For a left-censored variable \( (R_i = \infty) \), this formula is

\[
E[y_i] = \Phi(a_i) L_i + (x_i' \beta + \lambda \sigma_i)(1 - \Phi(a_i))
\]

where \( \lambda = \frac{\phi(a_i)}{1 - \Phi(a_i)} \).

For a right-censored variable \( (L_i = -\infty) \), this formula is

\[
E[y_i] = (x_i' \beta + \lambda \sigma_i) \Phi(b_i) + (1 - \Phi(b_i)) R_i
\]

where \( \lambda = -\frac{\phi(b_i)}{\Phi(b_i)} \).

For a noncensored variable, this formula is

\[
E[y_i] = x_i' \beta
\]

The conditional expected value is the expectation given that the variable is inside the boundaries:

\[
E[y_i \mid L_i < y_i < R_i] = x_i' \beta + \lambda \sigma_i
\]

Probability

Probability applies only to discrete responses. It is the marginal probability that the discrete response is taking the value of the observation. If you specify the PROBALL option, then the probability for all possible responses of the discrete variables is computed.
Technical Efficiency

Technical efficiency for each producer is computed only for stochastic frontier models. In general, the stochastic production frontier can be written as

$$y_i = f(x_i; \beta) \exp\{v_i\} TE_i$$

where $y_i$ denotes producer $i$'s actual output, $f(\cdot)$ is the deterministic part of the production frontier, $\exp\{v_i\}$ is a producer-specific error term, and $TE_i$ is the technical efficiency coefficient, which can be written as

$$TE_i = \frac{y_i}{f(x_i; \beta) \exp\{v_i\}}$$

For a Cobb-Douglas production function, $TE_i = \exp\{-u_i\}$. For more information, see the section “Stochastic Frontier Production and Cost Models” on page 190.

The cost frontier can be written in general as

$$E_i = c(y_i, w_i; \beta) \exp\{v_i\} / CE_i$$

where $w_i$ denotes producer $i$'s input prices, $c(\cdot)$ is the deterministic part of the cost frontier, $\exp\{v_i\}$ is a producer-specific error term, and $CE_i$ is the cost efficiency coefficient, which can be written as

$$CE_i = \frac{c(x_i, w_i; \beta) \exp\{v_i\}}{E_i}$$

For a Cobb-Douglas cost function, $CE_i = \exp\{-u_i\}$. For more information, see the section “Stochastic Frontier Production and Cost Models” on page 190. Hence, both technical and cost efficiency coefficients are the same. The estimates of technical efficiency are provided in the following subsections.

Normal-Half-Normal Model

Define $\mu_* = -\epsilon \sigma_u^2 / \sigma^2$ and $\sigma_*^2 = \sigma^2 \sigma_u^2 / \sigma^2$. Then, as shown by Jondrow et al. (1982), conditional density is as follows:

$$f(u|\epsilon) = \frac{f(u, \epsilon)}{f(\epsilon)} = \frac{1}{\sqrt{2\pi \sigma_*}} \exp \left\{ -\frac{(u - \mu_*)^2}{2\sigma_*^2} \right\} / \left[ 1 - \Phi \left( \frac{-\mu_*}{\sigma_*} \right) \right]$$

Hence, $f(u|\epsilon)$ is the density for $N^+ (\mu_*, \sigma_*^2)$.

From this result, it follows that the estimate of technical efficiency (Battese and Coelli 1988) is

$$TE_1 = E(\exp\{-u_i\}|\epsilon_i) = \left[ \frac{1 - \Phi(\sigma_* - \mu_* / \sigma_*)}{1 - \Phi(-\mu_* / \sigma_*)} \right] \exp \left\{ -\mu_* + \frac{1}{2} \sigma_*^2 \right\}$$

The second version of the estimate (Jondrow et al. 1982) is

$$TE_2 = \exp\{-E(u_i|\epsilon_i)\}$$

where

$$E(u_i|\epsilon_i) = \mu_* + \sigma_* \left[ \frac{\phi(-\mu_* / \sigma_*)}{1 - \Phi(-\mu_* / \sigma_*)} \right] = \sigma_* \left[ \frac{\phi(\epsilon_i \lambda / \sigma)}{1 - \Phi(\epsilon_i \lambda / \sigma)} - \left( \frac{\epsilon_i \lambda}{\sigma} \right) \right]$$
Chapter 5: The CQLIM Procedure

Normal-Exponential Model

Define \( A = -\tilde{\mu} / \sigma_v \) and \( \tilde{\mu} = -\epsilon - \sigma_u^2 / \sigma_v \). Then, as shown by Kumbhakar and Lovell (2000), conditional density is as follows:

\[
f(u|\epsilon) = \frac{1}{\sqrt{2\pi}\sigma_v \Phi(-\tilde{\mu} / \sigma_v)} \exp \left\{ -\frac{(u - \tilde{\mu})^2}{2\sigma^2} \right\}
\]

Hence, \( f(u|\epsilon) \) is the density for \( N^+(\tilde{\mu}, \sigma_v^2) \).

From this result, it follows that the estimate of technical efficiency is

\[
TE_{1i} = E(\exp\{-u_i|\epsilon_i\}) = \left[ 1 - \Phi(\sigma_v - \tilde{\mu}_i / \sigma_v) \right] \exp \left\{ -\tilde{\mu}_i + \frac{1}{2} \sigma_v^2 \right\}
\]

The second version of the estimate is

\[
TE_{2i} = \exp\{-E(u_i|\epsilon_i)\}
\]

where

\[
E(u_i|\epsilon_i) = \tilde{\mu}_i + \sigma_v \left[ \frac{\phi(-\tilde{\mu}_i / \sigma_v)}{1 - \Phi(-\tilde{\mu}_i / \sigma_v)} \right] = \sigma_v \left[ \frac{\phi(A)}{\Phi(-A)} - A \right]
\]

Normal-Truncated Normal Model

Define \( \tilde{\mu} = (-\sigma_u^2 \epsilon_i + \mu \sigma_v^2) / \sigma^2 \) and \( \sigma_\ast^2 = \sigma_u^2 \sigma_v^2 / \sigma^2 \). Then, as shown by Kumbhakar and Lovell (2000), conditional density is as follows:

\[
f(u|\epsilon) = \frac{1}{\sqrt{2\pi} \sigma_\ast [1 - \Phi(-\tilde{\mu} / \sigma_\ast)]} \exp \left\{ -\frac{(u - \tilde{\mu})^2}{2\sigma_\ast^2} \right\}
\]

Hence, \( f(u|\epsilon) \) is the density for \( N^+(\tilde{\mu}, \sigma_\ast^2) \).

From this result, it follows that the estimate of technical efficiency is

\[
TE_{1i} = E(\exp\{-u_i|\epsilon_i\}) = \frac{1 - \Phi(\sigma_\ast - \tilde{\mu}_i / \sigma_\ast)}{1 - \Phi(-\tilde{\mu}_i / \sigma_\ast)} \exp \left\{ -\tilde{\mu}_i + \frac{1}{2} \sigma_\ast^2 \right\}
\]

The second version of the estimate is

\[
TE_{2i} = \exp\{-E(u_i|\epsilon_i)\}
\]

where

\[
E(u_i|\epsilon_i) = \tilde{\mu}_i + \sigma_\ast \left[ \frac{\phi(\tilde{\mu}_i / \sigma_\ast)}{1 - \Phi(-\tilde{\mu}_i / \sigma_\ast)} \right]
\]
Naming

Naming of Parameters

In the CQLIM procedure, the parameters are named in the same way as in other SAS procedures, such as the REG and PROBIT procedures. The constant in the regression equation is called Intercept. The coefficients of independent variables are named by the independent variables. The standard deviation of the errors is called _Sigma. By default, _Limit1 is set to 0 and the limit parameters start from \( i = 2 \). If you also specify the HETERO statement, the coefficients of the independent variables in the HETERO statement are called _H_x, where \( x \) is the name of the independent variable.

Naming of Output Variables

Table 5.3 shows the **options** in the OUTPUT statement, along with the corresponding variable names and their explanations.

<table>
<thead>
<tr>
<th>output-option</th>
<th>Variable Name</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONDITIONAL</td>
<td>CEXPCT_y</td>
<td>Conditional expected value of ( y ), conditioned on the truncation</td>
</tr>
<tr>
<td>ERRSTD</td>
<td>ERRSTD_y</td>
<td>Standard deviation of error term</td>
</tr>
<tr>
<td>EXPECTED</td>
<td>EXPCT_y</td>
<td>Unconditional expected value of ( y )</td>
</tr>
<tr>
<td>MARGINAL</td>
<td>MEFF_x</td>
<td>Marginal effect of ( x ) on ( y ) (( \frac{\partial y}{\partial x} )) with single equation</td>
</tr>
<tr>
<td>PREDICTED</td>
<td>P_y</td>
<td>Predicted value of ( y )</td>
</tr>
<tr>
<td>RESIDUAL</td>
<td>RESID_y</td>
<td>Residual of ( y ), (( y ) – PredictedY)</td>
</tr>
<tr>
<td>PROB</td>
<td>PROB_y</td>
<td>Probability that ( y ) is taking the observed value in this observation (discrete ( y ) only)</td>
</tr>
<tr>
<td>PROBALL</td>
<td>PROB1_y</td>
<td>Probability that ( y ) is taking the ith value (discrete ( y ) only)</td>
</tr>
<tr>
<td>MILLS</td>
<td>MILLS_y</td>
<td>Inverse Mills ratio for ( y )</td>
</tr>
<tr>
<td>TE1</td>
<td>TE1</td>
<td>Technical efficiency estimate for each producer proposed by Battese and Coelli (1988)</td>
</tr>
<tr>
<td>TE2</td>
<td>TE2</td>
<td>Technical efficiency estimate for each producer proposed by Jondrow et al. (1982)</td>
</tr>
<tr>
<td>XBETA</td>
<td>XBETA_y</td>
<td>Structure part (( x'\beta )) of ( y ) equation</td>
</tr>
</tbody>
</table>

ODS Table Names

PROC CQLIM assigns a name to each table that it creates. You can use these names to refer to the tables when you use the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in Table 5.4.
### Table 5.4 ODS Tables Produced in PROC CQLIM

<table>
<thead>
<tr>
<th>ODS Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ODS Tables Created by the MODEL Statement and TEST Statement</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DiscreteRespProfile</td>
<td>Response profile</td>
<td>Default</td>
</tr>
<tr>
<td>FitModelSummary</td>
<td>Summary of nonlinear estimation</td>
<td>Default</td>
</tr>
<tr>
<td>GoodnessOfFit</td>
<td>Pseudo-R-square measures</td>
<td>Default</td>
</tr>
<tr>
<td>ParameterEstimates</td>
<td>Parameter estimates</td>
<td>Default</td>
</tr>
<tr>
<td>SummaryContResponse</td>
<td>Summary of continuous response</td>
<td>Default</td>
</tr>
<tr>
<td>Covariance</td>
<td>Covariance of parameter estimates</td>
<td>COVB</td>
</tr>
<tr>
<td>Correlation</td>
<td>Correlation of parameter estimates</td>
<td>CORRB</td>
</tr>
<tr>
<td>IterationHistory</td>
<td>Iteration history</td>
<td>ITPRINT</td>
</tr>
<tr>
<td>InitialParameterEstimates</td>
<td>Optimization start</td>
<td>ITPRINT</td>
</tr>
<tr>
<td>OptimizationResults</td>
<td>Optimization results</td>
<td>ITPRINT</td>
</tr>
<tr>
<td><strong>ODS Tables Created by the TEST Statement</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TestResults</td>
<td>Test results</td>
<td>Default</td>
</tr>
</tbody>
</table>

---

### Examples: The CQLIM Procedure

#### Example 5.1: Model with Censoring

This example uses the CQLIM procedure to process a large data set in the distributed computing environment.

The following DATA step generates a data set that contains 5 million observations from a censored model. The model contains seven variables.

```sas
data simulate;
  call streaminit(12345);
  array vars x1-x7;
  array parms{7} (3 4 2 4 -3 -5 -3);
  intercept=2;

  do i=1 to 5000000;
    sum_xb=0;
    do j=1 to 7;
      vars[j]=rand('NORMAL',0,1);
      sum_xb=sum_xb+parms[j]*vars[j];
    end;
    y=intercept+sum_xb+400*rand('NORMAL',0,1);
    if y>400 then y=400;
    if y<0 then y=0;
    output;
  end;
  keep y x1-x7;
run;
```
The following statements estimate a censored model. To run these statements successfully, you need to promote the data set to your CAS engine libref. The DATA step assumes that your CAS engine libref is named mycas, but you can substitute any appropriately defined CAS engine libref.

```plaintext
proc cqlim data=mycas.simulate ;
    model y=x1-x7 /censored(lb=0 ub=400);
run;
```

Output 5.1.1 shows the estimation results for the censored model. The “Model Fit Summary” table shows detailed information about the model. All parameter estimates in the “Parameter Estimates” table are highly significant and correspond to their theoretical values that were set during the data generating process.

### Output 5.1.1 Censored Model: Summary

#### Estimating a Tobit Model

**The CQLIM Procedure**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Mean</th>
<th>Standard Error</th>
<th>Type</th>
<th>Lower Bound</th>
<th>Upper Bound</th>
<th>N Obs Lower Bound</th>
<th>N Obs Upper Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>127.0423</td>
<td>159.491090</td>
<td>Censored</td>
<td>0</td>
<td>400</td>
<td>249E4</td>
<td>8E5</td>
</tr>
</tbody>
</table>

**Model Fit Summary**

- **Execution Mode**: CAS
- **Dependent Variable**: y
- **Number of Observations**: 500000
- **Data Set**: SIMULATE
- **Model**: Unknown
- **Optimizer**: TKZNLO
- **Log Likelihood**: -1.527E7
- **Maximum Absolute Gradient**: 6.67E-9
- **Number of Iterations**: 8
- **Optimization Method**: Newton-Raphson
- **AIC**: 30537962
- **SBC**: 30538083
- **Covariance Estimation**: Hessian

Optimization routine cannot improve the function value.

| Parameter | DF | Estimate  | Standard Error | t Value | Approx Pr > |t| |
|-----------|----|-----------|----------------|---------|--------------|----|
| Intercept | 1  | 2.220381  | 0.222201       | 9.99    | <.0001       |
| x1        | 1  | 3.055524  | 0.201620       | 15.15   | <.0001       |
| x2        | 1  | 4.000179  | 0.201570       | 19.85   | <.0001       |
| x3        | 1  | 1.852739  | 0.201555       | 9.19    | <.0001       |
| x4        | 1  | 4.170277  | 0.201533       | 20.69   | <.0001       |
| x5        | 1  | -3.010677 | 0.201458       | -14.94  | <.0001       |
| x6        | 1  | -5.176016 | 0.201541       | -25.68  | <.0001       |
| x7        | 1  | -2.695934 | 0.201671       | -13.37  | <.0001       |
| Sigma     | 1  | 399.997844| 0.261930       | 1527.12 | <.0001       |
References


Chapter 6
The SEVSELECT Procedure

Contents

Overview: SEVSELECT Procedure .................................................. 202
    Using CAS Sessions and CAS Engine Librefs ............................... 203

Getting Started: SEVSELECT Procedure ......................................... 204
    A Simple Example of Fitting Predefined Distributions ................. 204
    An Example with Left-Truncation and Right-Censoring .............. 207
    An Example of Modeling Regression Effects ............................. 210

Syntax: SEVSELECT Procedure ..................................................... 214
    Functional Summary .......................................................... 215
    PROC SEVSELECT Statement ................................................. 218
    BY Statement ..................................................................... 224
    CLASS Statement .................................................................. 224
    DISPLAY Statement ............................................................. 225
    DISPLAYOUT Statement .......................................................... 226
    DIST Statement .................................................................... 226
    DIAGNOSTICS Statement .......................................................... 228
    EFFECT Statement ............................................................... 229
    LOSS Statement .................................................................... 230
    NLOPTIONS Statement ............................................................. 233
    OUTPUT Statement ................................................................. 233
    SCALEMODEL Statement .......................................................... 236
    SELECTION Statement ............................................................. 238
    WEIGHT Statement ................................................................. 240

Details: SEVSELECT Procedure ..................................................... 241
    Predefined Distributions ......................................................... 241
    Censoring and Truncation .......................................................... 251
    Parameter Estimation Method ..................................................... 253
    Parameter Initialization ............................................................. 255
    Estimating Regression Effects ..................................................... 256
    Empirical Distribution Function Estimation Methods .................... 261
    Statistics of Fit ..................................................................... 267
    Defining a Severity Distribution Model with the FCMP Procedure .... 272
    Predefined Utility Functions ....................................................... 285
    Input Data Tables .................................................................... 290
    Output Data Tables .................................................................. 291
    Displayed Output ..................................................................... 295

Examples: SEVSELECT Procedure .................................................. 297
    Example 6.1: Defining a Model for the Gaussian Distribution with a Scale Parameter 297
Overview: SEVSELECT Procedure

The SEVSELECT procedure estimates parameters of any arbitrary continuous probability distribution that is used to model the magnitude (severity) of a continuous-valued event of interest. Examples of such events include loss amounts paid by an insurance company and demand of a product as depicted by its sales. PROC SEVSELECT is especially useful when the severity of an event does not follow typical distributions (such as the normal distribution) that are often assumed by standard statistical methods.

PROC SEVSELECT provides a default set of probability distribution models that includes the Burr, exponential, gamma, generalized Pareto, inverse Gaussian (Wald), lognormal, Pareto, Tweedie, and Weibull distributions. In the simplest form, you can estimate the parameters of any of these distributions by using a list of severity values that are recorded in a data table. You can optionally group the values by a set of BY variables. PROC SEVSELECT computes the estimates of the model parameters, their standard errors, and their covariance structure by using the maximum likelihood method for each of the BY groups.

PROC SEVSELECT can fit multiple distributions at the same time and choose the best distribution according to a selection criterion that you specify. You can use seven different statistics of fit as selection criteria. They are log likelihood, Akaike’s information criterion (AIC), corrected Akaike’s information criterion (AICC), Schwarz Bayesian information criterion (BIC), Kolmogorov-Smirnov statistic (KS), Anderson-Darling statistic (AD), and Cramér–von Mises statistic (CvM).

You can request that the procedure output different types of diagnostic and inferential results, including the summary statistics of analysis variables, progress and status of the nonlinear estimation process, parameter estimates and their standard errors, estimated covariance structure of the parameters, and statistics of fit.

The following key features make PROC SEVSELECT unique among SAS procedures that can estimate continuous probability distributions:

- It enables you to fit a distribution model when the severity values are truncated, censored, or both. You can specify any combination of the following types of censoring and truncation effects: left-censoring, right-censoring, left-truncation, or right-truncation. This is especially useful in applications with an insurance-type model, where a severity (loss) is reported and recorded only if it is greater than the deductible amount (left-truncation) and where a severity value greater than or equal to the policy limit is recorded at the limit (right-censoring). Another useful application is that of interval-censored data, where you know both the lower limit (right-censoring) and upper limit (left-censoring) on the severity, but you do not know the exact value.

PROC SEVSELECT also enables you to specify a probability of observability for the left-truncated data, which is a probability of observing values greater than the left-truncation threshold. This additional
information can be useful in certain applications to more correctly model the distribution of the severity of events.

- It uses an appropriate estimator of the empirical distribution function (EDF). EDF is required to compute the KS, AD, and CvM statistics of fit. The procedure also provides the EDF estimates to your custom parameter initialization method. When you specify truncation or censoring, the EDF is estimated by using either Kaplan-Meier’s product-limit estimator or Turnbull’s estimator. The former is used by default when you specify only one form of censoring effect (right-censoring or left-censoring), and the latter is used by default when you specify both left-censoring and right-censoring effects.

- It enables you to define any arbitrary continuous parametric distribution model and to estimate its parameters. You just need to define the key components of the distribution, such as its probability density function (PDF) and cumulative distribution function (CDF), as a set of functions and subroutines written with the FCMP procedure, which is part of Base SAS software. As long as the functions and subroutines follow certain rules, the SEVSELECT procedure can fit the distribution model defined by them.

- It can model the influence of exogenous or regressor variables on a probability distribution, as long as the distribution has a scale parameter. A linear combination of regression effects is assumed to affect the scale parameter via an exponential link function. This type of model is referred to as the scale regression model.

  If a distribution does not have a scale parameter, then either it needs to have another parameter that can be derived from a scale parameter by using a supported transformation or it needs to be reparameterized to have a scale parameter. If neither of these is possible, then regression effects cannot be modeled.

  You can easily specify many types of regression effects by using various operators on a set of classification and continuous variables. You can specify classification variables in the CLASS statement. You can also construct the following special effects by using the EFFECT statement: collection effects, multimember effects, polynomial effects, and spline effects.

  If you specify a large number of regression effects, then you can use the SELECTION statement to tell PROC SEVSELECT to perform scale regression model selection.

PROC SEVSELECT is the next-generation version of PROC HPSEVERITY. It requires SAS Cloud Analytic Services (CAS) in order to run. Because PROC SEVSELECT is a next-generation high-performance analytical procedure, it also does the following:

- enables you to run on a cluster of machines that distribute the data and the computations
- exploits all the available cores and concurrent threads

---

**Using CAS Sessions and CAS Engine Librefs**

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

```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 2 in Chapter 1, “Shared Concepts.”

---

**Getting Started: SEVSELECT Procedure**

This section outlines the use of the SEVSELECT procedure to fit continuous probability distribution models. Three examples illustrate different features of the procedure.

---

**A Simple Example of Fitting Predefined Distributions**

The simplest way to use PROC SEVSELECT is to fit all the predefined distributions to a set of values and let the procedure identify the best fitting distribution.

Consider a lognormal distribution, whose probability density function (PDF) $f$ and cumulative distribution function (CDF) $F$ are as follows, respectively, where $\Phi$ denotes the CDF of the standard normal distribution:

$$
f(x; \mu, \sigma) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{1}{2} \left( \frac{\log(x) - \mu}{\sigma} \right)^2}
\quad \text{and} \quad
F(x; \mu, \sigma) = \Phi \left( \frac{\log(x) - \mu}{\sigma} \right)
$$

The following DATA step statements simulate a sample from a lognormal distribution with population parameters $\mu = 1.5$ and $\sigma = 0.25$, and store the sample in the variable `Y` of a data set `Work.Test_sev1`:
/*------------- Simple Lognormal Example -------------*/
data test_sev1(keep=y label='Simple Lognormal Sample');
call streaminit(45678);
label y='Response Variable';
Mu = 1.5;
Sigma = 0.25;
do n = 1 to 100;
y = exp(Mu) * rand('LOGNORMAL')**Sigma;
output;
end;
run;

NOTE: Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections “Using CAS Sessions and CAS Engine Librefs” on page 2 and “Loading a SAS Data Set onto a CAS Server” on page 3 in Chapter 1, “Shared Concepts.”

You can load the Work.Test_sev1 data set into a data table in your CAS session by using your CAS engine libref with the following DATA step:

data mycas.test_sev1;
set test_sev1;
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 203, but you can substitute any appropriately defined CAS engine libref.

The following statements fit all the predefined distribution models to the values of Y and identify the best distribution according to the corrected Akaike’s information criterion (AICC):

proc sevselect data=mycas.test_sev1 crit=aicc;
  loss y;
  dist _predefined_
run;

The PROC SEVSELECT statement specifies the input data table along with the model selection criterion, the LOSS statement specifies the variable to be modeled, and the DIST statement with the _PREDEFINED_ keyword specifies that all the predefined distribution models be fitted.

Some of the default output displayed by this step is shown in Figure 6.1 through Figure 6.3. First, information about the input data table is displayed followed by the “Model Selection” table, as shown in Figure 6.1. The model selection table displays the convergence status, the value of the selection criterion, and the selection status for each of the candidate models. The Converged column indicates whether the estimation process for a particular distribution model has converged, might have converged, or failed. The Selected column indicates whether a particular distribution has the best fit for the data according to the selection criterion. For this example, the lognormal distribution model is selected, because it has the lowest value for the selection criterion.

Figure 6.1 Data Table Information and Model Selection Table

<table>
<thead>
<tr>
<th>Input Data Table</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>TEST_SEV1</td>
</tr>
<tr>
<td>Caslib</td>
<td>&lt;&lt; active caslib &gt;&gt;</td>
</tr>
</tbody>
</table>
Next, the estimation information for each of the candidate models is displayed. The information for the lognormal model, which is the best fitting model, is shown in Figure 6.2. The first table displays a summary of the distribution. The second table displays the convergence status. This is followed by a summary of the optimization process which indicates the technique used, the number of iterations, the number of times the objective function was evaluated, and the log likelihood attained at the end of the optimization. Since the model with lognormal distribution has converged, PROC SEVSELECT displays its statistics of fit and parameter estimates. The estimates of $\mu = 1.49605$ and $\sigma = 0.26243$ are quite close to the population parameters of $\mu = 1.5$ and $\sigma = 0.25$ from which the sample was generated. The $p$-value for each estimate indicates the rejection of the null hypothesis that the estimate is 0, implying that both the estimates are significantly different from 0.

**Figure 6.2** Estimation Details for the Lognormal Model

**The SEVSELECT Procedure**

**Logn Distribution**

<table>
<thead>
<tr>
<th>Model Information</th>
<th>Distribution</th>
<th>Description</th>
<th>Distribution Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Logn</td>
<td>Lognormal Distribution</td>
<td>2</td>
</tr>
</tbody>
</table>

**Convergence Status**

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

<table>
<thead>
<tr>
<th>Optimization Summary</th>
<th>Optimization Technique</th>
<th>Iterations</th>
<th>Function Calls</th>
<th>Log Likelihood</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Trust Region</td>
<td>3</td>
<td>10</td>
<td>-157.72104</td>
</tr>
</tbody>
</table>
The parameter estimates of the Burr distribution are shown in Figure 6.3. These estimates are used in the next example.

**Figure 6.3** Parameter Estimates for the Burr Model

| Parameter | DF | Estimate | Standard Error | t Value | Approx Pr > |t|
|-----------|----|----------|----------------|--------|-------------|
| Mu        | 1  | 1.49605  | 0.02651        | 56.43  | <.0001      |
| Sigma     | 1  | 0.26243  | 0.01874        | 14.00  | <.0001      |
| Theta     | 1  | 4.62348  | 0.46181        | 10.01  | <.0001      |
| Alpha     | 1  | 1.15706  | 0.47493        | 2.44   | 0.0167      |
| Gamma     | 1  | 6.41227  | 0.99039        | 6.47   | <.0001      |

**An Example with Left-Truncation and Right-Censoring**

PROC SEVSELECT enables you to specify that the response variable values are left-truncated or right-censored. The following DATA step expands the data table of the previous example to simulate a scenario that is typically encountered by an automobile insurance company. The values of the variable $Y$ represent the loss values on claims that are reported to an auto insurance company. The variable $\text{THRESHOLD}$ records the deductible on the insurance policy. If the actual value of $Y$ is less than or equal to the deductible, then it is unobservable and does not get recorded. In other words, $\text{THRESHOLD}$ specifies the left-truncation of $Y$. $\text{LIMIT}$ records the policy limit. If the value of $Y$ is equal to or greater than the recorded value, then the observation is right-censored.

```sas
/*----- Lognormal Model with left-truncation and censoring -----*/
data test_sev2(keep=y threshold limit
    label='A Lognormal Sample With Censoring and Truncation');
  set test_sev1;
  label y='Censored & Truncated Response';
  if _n_ = 1 then call streaminit(45679);  
  if (rand('UNIFORM') < 0.2) then 
    if (rand('UNIFORM') < 0.2) then
```
threshold = y * (1 - rand('UNIFORM'));
else
    threshold = .;
/* make about 15% of the observations right-censored */
iscens = (rand('UNIFORM') < 0.15);
if (iscens) then
    limit = y;
else
    limit = .;
run;

The following DATA step loads the Work.Test_sev2 data set into a data table in your CAS session that is associated with the mycas CAS engine libref:

    data mycas.test_sev2;
    set test_sev2;
    run;

The following statements fit four predefined distributions (lognormal, Burr, gamma, and Weibull) and use the AICC to identify the distribution that best fits the data:

    proc sevselect data=mycas.test_sev2 crit=aicc print=all ;
    loss y / lt=threshold rc=limit;
    dist logn burr gamma weibull;
    run;

The LOSS statement specifies the left-truncation and right-censoring variables. The DIST statement specifies the candidate distributions. The PRINT= option in the PROC SEVSELECT statement requests that all the displayed output be prepared.

Some of the key results that PROC SEVSELECT prepares are shown in Figure 6.4 through Figure 6.7. In addition to the estimates of the range, mean, and standard deviation of Y, the “Descriptive Statistics for y” table shown in Figure 6.4 also indicates the number of observations that are left-truncated or right-censored. The “Model Selection” table in Figure 6.4 shows that models with all the candidate distributions have converged and that the Logn (lognormal) model has the best fit for the data according to the AICC criterion.

**Figure 6.4** Summary Results for the Truncated and Censored Data

### The SEVSELECT Procedure

<table>
<thead>
<tr>
<th>Descriptive Statistics for y</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observations</td>
</tr>
<tr>
<td>Observations Used for Estimation</td>
</tr>
<tr>
<td>Minimum</td>
</tr>
<tr>
<td>Maximum</td>
</tr>
<tr>
<td>Mean</td>
</tr>
<tr>
<td>Standard Deviation</td>
</tr>
<tr>
<td>Left Truncated Observations</td>
</tr>
<tr>
<td>Right Censored Observations</td>
</tr>
</tbody>
</table>
PROC SEVSELECT also prepares a table that shows all the fit statistics for all the candidate models. It is useful to see which model would be the best fit according to each of the criteria. The “All Fit Statistics” table prepared for this example is shown in Figure 6.5. It indicates that the lognormal model is chosen by all the criteria.

**Figure 6.5** Comparing All Statistics of Fit for the Truncated and Censored Data

<table>
<thead>
<tr>
<th>Distribution</th>
<th>-2 Log Likelihood</th>
<th>AIC</th>
<th>AICC</th>
<th>SBC</th>
<th>KS</th>
<th>AD</th>
<th>CVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logn</td>
<td>294.80301*</td>
<td>298.80301*</td>
<td>298.92672*</td>
<td>304.01335*</td>
<td>0.72456*</td>
<td>0.94385</td>
<td>0.14529*</td>
</tr>
<tr>
<td>Burr</td>
<td>296.41229</td>
<td>302.41229</td>
<td>302.66229</td>
<td>310.22780</td>
<td>0.76247</td>
<td>0.91887*</td>
<td>0.14894</td>
</tr>
<tr>
<td>Gamma</td>
<td>295.32921</td>
<td>299.32921</td>
<td>299.45293</td>
<td>304.53955</td>
<td>0.73034</td>
<td>0.97265</td>
<td>0.14801</td>
</tr>
<tr>
<td>Weibull</td>
<td>305.14408</td>
<td>309.14408</td>
<td>309.26779</td>
<td>314.35442</td>
<td>0.89590</td>
<td>1.23620</td>
<td>0.18087</td>
</tr>
</tbody>
</table>

* Asterisk (*) denotes the best model in the column.

**Specifying Initial Values for Parameters**

All the predefined distributions have parameter initialization functions built into them. For the current example, Figure 6.6 shows the initial values that are obtained by the predefined method for the Burr distribution. It also shows the summary of the optimization process and the final parameter estimates.

**Figure 6.6** Burr Model Summary for the Truncated and Censored Data

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Initial Value</th>
<th>Lower Bound</th>
<th>Upper Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>Theta</td>
<td>4.77976</td>
<td>Infy</td>
<td></td>
</tr>
<tr>
<td>Alpha</td>
<td>2.00000</td>
<td>Infy</td>
<td></td>
</tr>
<tr>
<td>Gamma</td>
<td>2.00000</td>
<td>Infy</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Optimization Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimization Technique</td>
</tr>
<tr>
<td>Iterations</td>
</tr>
<tr>
<td>Function Calls</td>
</tr>
<tr>
<td>Log Likelihood</td>
</tr>
</tbody>
</table>
You can specify a different set of initial values if estimates are available from fitting the distribution to similar data. For this example, the parameters of the Burr distribution can be initialized with the final parameter estimates of the Burr distribution that were obtained in the first example (shown in Figure 6.3). One of the ways in which you can specify the initial values is as follows:

```plaintext
/*------ Specifying initial values using INIT= option -------*/
proc sevselect data=mycas.test_sev2 crit=aicc print=all;
    loss y / lt=threshold rc=limit;
    dist burr(init=(theta=4.62348 alpha=1.15706 gamma=6.41227));
run;
```

The names of the parameters that are specified in the INIT option must match the parameter names in the definition of the distribution. The results obtained with these initial values are shown in Figure 6.7. These results indicate that new set of initial values causes the optimizer to reach the same solution with fewer iterations and function evaluations as compared to the default initialization.

### An Example of Modeling Regression Effects

Consider a scenario in which the magnitude of the response variable might be affected by some regressor (exogenous or independent) variables. The SEVSELECT procedure enables you to model the effect of such variables on the distribution of the response variable via an exponential link function. In particular, if
you have \( k \) random regressor variables denoted by \( x_j \) \((j = 1, \ldots, k)\), then the distribution of the response variable \( Y \) is assumed to have the form

\[
Y \sim \exp\left(\sum_{j=1}^{k} \beta_j x_j \right) \cdot \mathcal{F}(\Theta)
\]

where \( \mathcal{F} \) denotes the distribution of \( Y \) with parameters \( \Theta \) and \( \beta_j \) \((j = 1, \ldots, k)\) denote the regression parameters (coefficients).

For the effective distribution of \( Y \) to be a valid distribution from the same parametric family as \( \mathcal{F} \), it is necessary for \( \mathcal{F} \) to have a scale parameter. The effective distribution of \( Y \) can be written as

\[
Y \sim \mathcal{F}(\theta, \Omega)
\]

where \( \theta \) denotes the scale parameter and \( \Omega \) denotes the set of nonscale parameters. The scale \( \theta \) is affected by the regressors as

\[
\theta = \theta_0 \cdot \exp\left(\sum_{j=1}^{k} \beta_j x_j \right)
\]

where \( \theta_0 \) denotes a base value of the scale parameter.

Given this form of the model, PROC SEVSELECT allows a distribution to be a candidate for modeling regression effects only if it has an untransformed or a log-transformed scale parameter.

All the predefined distributions, except the lognormal distribution, have a direct scale parameter (that is, a parameter that is a scale parameter without any transformation). For the lognormal distribution, the parameter \( \mu \) is a log-transformed scale parameter. This can be verified by replacing \( \mu \) with a parameter \( \theta = e^\mu \), which results in the following expressions for the PDF \( f \) and the CDF \( F \) in terms of \( \theta \) and \( \sigma \), respectively, where \( \Phi \) denotes the CDF of the standard normal distribution:

\[
f(x; \theta, \sigma) = \frac{1}{x \sigma \sqrt{2\pi}} e^{-\frac{1}{2} \left( \frac{\log(x) - \log(\theta)}{\sigma} \right)^2}
\]

and

\[
F(x; \theta, \sigma) = \Phi \left( \frac{\log(x) - \log(\theta)}{\sigma} \right)
\]

With this parameterization, the PDF satisfies the \( f(x; \theta, \sigma) = \frac{1}{\sigma} f\left(x; \frac{\theta}{\sigma}\right) \) condition and the CDF satisfies the \( F(x; \theta, \sigma) = F\left(\frac{x}{\theta}; \frac{\sigma}{\theta}\right) \) condition. This makes \( \theta \) a scale parameter. Hence, \( \mu = \log(\theta) \) is a log-transformed scale parameter and the lognormal distribution is eligible for modeling regression effects.

The following DATA step simulates a lognormal sample whose scale is decided by the values of the three regressors \( X1, X2, \) and \( X3 \) as follows:

\[
\mu = \log(\theta) = 1 + 0.75 X1 - X2 + 0.25 X3
\]

```sas
/*----------- Lognormal Model with Regressors -----------*/
data test_sev3(keep=y x1-x3 label='A Lognormal Sample Affected by Regressors');
array x{*} x1-x3;
array b{4} _TEMPORARY_ (1 0.75 -1 0.25);
call streaminit(45678);
label y='Response Influenced by Regressors';
Sigma = 0.25;
```
do n = 1 to 100;
   Mu = b(1); /* log of base value of scale */
   do i = 1 to dim(x);
      x(i) = rand('UNIFORM');
      Mu = Mu + b(i+1) * x(i);
   end;
   y = exp(Mu) * rand('LOGNORMAL')**Sigma;
   output;
end;
run;

The following DATA step loads the Work.Test_sev3 data set into a data table in your CAS session that is associated with the mycas CAS engine libref:

data mycas.test_sev3;
   set test_sev3;
run;

The following PROC SEVSELECT step fits the lognormal, Burr, and gamma distribution models to these data. The regressors are specified in the SCALEMODEL statement.

proc sevselect data=mycas.test_sev3 crit=aicc print=all;
   loss y;
   scalemodel x1-x3;
   dist logn burr gamma;
run;

Some of the key results that PROC SEVSELECT prepares are shown in Figure 6.8 through Figure 6.12. The descriptive statistics of all the variables are shown in Figure 6.8.

<table>
<thead>
<tr>
<th>Variable</th>
<th>N</th>
<th>Minimum</th>
<th>Maximum</th>
<th>Mean</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>100</td>
<td>0.0005115</td>
<td>0.97971</td>
<td>0.51689</td>
<td>0.28206</td>
</tr>
<tr>
<td>x2</td>
<td>100</td>
<td>0.01883</td>
<td>0.99937</td>
<td>0.47345</td>
<td>0.28885</td>
</tr>
<tr>
<td>x3</td>
<td>100</td>
<td>0.00255</td>
<td>0.97558</td>
<td>0.48301</td>
<td>0.29709</td>
</tr>
</tbody>
</table>

The comparison of the fit statistics of all the models is shown in Figure 6.9. It indicates that the lognormal model is the best model according to each of the likelihood-based statistics, whereas the gamma model is the best model according to two of the three EDF-based statistics.
An Example of Modeling Regression Effects

Figure 6.9  Comparison of Statistics of Fit for the Regression Example

<table>
<thead>
<tr>
<th>Distribution</th>
<th>-2 Log Likelihood</th>
<th>AIC</th>
<th>AICC</th>
<th>SBC</th>
<th>KS</th>
<th>AD</th>
<th>CvM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logn</td>
<td>187.49609*</td>
<td>197.49609*</td>
<td>198.13439*</td>
<td>210.52194*</td>
<td>1.28165</td>
<td>6.49698</td>
<td>0.44474</td>
</tr>
<tr>
<td>Burr</td>
<td>190.69154</td>
<td>202.69154</td>
<td>203.59476</td>
<td>218.32256</td>
<td>1.31797</td>
<td>5.17575*</td>
<td>0.46811</td>
</tr>
<tr>
<td>Gamma</td>
<td>188.91483</td>
<td>198.91483</td>
<td>199.55313</td>
<td>211.94069</td>
<td>1.25670*</td>
<td>5.95959</td>
<td>0.43184*</td>
</tr>
</tbody>
</table>

*Asterisk (*) denotes the best model in the column.*

The model information and the convergence results of the lognormal model are shown in Figure 6.10. The iteration history gives you a summary of how the optimizer is traversing the surface of the log-likelihood function in its attempt to reach the optimum. Both the change in the log likelihood and the maximum gradient of the objective function with respect to any of the parameters typically approach 0 if the optimizer converges.

Figure 6.10  Convergence Results for the Lognormal Model with Regressors

The SEVSELECT Procedure

Logn Distribution

<table>
<thead>
<tr>
<th>Model Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distribution</td>
</tr>
<tr>
<td>Description</td>
</tr>
<tr>
<td>Distribution Parameters</td>
</tr>
<tr>
<td>Regression Parameters</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Convergence Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convergence criterion (GCONV=1E-8) satisfied.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Optimization Iteration History</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iter</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Optimization Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimization Technique</td>
</tr>
<tr>
<td>Iterations</td>
</tr>
<tr>
<td>Function Calls</td>
</tr>
<tr>
<td>Log Likelihood</td>
</tr>
</tbody>
</table>

The final parameter estimates of the lognormal model are shown in Figure 6.11. All the estimates are significantly different from 0. The estimate that is reported for the parameter $\mu$ is the base value for the log-transformed scale parameter $\mu$. Let $x_i (1 \leq i \leq 3)$ denote the observed value for regressor $X_i$. If the lognormal distribution is chosen to model $Y$, then the effective value of the parameter $\mu$ varies with the observed values of regressors as

$$\mu = 1.04047 + 0.65221 x_1 - 0.91116 x_2 + 0.16243 x_3$$
These estimated coefficients are reasonably close to the population parameters (that is, within one or two standard errors).

**Figure 6.11** Parameter Estimates for the Lognormal Model with Regressors

| Parameter | DF | Estimate | Standard Error | t Value | Approx Pr > |t|
|-----------|----|----------|----------------|---------|-------------|
| Mu        | 1  | 1.04047  | 0.07614        | 13.66   | <.0001      |
| Sigma     | 1  | 0.22177  | 0.01609        | 13.78   | <.0001      |
| x1        | 1  | 0.65221  | 0.08167        | 7.99    | <.0001      |
| x2        | 1  | -0.91116 | 0.07946        | -11.47  | <.0001      |
| x3        | 1  | 0.16243  | 0.07782        | 2.09    | 0.0395      |

The estimates of the gamma distribution model, which is the best model according to a majority of the EDF-based statistics, are shown in **Figure 6.12**. The estimate that is reported for the parameter \( \Theta \) is the base value for the scale parameter \( \theta \). If the gamma distribution is chosen to model \( Y \), then the effective value of the scale parameter is \( \hat{\theta} = 0.14293 \exp(0.64562 x_1 - 0.89831 x_2 + 0.14901 x_3) \).

**Figure 6.12** Parameter Estimates for the Gamma Model with Regressors

| Parameter | DF | Estimate | Standard Error | t Value | Approx Pr > |t|
|-----------|----|----------|----------------|---------|-------------|
| Theta     | 1  | 0.14293  | 0.02329        | 6.14    | <.0001      |
| Alpha     | 1  | 20.37726 | 2.93277        | 6.95    | <.0001      |
| x1        | 1  | 0.64562  | 0.08224        | 7.85    | <.0001      |
| x2        | 1  | -0.89831 | 0.07962        | -11.28  | <.0001      |
| x3        | 1  | 0.14901  | 0.07870        | 1.89    | 0.0613      |

**Syntax: SEVSELECT Procedure**

The following statements are available in the SEVSELECT procedure:

```plaintext
PROC SEVSELECT options ;
   BY variable-list ;
   LOSS < response-variable> < / censoring-truncation-options> ;
   WEIGHT weight-variable ;
   DIST distribution-name-or-keyword < (distribution-option)< distribution-name-or-keyword < (distribution-option)> > . . . < / options > ;
   CLASS variable < (options) > . . . < variable < (options) > < / global-options > ;
   EFFECT name=effect-type(variables < / options > ) ;
   SCALEMODEL regression-effect-list < / scalemodel-options > ;
   SELECTION < METHOD=method < (method-options) > < options > ;
   DISPLAY < table-list > < / options > ;
   DISPLAYOUT table-spec-list < / options > ;
   OUTPUT < OUT=CAS-libref.data-table > output-options ;
   NLOPTIONS options ;
```

### Table 6.1 Functional Summary

<table>
<thead>
<tr>
<th>Description</th>
<th>Statement</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Statements</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specifies BY-group processing</td>
<td>BY</td>
<td></td>
</tr>
<tr>
<td>Specifies the response variable to model along with censoring and truncation effects</td>
<td>LOSS</td>
<td></td>
</tr>
<tr>
<td>Specifies the weight variable</td>
<td>WEIGHT</td>
<td></td>
</tr>
<tr>
<td>Specifies distributions to fit</td>
<td>DIST</td>
<td></td>
</tr>
<tr>
<td>Specifies the classification variables</td>
<td>CLASS</td>
<td></td>
</tr>
<tr>
<td>Specifies the constructed regression effects</td>
<td>EFFECT</td>
<td></td>
</tr>
<tr>
<td>Specifies the regression effects to model</td>
<td>SCALEMODEL</td>
<td></td>
</tr>
<tr>
<td>Specifies the options for selecting the best subset of regression effects</td>
<td>SELECTION</td>
<td></td>
</tr>
<tr>
<td>Specifies the ODS tables to display (an alternative to the PRINT= option)</td>
<td>DISPLAY</td>
<td></td>
</tr>
<tr>
<td>Specifies the ODS tables to save as CAS output tables</td>
<td>DISPLAYOUT</td>
<td></td>
</tr>
<tr>
<td>Specifies the scoring functions and quantiles to write</td>
<td>OUTPUT</td>
<td></td>
</tr>
<tr>
<td>Specifies optimization options</td>
<td>NLOPTIONS</td>
<td></td>
</tr>
<tr>
<td><strong>Input and Output Options</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specifies the input data table</td>
<td>PROC SEVSELECT DATA=</td>
<td></td>
</tr>
<tr>
<td>Specifies the input data table for parameter estimates</td>
<td>PROC SEVSELECT INEST=</td>
<td></td>
</tr>
<tr>
<td>Specifies the output data table for estimates of scoring functions and quantiles</td>
<td>OUTPUT OUT=</td>
<td></td>
</tr>
<tr>
<td>Specifies the output data table for parameter estimates</td>
<td>PROC SEVSELECT OUTTEST=</td>
<td></td>
</tr>
<tr>
<td><strong>Data Interpretation Options</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specifies left-censoring</td>
<td>LOSS</td>
<td>LEFTCENSORED=</td>
</tr>
<tr>
<td>Specifies left-truncation</td>
<td>LOSS</td>
<td>LEFTTRUNCATED=</td>
</tr>
<tr>
<td>Specifies the probability of observability</td>
<td>LOSS</td>
<td>PROBOBSERVED=</td>
</tr>
<tr>
<td>Specifies right-censoring</td>
<td>LOSS</td>
<td>RIGHTCENSORED=</td>
</tr>
<tr>
<td>Specifies right-truncation</td>
<td>LOSS</td>
<td>RIGHTTRUNCATED=</td>
</tr>
<tr>
<td><strong>Distribution Processing Options</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specifies the CAS table to read the distribution definitions from</td>
<td>DIST</td>
<td>INFUNCDEF=</td>
</tr>
<tr>
<td>Specifies that distributions be listed in the log without estimating any models that use them</td>
<td>DIST</td>
<td>LISTONLY</td>
</tr>
<tr>
<td>Description</td>
<td>Statement</td>
<td>Option</td>
</tr>
<tr>
<td>-----------------------------------------------------------------------------</td>
<td>-------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>Specifies the CAS table to write the distribution definitions to</td>
<td>DIST</td>
<td>OUTFUNCDEF=</td>
</tr>
<tr>
<td>Specifies that distributions be validated without estimating any models that use them</td>
<td>DIST</td>
<td>VALIDATEONLY</td>
</tr>
</tbody>
</table>

**Model Estimation Options**

<table>
<thead>
<tr>
<th>Description</th>
<th>Statement</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specifies the method for computing mixture distribution</td>
<td>SCALEMODEL</td>
<td>DFMIXTURE=</td>
</tr>
<tr>
<td>Specifies that informative missingness be used to model missing values of regressors</td>
<td>SCALEMODEL</td>
<td>INFORMATIVE</td>
</tr>
<tr>
<td>Specifies initial values for model parameters</td>
<td>DIST</td>
<td>INIT=</td>
</tr>
<tr>
<td>Specifies the sample to be used for computing initial parameter estimates</td>
<td>PROC SEVSELECT</td>
<td>INITSAMPLE</td>
</tr>
<tr>
<td>Specifies the offset variable in the scale regression model</td>
<td>SCALEMODEL</td>
<td>OFFSET=</td>
</tr>
<tr>
<td>Specifies the denominator for computing covariance estimates</td>
<td>PROC SEVSELECT</td>
<td>VARDEF=</td>
</tr>
</tbody>
</table>

**Regression Effect Selection Options**

<table>
<thead>
<tr>
<th>Description</th>
<th>Statement</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specifies whether and how the model hierarchy requirement is applied</td>
<td>SELECTION</td>
<td>HIERARCHY=</td>
</tr>
<tr>
<td>Specifies the set of regression effects to be forced in all models</td>
<td>SCALEMODEL</td>
<td>INCLUDE=</td>
</tr>
<tr>
<td>Specifies the selection method</td>
<td>SELECTION</td>
<td>METHOD=</td>
</tr>
<tr>
<td>Specifies the selection order for displaying effects in the selected model</td>
<td>SELECTION</td>
<td>ORDERSELECT</td>
</tr>
<tr>
<td>Specifies the optimization technique to use for the intermediate selection steps</td>
<td>PROC SEVSELECT</td>
<td>SELECTNLOTECH=</td>
</tr>
<tr>
<td>Specifies the set of regression effects to begin the selection process</td>
<td>SCALEMODEL</td>
<td>START=</td>
</tr>
<tr>
<td>Specifies how to apply the STOP= criterion</td>
<td>SELECTION</td>
<td>STOPIPHORIZON=</td>
</tr>
</tbody>
</table>

**Regression Effect Selection Method Options**

<table>
<thead>
<tr>
<th>Description</th>
<th>Statement</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specifies a criterion for choosing the best model at each step</td>
<td>SELECTION</td>
<td>CHOOSE=</td>
</tr>
<tr>
<td>Specifies that the competitive form of the stepwise selection method be used</td>
<td>SELECTION</td>
<td>COMPETITIVE</td>
</tr>
<tr>
<td>Specifies the maximum number of effects in the model</td>
<td>SELECTION</td>
<td>MAXEFFECTS=</td>
</tr>
<tr>
<td>Specifies the maximum number of selection steps</td>
<td>SELECTION</td>
<td>MAXSTEPS=</td>
</tr>
<tr>
<td>Specifies the minimum number of effects in the model</td>
<td>SELECTION</td>
<td>MINEFFECTS=</td>
</tr>
<tr>
<td>Specifies a criterion to decide which effects enter or leave the model at each step</td>
<td>SELECTION</td>
<td>SELECT=</td>
</tr>
</tbody>
</table>
### Table 6.1  continued

<table>
<thead>
<tr>
<th>Description</th>
<th>Statement</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specifies a criterion for stopping the selection process</td>
<td>SELECTION</td>
<td>STOP=</td>
</tr>
<tr>
<td><strong>Empirical Distribution Function (EDF) Estimation Options</strong></td>
<td>PROC SEVSELECT</td>
<td>EMPIRICALCDF=</td>
</tr>
<tr>
<td>Specifies the nonparametric method of CDF estimation</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>EMPIRICALCDF=MODIFIEDKM Options</strong></td>
<td>PROC SEVSELECT</td>
<td>ALPHA=</td>
</tr>
<tr>
<td>Specifies the $\alpha$ value for the lower bound on risk set size</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specifies the $c$ value for the lower bound on risk set size</td>
<td>PROC SEVSELECT</td>
<td>C=</td>
</tr>
<tr>
<td>Specifies the absolute lower bound on risk set size</td>
<td>PROC SEVSELECT</td>
<td>RSLB=</td>
</tr>
<tr>
<td><strong>EMPIRICALCDF=TURNBULL Options</strong></td>
<td>PROC SEVSELECT</td>
<td>ENSUREMLE</td>
</tr>
<tr>
<td>Specifies that the final EDF estimates be maximum likelihood estimates</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specifies the relative convergence criterion</td>
<td>PROC SEVSELECT</td>
<td>EPS=</td>
</tr>
<tr>
<td>Specifies the maximum number of iterations</td>
<td>PROC SEVSELECT</td>
<td>MAXITER=</td>
</tr>
<tr>
<td>Specifies the threshold below which an EDF estimate is deemed to be 0</td>
<td>PROC SEVSELECT</td>
<td>ZEROPROB=</td>
</tr>
<tr>
<td><strong>OUT= Data Table Generation Options</strong></td>
<td>OUTPUT</td>
<td>COPYVARS=</td>
</tr>
<tr>
<td>Specifies the variables to copy from the DATA= data table to the OUT= data table</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specifies the scoring functions to estimate</td>
<td>OUTPUT</td>
<td>FUNCTIONS=</td>
</tr>
<tr>
<td>Specifies the quantiles to estimate</td>
<td>OUTPUT</td>
<td>QUANTILES=</td>
</tr>
<tr>
<td><strong>OUTEST= Data Table Generation Options</strong></td>
<td>PROC SEVSELECT</td>
<td>COVOUT</td>
</tr>
<tr>
<td>Specifies that the OUTEST= data table contain covariance estimates</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specifies that only the selected regression parameters be written</td>
<td>PROC SEVSELECT</td>
<td>SELECTOUT</td>
</tr>
<tr>
<td>Specifies that estimates of parameters not in the final model be 0</td>
<td>PROC SEVSELECT</td>
<td>ZEROEST</td>
</tr>
<tr>
<td><strong>Displayed Output Options</strong></td>
<td>PROC SEVSELECT</td>
<td>CRITERION=</td>
</tr>
<tr>
<td>Specifies the criterion to report in the model selection table</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specifies the level of detail to be produced about the selection process</td>
<td>SELECTION</td>
<td>DETAILS=</td>
</tr>
<tr>
<td>Limits or suppresses the display of class levels</td>
<td>PROC SEVSELECT</td>
<td>NOCLPRINT</td>
</tr>
<tr>
<td>Suppresses all displayed and graphical output</td>
<td>PROC SEVSELECT</td>
<td>NOPRINT</td>
</tr>
<tr>
<td>Specifies which output to display</td>
<td>PROC SEVSELECT</td>
<td>PRINT=</td>
</tr>
</tbody>
</table>
Chapter 6: The SEVSELECT Procedure

PROC SEVSELECT Statement

PROC SEVSELECT options;

The PROC SEVSELECT statement invokes the procedure. You can specify two types of options in the PROC SEVSELECT statement. One set of options controls input and output. The other set of options controls the model estimation and selection process.

The following options control the input data tables used by PROC SEVSELECT and various forms of output generated by PROC SEVSELECT. The options are listed in alphabetical order.

COVOUT

specifies that the OUTEST= data table contain the estimate of the covariance structure of the parameters. This option has no effect if you do not specify the OUTEST= option. For more information about how the covariance is reported in the OUTEST= data table, see the section “OUTEST= Data Table” on page 292.

CRITERION=criterion-option
CRITERIA=criterion-option
CRIT=criterion-option

specifies the criterion to report in the model selection table.

If you specify two or more candidate models for estimation, then the one with the best value for the selection criterion is marked as the best model in the model selection table that is displayed by default or when you specify the PRINT=SELECTION option.

You can specify one of the following criterion-options:

AD specifies the Anderson-Darling (AD) statistic value, which is computed by using the empirical distribution function (EDF) estimate, as the selection criterion. A lower value is deemed better.

AIC specifies Akaike's information criterion (AIC) as the selection criterion. A lower value is deemed better.

AICC specifies the finite-sample corrected Akaike's information criterion (AICC) as the selection criterion. A lower value is deemed better.

BIC specifies the Schwarz Bayesian information criterion (BIC) as the selection criterion. A lower value is deemed better.

CVM specifies the Cramér–von Mises (CVM) statistic value, which is computed by using the empirical distribution function (EDF) estimate, as the selection criterion. A lower value is deemed better.

KS specifies the Kolmogorov-Smirnov (KS) statistic value, which is computed by using the empirical distribution function (EDF) estimate, as the selection criterion. A lower value is deemed better.

LOGLIKELIHOOD | LL specifies \(-2 \times \log(L)\) as the selection criterion, where \(L\) is the likelihood of the data. A lower value is deemed better. This is the default.

For more information about these criterion-options, see the section “Statistics of Fit” on page 267.
PROC SEVSELECT Statement

DATA=CAS-libref.data-table
names the input data table for PROC SEVSELECT 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 203.

data-table specifies the name of the input data table.

INEST=CAS-libref.data-table
names the input data table that contains the initial values of the parameter estimates to start the
optimization process. *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 input data table. For more information
about this two-level name, see the DATA= option and the section “Using CAS Sessions and CAS
Engine Librefs” on page 203. The CAS-libref must be identical to the CAS-libref that you specify in
the DATA= option.

The initial values that you specify in the INIT= option in the DIST statement take precedence over any
initial values that you specify in the INEST= data table. For more information about the variables in
this data table, see the section “INEST= Data Table” on page 291.

NOCCLPRINT<=number>
suppresses the display of the “Class Level Information” table if you do not specify *number*. If you
specify *number*, the values of the classification variables are displayed for only those variables whose
number of levels is less than *number*. Specifying a *number* helps reduce the size of the “Class Level
Information” table if some classification variables have a large number of levels. This option has no
effect if you do not specify the CLASS statement.

NOPRINT
turns off all displayed and graphical output. If you specify this option, then any value that you specify
for the PRINT= option is ignored.

OUTEST=CAS-libref.data-table
names the output data table to contain estimates of the parameter values and their standard errors for
each model whose parameter estimation process converges. *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 203. The CAS-libref must be identical
to the CAS-libref that you specify in the DATA= option.

For more information about the variables in this data table, see the section “OUTEST= Data Table” on
page 292.

PRINT <(global-display-option)><=display-option>
PRINT <(global-display-option)><=(display-option . . .display-option)>
specifies the desired displayed output. If you specify more than one *display-option*, then separate them
with spaces and enclose them in parentheses.

You can specify the following *global-display-option*:
ONLY

turns off the default displayed output and displays only the requested output.

You can specify the following display-options:

ALL

displays all the output.

ALLFITSTATS

displays the comparison of all the statistics of fit for all the models in one table. The table does not include the models whose parameter estimation process does not converge.

CONVSTATUS

displays the convergence status of the parameter estimation process.

DESCSTATS

displays the descriptive statistics for the response variable. If you specify the SCALEMODEL statement, then this option also displays the descriptive statistics for the regression effects that do not contain a CLASS variable.

DISTINFO

displays the information about each specified distribution. For each distribution, the information includes the name, description, validity status, and number of distribution parameters.

ESTIMATES | PARMEST

displays the final estimates of parameters. The estimates are not displayed for models whose parameter estimation process does not converge.

ESTIMATIONDETAILS

displays the details of the estimation process for all the models in one table.

INITIALVALUES

displays the initial values and bounds used for estimating each model.

NLOHISTORY

displays the iteration history of the nonlinear optimization process used for estimating the parameters.

NLOSUMMARY

displays the summary of the nonlinear optimization process used for estimating the parameters.

NONE

displays none of the output. If you specify this option, then it overrides all other display options. The default displayed output is also suppressed.

SELECTION | SELECT

displays the model selection table.

STATISTICS | FITSTATS

displays the statistics of fit for each model. The statistics of fit are not displayed for models whose parameter estimation process does not converge.

If you do not specify the PRINT= option or if you do not specify the ONLY global-display-option, then the default displayed output is equivalent to specifying PRINT=(SELECTION CONVSTATUS NLOSUMMARY STATISTICS ESTIMATES).
PROC SEVSELECT Statement ♦ 221

SELECTOUT
specifies that only the regression parameters that correspond to the selected effects be written to the OUTEST= data table. If you omit this option, then the special missing value .N is written for such regression parameters. For more information, see the section “OUTEST= Data Table” on page 292.

This option has no effect if you do not specify the SCALEMODEL and SELECTION statements.

VARDEF=DF | N
specifies the denominator to use for computing the covariance estimates. You can specify one of the following values:

DF specifies that the number of nonmissing observations minus the model degrees of freedom (number of parameters) be used.

N specifies that the number of nonmissing observations be used.

For more information about the covariance estimation, see the section “Estimating Covariance and Standard Errors” on page 254.

ZEROEST
specifies that zero be written to the OUTEST= data table as an estimate of the regression parameter that is not in the final model (because it is either collinear or not selected). If you omit this option, then the special missing value .R or .N is written for such regression parameters. For more information, see the section “OUTEST= Data Table” on page 292.

This option has no effect if you do not specify the SCALEMODEL statement.

The following options control the model estimation and selection process:

EMPIRICALCDF | EDF=method
specifies the method to use for computing the nonparametric or empirical estimate of the cumulative distribution function of the data. You can specify one of the following values for method:

AUTOMATIC | AUTO
specifies that the method be chosen automatically based on the data specification.

If you do not specify any censoring or truncation, then the standard empirical estimation method (STANDARD) is chosen. If you specify both right-censoring and left-censoring, then Turnbull’s estimation method (TURNBULL) is chosen. For all other combinations of censoring and truncation, the Kaplan-Meier method (KAPLANMEIER) is chosen.

KAPLANMEIER | KM
specifies that the product limit estimator proposed by Kaplan and Meier (1958) be used. Specification of this method has no effect when you specify both right-censoring and left-censoring.

MODIFIEDKM | MKM <(options)>
specifies that the modified product limit estimator be used. Specification of this method has no effect when you specify both right-censoring and left-censoring.

This method allows Kaplan-Meier’s product limit estimates to be more robust by ignoring the contributions to the estimate due to small risk-set sizes. The risk set is the set of observations at the risk of failing, where an observation is said to fail if it has not been processed yet and might experience censoring or truncation. You can specify the minimum risk-set size that makes it
eligible to be included in the estimation either as an absolute lower bound on the size (RSLB= option) or a relative lower bound determined by the formula \( cn^\alpha \) proposed by Lai and Ying (1991). You can specify the values of \( c \) and \( \alpha \) by using the C= and ALPHA= options, respectively. By default, the relative lower bound is used with values of \( c = 1 \) and \( \alpha = 0.5 \). However, you can modify the default by using the following options:

**ALPHA | A=number**
- specifies the value to use for \( \alpha \) when the lower bound on the risk set size is defined as \( cn^\alpha \). This value must satisfy \( 0 < \alpha < 1 \).

**C=number**
- specifies the value to use for \( c \) when the lower bound on the risk set size is defined as \( cn^\alpha \). This value must satisfy \( c > 0 \).

**RSLB=number**
- specifies the absolute lower bound on the risk set size to be included in the estimate.

**NOTURNBULL**
- specifies that the method be chosen automatically based on the data specification and that Turnbull’s method not be used. This option is the default.

This method first replaces each left-censored or interval-censored observation with an uncensored observation. If the resulting set of observations has any truncated or right-censored observations, then the Kaplan-Meier method (KAPLANMEIER) is chosen. Otherwise, the standard empirical estimation method (STANDARD) is chosen. The observations are modified only for the purpose of computing the EDF estimates; the modification does not affect the parameter estimation process.

**STANDARD | STD**
- specifies that the standard empirical estimation method be used. If you specify both right-censoring and left-censoring, then the specification of this method has no effect. If you specify any other combination of censoring or truncation effects, then this method ignores such effects, and can thus result in estimates that are more biased than those obtained with other methods that are more suitable for censored or truncated data.

**TURNBULL | EM <options>**
- specifies that the Turnbull’s method be used. This method is used when you specify both right-censoring and left-censoring. An iterative expectation-maximization (EM) algorithm proposed by Turnbull (1976) is used to compute the empirical estimates. If you also specify truncation, then the modification suggested by Frydman (1994) is used.

This method is used if you specify both right-censoring and left-censoring and if you explicitly specify the EMPIRICALCDF=TURNBULL option.

You can modify the default behavior of the EM algorithm by using the following options:

**ENSUREMLE**
- specifies that the final EDF estimates be maximum likelihood estimates. The Kuhn-Tucker conditions are computed for the likelihood maximization problem and checked to ensure that EM algorithm converges to maximum likelihood estimates. The method generalizes the method proposed by Gentleman and Geyer (1994) by taking into account any truncation information that you might specify.
EPS=number
specifies the maximum relative error to be allowed between estimates of two consecutive iterations. This criterion is used to check the convergence of the algorithm. If you do not specify this option, then PROC SEVSELECT uses a default value of 1.0E–8.

MAXITER=number
specifies the maximum number of iterations to attempt to find the empirical estimates. If you do not specify this option, then PROC SEVSELECT uses a default value of 500.

ZEROPROB=number
specifies the threshold below which an empirical estimate of the probability is considered zero. This option is used to decide if the final estimate is a maximum likelihood estimate. This option does not have an effect if you do not specify the ENSUREMLE option. If you specify the ENSUREMLE option, but do not specify this option, then PROC SEVSELECT uses a default value of 1.0E–8.

For more information about each of the methods, see the section “Empirical Distribution Function Estimation Methods” on page 261.

INITSAMPLE (initsample-option)
INITSAMPLE (initsample-option . . . initsample-option)
specifies that a sample of the input data be used for initializing the distribution parameters. If you specify more than one initsample-option, then separate them with spaces.

When you do not specify initial values for the distribution parameters, PROC SEVSELECT needs to compute the empirical distribution function (EDF) estimates as part of the default method for parameter initialization. The EDF estimation process can be expensive, especially when you specify censoring or truncation effects for the loss variable. Furthermore, it is not amenable to parallelism due to the sequential nature of the algorithm for truncation effects. You can use the INITSAMPLE option to specify that only a fraction of the input data be used in order to reduce the time taken to compute the EDF estimates. PROC SEVSELECT uses the uniform random sampling method to select the sample, the size and randomness of which are controlled by the following initsample-options:

FRACTION=number
specifies the fraction, between 0 and 1, of the input data to be used for sampling.

SEED=number
specifies the seed to be used for the uniform random number generator. This option enables you to select the same sample from the same input data across different runs of PROC SEVSELECT, which can be useful for replicating the results across different runs. If you do not specify the seed value, PROC SEVSELECT generates a seed that is based on the system clock.

SIZE=number
specifies the size of the sample. If the data are distributed across different nodes, then this size applies to the sample that is prepared at each node. For example, let the input data table of size 100,000 observations be distributed across 10 nodes such that each node has 10,000 observations. If you specify SIZE=1000, then each node computes a local EDF estimate by using a sample of size 1,000 selected randomly from its 10,000 observations. If you specify both of the SIZE= and FRACTION= options, then the value that you specify in the SIZE= option is used and the FRACTION= option is ignored.
If you do not specify the INITSAMPLE option, then a uniform random sample of at most 10,000 observations is used for EDF estimation on each node that has observations.

**SELECTNL$OTECH**=CONGRA | DBLDOG | NEWRAP | NMSIMP | NRRIDG | QUANEW | TRUREG

specifies the nonlinear optimization technique to use for the intermediate steps of the regression effect selection process. The can specify one of the following techniques:

- **CONGRA** performs a conjugate-gradient optimization.
- **DBLDOG** performs a version of double-dogleg optimization.
- **NEWRAP** performs a Newton-Raphson optimization that combines a line-search algorithm with ridging.
- **NMSIMP** performs a Nelder-Mead simplex optimization.
- **NRRIDG** performs a Newton-Raphson optimization with ridging.
- **QUANEW** performs a dual quasi-Newton optimization.
- **TRUREG** performs a trust region optimization.

This option has no effect if you do not specify the SELECTION statement.

If you omit this option, then PROC SEVSELECT uses the dual quasi-Newton optimization method (QUANEW) to estimate the parameters of the intermediate models in the regression effect selection process.

This option does not affect the nonlinear optimization method that PROC SEVSELECT uses to estimate the parameters of the final selected model. You can control that method by specifying the **TECHNIQUE=** option in the NLOPTIONS statement.

---

**BY Statement**

**BY** variable-list ;

A BY statement can be used in the SEVSELECT procedure to process the input data table in groups of observations defined by the BY variables.

If you specify the BY statement, then unlike procedures that do not use a CAS server, PROC SEVSELECT does not need the input data table to be sorted in the order of the BY variables. As a consequence, the NOTSORTED and DESCENDING options have no effect and PROC SEVSELECT ignores them.

---

**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 6.2 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 4 in Chapter 1, “Shared Concepts.”
### Table 6.2  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>

#### DISPLAY Statement

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

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

- **EXCLUDE**
  - displays all display tables except those specified in the **table-list**.

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

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

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 display tables whose paths end in the specified name are selected for display or exclusion. For example, both `SelectionSummary` and `Summary.SelectionSummary` select `Bygroup1.Summary.SelectionSummary`. 
A regular expression is enclosed in “/”. 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.

**DIST Statement**

```
DIST distribution-name-or-keyword < (distribution-option) < distribution-name-or-keyword < (distribution-option) > > . . . > < / options > ;
```

The DIST statement specifies candidate distributions to be estimated by the SEVSELECT procedure. You can specify multiple DIST statements, and each statement can contain one or more distribution specifications.

For your convenience, PROC SEVSELECT provides the following 10 different predefined distributions (the name in parentheses is the name to use in the DIST statement): Burr (Burr), exponential (EXP), gamma (Gamma), generalized Pareto (GPD), inverse Gaussian or Wald (IGAUSS), lognormal (LOGN), Pareto (PARETO), Tweedie (TWEEDIE), scaled Tweedie (STWEEDIE), and Weibull (WEIBULL). These are described in detail in the section “Predefined Distributions” on page 241.

You can specify any of the predefined distributions or any distribution that you have defined. If a distribution that you specify is not a predefined distribution, then you must submit the CMPLIB= system option with appropriate libraries before you submit the PROC SEVSELECT step to enable the procedure to find the
functions associated with your distribution. The predefined distributions are defined in the Sashelp.Svrtdist library. However, you are not required to specify this library in the CMPLIB= system option. For more information about defining your own distributions, see the section “Defining a Severity Distribution Model with the FCMP Procedure” on page 272.

As a convenience, you can also use a shortcut keyword to indicate a list of distributions. You can specify one or more of the following keywords:

ALCHEMY specifies all the predefined distributions and the distributions that you have defined in the libraries that you specify in the CMPLIB= system option. In addition to the eight predefined distributions included by the _PREDEFINED_ keyword, this list also includes the Tweedie and scaled Tweedie distributions that are defined in the Sashelp.Svrtdist library.

_PREDEFINED_ specifies the list of eight predefined distributions: BURR, EXP, GAMMA, GPD, IGAUSS, LOGN, PARETO, and WEIBULL. Although the TWEEDIE and STWEEDIE distributions are available in the Sashelp.Svrtdist library along with these eight distributions, they are not included by this keyword. If you want to fit the TWEEDIE and STWEEDIE distributions, then you must specify them explicitly or use the _ALL_ keyword.

_USER_ specifies the list of all the distributions that you have defined in the libraries that you specify in the CMPLIB= system option. This list does not include the distributions defined in the Sashelp.Svrtdist library, even if you specify the Sashelp.Svrtdist library in the CMPLIB= option.

The use of these keywords, especially _ALL_, can result in a large list of distributions, which might take a longer time to estimate. A warning is printed to the SAS log if the number of total distribution models to estimate exceeds 10.

The following distribution-option values can be used in the DIST statement for a distribution name that is not a shortcut keyword:

INIT=(name=value ... name=value) specifies the initial values to be used for the distribution parameters to start the parameter estimation process. You must specify the values by parameter names, and the parameter names must match the names used in the model definition. For example, let a model M’s definition contain an M_PDF function with the following signature:

    function M_PDF(x, alpha, beta);

For this model, the names alpha and beta must be used for the INIT option. The names are case-insensitive. If you do not specify initial values for some parameters in the INIT statement, then a default value of 0.001 is assumed for those parameters. If you specify an incorrect parameter, PROC SEVSELECT prints a warning to the SAS log and does not fit the model. All specified values must be nonmissing.

If you are modeling regression effects, then the initial value of the first distribution parameter (alpha in the preceding example) should be the initial base value of the scale parameter or log-transformed scale parameter. For more information, see the section “Estimating Regression Effects” on page 256.
The use of INIT= option is one of the three methods available for initializing the parameters. For more information, see the section “Parameter Initialization” on page 255. If none of the initialization methods is used, then PROC SEVSELECT initializes all parameters to 0.001.

You can specify the following options in the DIST statement:

- **INFUNCDEF=** "CAS-table-name"
- **INTAB=** "CAS-table-name"
- **INFUNCDEF=** CAS-table-name < (CASLIB="caslib") >
- **INTAB=** CAS-table-name < (CASLIB="caslib") >

  specifies a data table on the CAS server that contains the distribution function definitions. If you use the first form to specify the data table, then PROC SEVSELECT assumes that the table is in the caslib that is active in the current CAS session. To specify a different caslib, use the second form.

- **LISTONLY**
  prints the list of all candidate distributions to the SAS log without doing any further processing on them. This option is especially useful when you use a shortcut keyword to include a list of distributions. It enables you to find out which distributions are included by the keyword.

- **OUTFUNCDEF=** "CAS-table-name" < option >
- **OUTTAB=** "CAS-table-name" < option >

  specifies the output data table on the CAS server to write the distribution function definitions to. The table is created in the caslib that is active for the current CAS session.

  This option is useful for creating a data table that you can use to invoke the actions in the severity action set from the PROC CAS, Lua, or Python client.

  You can specify the following option:

  - **GLOBAL**
  - **PROMOTED**

    promotes the data table to make it available globally.

    If you do not specify this option, the table is accessible only to the current CAS session that is executing the PROC SEVSELECT step. When the current CAS session is terminated, the OUTFUNCDEF= table is deleted.

    When you specify this option, the table persists in the memory of the CAS server and is available to other CAS sessions that you might start after the CAS session that is executing the current PROC SEVSELECT step is terminated.

- **VALIDATEONLY**
- **VALIDATEONLY (NOSCALEMODEL)**

  checks all candidate distributions for validity without doing any further processing on them. The first form checks whether each distribution’s first parameter is a scale parameter, which is a requirement for fitting a scale regression model for that distribution. You can disable that check by using the second form, which specifies the NOSCALEMODEL option; this is useful if you do not want to fit a scale regression model for a distribution.

  PROC SEVSELECT writes the distribution information to the DistributionInfo ODS table.
This option is especially useful when you specify your own distributions or when you specify the _USER_ or _ALL_ keyword in the DIST statement. It enables you to check whether your custom distribution definitions satisfy PROC SEVSELECT’s requirements for the specified modeling task. It is recommended that you specify the SCALEMODEL statement if you intend to fit a model with regression effects, because the SCALEMODEL statement instructs PROC SEVSELECT to perform additional checks to validate whether regression effects can be modeled on each candidate distribution.

This option is also useful in conjunction with the OUTFUNCDEF= option when you define or update a custom distribution and want to write its definition to the CAS table that you can use to invoke the actions in the severity action set from the PROC CAS, Lua, or Python client.

**EFFECT Statement**

```plaintext
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 36 in Chapter 1, “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 6.3 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>
</tbody>
</table>
**Table 6.3 continued**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>WEIGHT=</strong></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 KNOTMIN= value and ending at the variables’ minimum value
- **NATURALCUBIC** Specifies a natural cubic spline basis for the spline effect
- **SEPARATE** Treats the spline basis for each variable as a separate effect when multiple variables are specified
- **SPLIT** Treats each design matrix column as a separate effect for selection methods

For more information about the syntax of these *effect-types* and how columns of constructed effects are computed, see the section “EFFECT Statement” on page 8 in Chapter 1, “Shared Concepts.”

---

**LOSS Statement**

```plaintext
LOSS <response-variable-name> < / censoring-truncation-options > ;
```

The LOSS statement specifies the name of the response or loss variable whose distribution needs to be modeled. You can also specify additional options to indicate any truncation or censoring of the response. The specification of response variable is optional if you specify at least one type of censoring. You must specify a response variable if you do not specify any censoring. If you specify more than one LOSS statement, then the first statement is used.
All the analysis variables that you specify in this statement must be present in the input data table that you specify by using the DATA= option in the PROC SEVSELECT statement. The response variable is expected to have nonmissing values. If the variable has a missing value in an observation, then a warning is written to the SAS log and that observation is ignored.

The following censoring-truncation-options can be used in the LOSS statement:

\textbf{LEFTCENSORED} | \texttt{LC=} \texttt{variable-name}  
\textbf{LEFTCENSORED} | \texttt{LC=} \texttt{number}  

specifies the left-censoring variable or a global left-censoring limit.

You can use the \texttt{variable-name} argument to specify a data table variable that contains the left-censoring limit. If the value of this variable is missing, then PROC SEVSELECT assumes that such observations are not left-censored.

Alternatively, you can use the \texttt{number} argument to specify a left-censoring limit value that applies to all the observations in the data table. This limit must be a nonzero positive number.

By the definition of left-censoring, an exact value of the response is not known when it is less than or equal to the left-censoring limit. If you specify the response variable and the value of that variable is less than or equal to the value of the left-censoring limit for some observations, then PROC SEVSELECT treats such observations as left-censored and the value of the response variable is ignored. If you specify the response variable and the value of that variable is greater than the value of the left-censoring limit for some observations, then PROC SEVSELECT assumes that such observations are not left-censored and the value of the left-censoring limit is ignored.

If you specify both right-censoring and left-censoring limits, then the left-censoring limit must be greater than or equal to the right-censoring limit. If both limits are identical, then the observation is assumed to be uncensored.

For more information about left-censoring, see the section “Censoring and Truncation” on page 251.

\textbf{LEFTTRUNCATED} | \texttt{LT=} \texttt{variable-name} < \texttt{(left-truncation-option)>}  
\textbf{LEFTTRUNCATED} | \texttt{LT=} \texttt{number} < \texttt{(left-truncation-option)>}  

specifies the left-truncation variable or a global left-truncation threshold.

You can use the \texttt{variable-name} argument to specify a data table variable that contains the left-truncation threshold. If the value of this variable is missing or 0 for some observations, then PROC SEVSELECT assumes that such observations are not left-truncated.

Alternatively, you can use the \texttt{number} argument to specify a left-truncation threshold that applies to all the observations in the data table. This threshold must be a nonzero positive number.

It is assumed that the response variable contains the observed values. By the definition of left-truncation, you can observe only a value that is greater than the left-truncation threshold. If a response variable value is less than or equal to the left-truncation threshold, a warning is printed to the SAS log, and the observation is ignored. For more information about left-truncation, see the section “Censoring and Truncation” on page 251.

You can specify the following left-truncation-option for an alternative interpretation of the left-truncation threshold:
Chapter 6: The SEVSELECT Procedure

PROBOBSERVED | POBS=number

specifies the probability of observability, which is defined as the probability that the underlying severity event is observed (and recorded) for the specified left-threshold value.

The specified number must lie in the (0.0, 1.0] interval. A value of 1.0 is equivalent to specifying that there is no left-truncation, because it means that no severity events can occur with a value less than or equal to the threshold. If you specify value of 1.0, PROC SEVSELECT prints a warning to the SAS log and proceeds by assuming that LEFTTRUNCATED= option is not specified.

For more information, see the section “Probability of Observability” on page 252.

RIGHTCENSORED | RC=variable-name

RIGHTCENSORED | RC=number

specifies the right-censoring variable or a global right-censoring limit.

You can use the variable-name argument to specify a data table variable that contains the right-censoring limit. If the value of this variable is missing, then PROC SEVSELECT assumes that such observations are not right-censored.

Alternatively, you can use the number argument to specify a right-censoring limit value that applies to all the observations in the data table. This limit must be a nonzero positive number.

By the definition of right-censoring, an exact value of the response is not known when it is greater than or equal to the right-censoring limit. If you specify the response variable and the value of that variable is greater than or equal to the value of the right-censoring limit for some observations, then PROC SEVSELECT treats such observations as right-censored and the value of the response variable is ignored. If you specify the response variable and the value of that variable is less than the value of the right-censoring limit for some observations, then PROC SEVSELECT assumes that such observations are not right-censored and the value of the right-censoring limit is ignored.

If you specify both right-censoring and left-censoring limits, then the left-censoring limit must be greater than or equal to the right-censoring limit. If both limits are identical, then the observation is assumed to be uncensored.

For more information about right-censoring, see the section “Censoring and Truncation” on page 251.

RIGHTTRUNCATED | RT=variable-name

RIGHTTRUNCATED | RT=number

specifies the right-truncation variable or a global right-truncation threshold.

You can use the variable-name argument to specify a data table variable that contains the right-truncation threshold. If the value of this variable is missing for some observations, then PROC SEVSELECT assumes that such observations are not right-truncated.

Alternatively, you can use the number argument to specify a right-truncation threshold that applies to all the observations in the data table. This threshold must be a nonzero positive number.

It is assumed that the response variable contains the observed values. By the definition of right-truncation, you can observe only a value that is less than or equal to the right-truncation threshold. If a response variable value is greater than the right-truncation threshold, a warning is printed to the SAS log, and the observation is ignored. For more information about right-truncation, see the section “Censoring and Truncation” on page 251.
**NLOPTINS Statement**

**NLOPTINS** options;  

The SEVSELECT procedure uses the nonlinear optimization (NLO) subsystem to perform nonlinear optimization of the likelihood function to obtain the estimates of distribution and regression parameters. You can use the NLOPTINS statement to control different aspects of this optimization process. If you specify more than one NLOPTINS statement, then the first statement is used.

For most problems, the default settings of the optimization process are adequate. However, in some cases it might be useful to change the optimization technique or to change the maximum number of iterations. The following statement uses the MAXITER= option to set the maximum number of iterations to 200 and uses the TECH= option to change the optimization technique to the double-dogleg optimization (DBLDOG) rather than the default technique, the trust region optimization (TRUREG), used in the SEVSELECT procedure:

```plaintext
nloptions tech=dbldog maxiter=200;
```

For more information about the options you can specify in the NLOPTINS statement, see the section “Optimization Options” on page 26. For more information about the optimization methods, see the section “Choosing an Optimization Algorithm” on page 50.

**OUTPUT Statement**

**OUTPUT** <**OUT**=CAS-libref.data-table> output-options;  

The OUTPUT statement creates the output data table that contains the estimates of scoring functions and quantiles.

You must specify the following option:

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

names the output data table for PROC SEVSELECT 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 203.

- **data-table** specifies the name of the output data table.

To control the contents of the OUT= data table, specify the following **output-options**:

**COPYVARS**=variable-list  

specifies the names of the variables that you want to copy from the input DATA= data table to the OUT= data table. If you want to specify more than one name, then separate them by spaces.

If you specify the BY statement, then the BY variables are not automatically copied to the OUT= data table, so you must specify the BY variables in the COPYVARS= option.
FUNCTIONS=(function<(arg)><=variable><function<(arg)><=variable>> ...) specifies the scoring functions that you want to estimate. For each scoring function that you want to estimate, specify the suffix of the scoring function as the function. For each function that you specify in this option and for each distribution $D$ that you specify in the DIST statement, the FCMP function $D\_function$ must be available in the search path that you specify by using the CMPLIB= system option. The signature of $D\_function$ must be identical to the signature of the required distribution function such as $D\_CDF$ or $D\_LOGCDF$. For example, for the function ‘FOO\_BAR’ to be a scoring function, you must specify the distribution ‘FOO’ in the DIST statement, and you must define ‘FOO\_BAR’ in the following manner if the distribution ‘FOO’ has parameters named ‘P1’ and ‘P2’:

```sas
function FOO\_BAR(y, P1, P2);
    /* Code to compute BAR by using y, P1, and P2 */
    R = <computed BAR>;
    return (R);
endsub;
```

You can then specify BAR as the function in the FUNCTIONS= option. For more information about the signature that defines a distribution function, see the description of the $D\_CDF$ function in the section “Defining a Severity Distribution Model with the FCMP Procedure” on page 272.

If you want to evaluate the scoring function at a specific value of the response variable, then specify a number arg, which is enclosed in parentheses immediately after the function. If you do not specify arg or if you specify a missing value as arg, then for each observation in the DATA= data table, PROC SEVSELECT computes the value $v$ by using the following table and evaluates the scoring function at $v$, where $y$, $r$, and $l$ denote the values of the response variable, right-censoring limit, and left-censoring limit, respectively:

<table>
<thead>
<tr>
<th>Right-Censored</th>
<th>Left-Censored</th>
<th>$v$</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>No</td>
<td>$y$</td>
</tr>
<tr>
<td>No</td>
<td>Yes</td>
<td>$l$</td>
</tr>
<tr>
<td>Yes</td>
<td>No</td>
<td>$r$</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>$(l + r)/2$</td>
</tr>
</tbody>
</table>

You can specify the suffix of the variable that contains the estimate of the scoring function by specifying a valid SAS name as a variable. If you do not specify a variable, then PROC SEVSELECT uses function as the suffix of the variable name.

To illustrate the FUNCTIONS= option with an example, assume that you specify the following DIST and OUTPUT statements:

```sas
dist exp logn;
output out=mycas.score functions=(cdf pdf(1000)=f1000 mean);
```

Let both exponential (EXP) and lognormal (LOGN) distributions converge. If $\hat{\theta}$ is the final estimate of the scale parameter of the exponential distribution, then PROC SEVSELECT creates the following three scoring function variables for the exponential (EXP) distribution in the mycas.Score data table:
EXP_CDF contains the CDF estimate \( F_{\text{exp}}(v, \hat{\theta}) \), where \( F_{\text{exp}} \) denotes the CDF of the exponential distribution and \( v \) is the value that is determined by the preceding table.

EXP_F1000 contains the PDF estimate \( f_{\text{exp}}(1000, \hat{\theta}) \), where \( f_{\text{exp}} \) denotes the PDF of the exponential distribution.

EXP_MEAN contains the mean of the exponential distribution for the scale parameter \( \hat{\theta} \).

Similarly, if \( \hat{\mu} \) and \( \hat{\sigma} \) are the final estimates of the log-scale and shape parameters of the lognormal distribution, respectively, then PROC SEVSELECT creates the following three scoring function variables for the lognormal (LOGN) distribution in the mycas.Score data table:

LOGN_CDF contains the CDF estimate \( F_{\text{logn}}(v, \hat{\mu}, \hat{\sigma}) \), where \( F_{\text{logn}} \) denotes the CDF of the lognormal distribution and \( v \) is the value that is determined by the preceding table.

LOGN_F1000 contains the probability density function (PDF) estimate \( f_{\text{logn}}(1000, \hat{\mu}, \hat{\sigma}) \), where \( f_{\text{logn}} \) denotes the PDF of the lognormal distribution.

LOGN_MEAN contains the mean of the lognormal distribution for the parameters \( \hat{\mu} \) and \( \hat{\sigma} \).

If you specify the SCALEMODEL statement, then the value of the scale parameter of a distribution depends on the values of the regression parameters. So it might be different for different observations. In this example, the values of \( \hat{\theta} \) and \( \hat{\mu} \) might vary by observation, which might cause the values of the EXP_F1000, EXP_MEAN, LOGN_F1000, and LOGN_MEAN variables to vary by observation. The values of the EXP_CDF and LOGN_CDF variables might vary not only because of the varying values of \( v \) but also because of the varying values of \( \hat{\theta} \) and \( \hat{\mu} \).

If you do not specify the SCALEMODEL statement, then the values of scoring functions for which you specify a nonmissing argument \( \text{arg} \) and scoring functions that do not depend on the response variable value do not vary by observation. In this example, the values of the EXP_F1000, EXP_MEAN, LOGN_F1000, and LOGN_MEAN variables do not vary by observation.

If a distribution does not converge, then the scoring function variables for that distribution contain missing values in all observations.

**QUANTILES=quantile-options**

specifies the quantiles that you want to estimate. To use this option, for each distribution that you specify in the DIST statement, the FCMP function \( D_{\text{QUANTILE}} \) must be defined in the search path that you specify by using the CMPLIB= system option.

You can specify the following quantile-options:

**CDF=CDF-values**

**POINTS=CDF-values**

specifies the CDF values at which you want to estimate the quantiles. \( \text{CDF-values} \) can be one or more numbers, separated by spaces. Each number must be in the interval \((0,1)\).

** NAMES=variable-names**

specifies the suffixes of the names of the variables for each of the quantile estimates. If you specify \( n \) (\( n \geq 0 \)) names in the variable-names option and \( k \) values in the CDF= option, and if \( n < k \), then PROC SEVSELECT uses the \( n \) names to name the variables that correspond to the first \( n \) CDF values. For each of the remaining \( k - n \) CDF values, \( p_i \) (\( n < i \leq k \)), PROC SEVSELECT creates a variable name \( P_t \), where \( t \) is the text representation of \( 100p_i \) that
Chapter 6: The SEVSELECT Procedure

is formed by retaining at most NDECIMAL= digits after the decimal point and replacing the decimal point with an underscore ('_').

**NDECIMAL=number**

specifies the number of digits to keep after the decimal point when PROC SEVSELECT creates the name of the quantile estimate variable. If you do not specify this option, then the default value is 3.

For example, assume that you specify the following DIST and OUTPUT statements:

```
dist burr;
output out=score quantiles=(cdf=0.9 0.975 0.995 names=ninety var);
```

PROC SEVSELECT creates three quantile estimate variables, BRR_NINETY, BRR_VAR, and BRR_P99_5, in the mycas.Score data table for the Burr distribution. These variables contain the estimates of \( Q_{\text{Burr}}(p, \hat{\theta}, \hat{\alpha}, \hat{\gamma}) \), for \( p = 0.9, 0.975 \), and 0.995, respectively, where \( Q_{\text{Burr}} \) denotes the quantile function and \( \hat{\theta}, \hat{\alpha}, \) and \( \hat{\gamma} \) denote the parameter estimates of the Burr distribution.

If you specify the SCALEMODEL statement, then the quantile estimate might vary by observation, because the scale parameter of a distribution depends on the values of the regression parameters.

If you do not specify the SCALEMODEL statement, then the quantile estimates do not vary by observation, and if you do not specify any scoring functions in the FUNCTIONS= option whose estimates vary by observation, then the OUT= data table contains only one observation per BY group.

If a distribution does not converge, then the quantile estimate variables for that distribution contain missing values for all observations.

For more information about the variables and observations in the OUT= data table, see the section “OUT= Data Table” on page 291.

### SCALEMODEL Statement

```
SCALEMODEL regression-effect-list </scalemodel-options>;
```

The SCALEMODEL statement specifies regression effects. A regression effect is formed from one or more regressor variables according to effect construction rules. Each regression effect forms one element of \( \mathbf{X} \) in the linear model structure \( \mathbf{X}\beta \) that affects the scale parameter of the distribution. The SCALEMODEL statement in conjunction with the CLASS statement supports a rich set of effects. Effects are specified by a special notation that uses regressor variable names and operators. There are two types of regressor variables: classification (or CLASS) variables and continuous variables. Classification variables can be either numeric or character and are specified in a CLASS statement. To include CLASS variables in regression effects, you must specify the CLASS statement so that it appears before the SCALEMODEL statement. A regressor variable that is not declared in the CLASS statement is assumed to be continuous. For information about effect construction rules, see the section “Specification and Parameterization of Model Effects” on page 33 in Chapter 1, “Shared Concepts.”

All the regressor variables must be present in the input data table that you specify by using the DATA= option in the PROC SEVSELECT statement. The scale parameter of each candidate distribution is linked to the linear predictor \( \mathbf{X}\beta \) that includes an intercept. If a distribution does not have a scale parameter, then a model
based on that distribution is not estimated. If you specify more than one SCALEMODEL statement, then the first statement is used.

The regressor variables are expected to have nonmissing values. If any of the variables has a missing value in an observation, then a warning is written to the SAS log and that observation is ignored.

For more information about modeling regression effects, see the section “Estimating Regression Effects” on page 256.

You can specify the following scalemodel-options in the SCALEMODEL statement:

DFMIXTURE=method-name < (method-options) >
specifies the method for computing representative estimates of the cumulative distribution function (CDF) and the probability density function (PDF).

When you specify regression effects, the scale of the distribution depends on the values of the regressors. For a particular distribution family, each observation in the input data table implies a different scaled version of the distribution. To compute estimates of CDF and PDF that are comparable across different distribution families, PROC SEVSELECT needs to construct a single representative distribution from all such distributions. You can specify one of the following method-name values to specify the method that is used to construct the representative distribution. For more information about each of the methods, see the section “CDF and PDF Estimates with Regression Effects” on page 259.

FULL
specifies that the representative distribution be the mixture of N distributions such that each distribution has a scale value that is implied by each of the N observations that are used for estimation. This method is the slowest.

MEAN
specifies that the representative distribution be the one-point mixture of the distribution whose scale value is computed by using the mean of the N values of the linear predictor that are implied by the N observations that are used for estimation. If you do not specify the DFMIXTURE= option, then this method is used by default. This is also the fastest method.

QUANTILE < (K=q) >
specifies that the representative distribution be the mixture of a fixed number of distributions whose scale values are computed by using the quantiles from the sample of N values of the linear predictor that are implied by the N observations that are used for estimation.

You can use the K= option to specify the number of distributions in the mixture. If you specify K=q, then the mixture contains (q−1) distributions such that each distribution has as its scale one of the (q−1)-quantiles.

If you do not specify the K= option, then PROC SEVSELECT uses the default of 2, which implies the use of a one-point mixture with a distribution whose scale value is the median of all scale values.

RANDOM < (random-method-options) >
specifies that the representative distribution be the mixture of a fixed number of distributions whose scale values are computed by using the values of the linear predictor that are implied by a randomly chosen subset of the set of all observations that are used for estimation. The same subset of observations is used for each distribution family.

You can specify the following random-method-options to specify how the subset is chosen:
\(K=r\)
specifies the number of distributions to include in the mixture. If you do not specify this option, then PROC SEVSELECT uses the default of 15.

\[\text{SEED}=\text{number}\]
specifies the seed that is used to generate the uniform random sample of observation indices. If you do not specify this option, then PROC SEVSELECT generates a seed internally that is based on the current value of the system clock.

\[\text{INCLUDE}=n\]
\[\text{INCLUDE}=\text{single-effect}\]
\[\text{INCLUDE}=(\text{effects})\]
forces effects to be included in all models. If you specify INCLUDE=\(n\), then the first \(n\) effects that are listed in the SCALEMODEL statement are included in all models. If you specify INCLUDE=\(\text{single-effect}\) or INCLUDE=(\text{single-effect}), then the specified effects are forced into all models. The INCLUDE= option has no effect if you do not specify 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 \(A\times x\), 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 48 in Chapter 1, “Shared Concepts.”

\[\text{OFFSET}=\text{offset-variable-name}\]
specifies the name of the offset variable in the scale regression model. An offset variable is a regressor variable whose regression coefficient is known to be 1. For more information, see the section “Offset Variable” on page 257.

\[\text{START}=n\]
\[\text{START}=\text{single-effect}\]
\[\text{START}=(\text{effects})\]
specifies how to begin the selection process in the FORWARD, FORWARDSWAP, and STEPWISE selection methods from the initial model that you designate. If you specify START=\(n\), then the starting model consists of the first \(n\) effects that are listed in the SCALEMODEL statement. If you specify START=\(\text{single-effect}\) or START=(\text{single-effect}), then the starting model consists of these specified effects.

The START= option has no effect if you do not specify the SELECTION statement. This option is not supported when you specify METHOD=BACKWARD in the SELECTION statement.

\underline{SELECTION Statement}

\[\text{SELECTION < METHOD}=\text{method} < (\text{method-options}) >> <\text{options}> ;\]
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 the selection methods. The statement is fully documented in the section “SELECTION Statement” on page 23 in Chapter 1, “Shared Concepts.” The SEVSELECT procedure supports only a subset of the method values and method-options that are listed in that section.

**NOTE:** In the context of the SELECTION statement, the term model refers to the set of regression effects in the scale regression model. The final severity model consists of the set of distribution parameters and the set of selected regression effects.

You can use the following option to specify the selection method:

**METHOD=BACKWARD | FORWARD | FORWARDSWAP | STEPWISE**

specifies the selection method to use. You can specify one of the following methods:

- **BACKWARD** specifies the backward elimination method.
- **FORWARD** specifies the forward selection method.
- **FORWARDSWAP** specifies forward-swap selection, which is an extension of the forward selection method.
- **STEPWISE** specifies the stepwise regression method.

By default, METHOD=STEPWISE. For more information about each of the selection methods, see the section “Model Selection Methods” on page 44.

The SEVSELECT procedure supports a specific set of values for the following method-options:

**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. The SEVSELECT procedure supports only the following values for criterion:

- **AIC** specifies Akaike’s information criterion.
- **AICC** specifies the corrected Akaike’s information criterion.
- **SBC** specifies the Schwarz Bayesian information criterion.

By default, the CHOOSE= criterion is the same as the SELECT= criterion. For more information, see the detailed description of the CHOOSE= option.

**SELECT=criterion**

specifies the criterion that the procedure uses to determine the order in which effects enter or leave at each step of the selection method. The SEVSELECT procedure supports the following values for criterion:

- **AIC** specifies Akaike’s information criterion.
- **AICC** specifies the corrected Akaike’s information criterion.
- **SBC** specifies the Schwarz Bayesian information criterion.

By default, SELECT=SBC. For more information, see the detailed description of the SELECT= option.
STOP=criterion

specifies a criterion that is used to stop the selection process. The SEVSELECT procedure supports only the following values for criterion:

- **AIC** specifies Akaike’s information criterion.
- **AICC** specifies the corrected Akaike’s information criterion.
- **SBC** specifies the Schwarz Bayesian information criterion.

By default, the STOP= criterion is the same as the SELECT= criterion. For more information, see the detailed description of the STOP= option.

The SEVSELECT procedure supports the following method-options as they are described in the section “SELECTION Statement” on page 23 in Chapter 1, “Shared Concepts”:

- COMPETITIVE
- MAXEFFECTS= 
- MAXSTEPS= 
- MINEFFECTS=

The SEVSELECT procedure supports the following statement options as they are described in the section “SELECTION Statement” on page 23 in Chapter 1, “Shared Concepts”:

- DETAILS= 
- HIERARCHY= 
- ORDERSELECT 
- STOPHORIZON=

### WEIGHT Statement

**WEIGHT** variable-name ;

The WEIGHT statement specifies the name of a variable whose values represent the weight of each observation. PROC SEVSELECT associates a weight of \( w \) to each observation, where \( w \) is the value of the WEIGHT variable for the observation. If the weight value is missing or less than or equal to 0, then the observation is ignored and a warning is written to the SAS log. When you do not specify the WEIGHT statement, each observation is assigned a weight of 1. If you specify more than one WEIGHT statement, then the last statement is used.

The weights are normalized so that they add up to the actual sample size. In particular, the weight of each observation is multiplied by \( \frac{N}{\sum_{i=1}^{N} w_i} \), where \( N \) is the sample size. All computations, including the computations of the EDF-based statistics of fit, use normalized weights.
Details: SEVSELECT Procedure

Predefined Distributions

For the response variable \( Y \), PROC SEVSELECT assumes the model

\[
Y \sim \mathcal{F}(\Theta)
\]

where \( \mathcal{F} \) is a continuous probability distribution with parameters \( \Theta \). The model hypothesizes that the observed response is generated from a stochastic process that is governed by the distribution \( \mathcal{F} \). This model is usually referred to as the error model. Given a representative input sample of response variable values, PROC SEVSELECT estimates the model parameters for any distribution \( \mathcal{F} \) and computes the statistics of fit for each model. This enables you to find the distribution that is most likely to generate the observed sample.

A set of predefined distributions is provided with the SEVSELECT procedure. A summary of the distributions is provided in Table 6.4. For each distribution, the table lists the name of the distribution that should be used in the DIST statement, the parameters of the distribution along with their bounds, and the mathematical expressions for the probability density function (PDF) and cumulative distribution function (CDF) of the distribution.

All the predefined distributions, except LOGN and TWEEDIE, are parameterized such that their first parameter is the scale parameter. For LOGN, the first parameter \( \mu \) is a log-transformed scale parameter. TWEEDIE does not have a scale parameter. The presence of scale parameter or a log-transformed scale parameter enables you to use all of the predefined distributions, except TWEEDIE, as a candidate for estimating regression effects.

A distribution model is associated with each predefined distribution. You can also define your own distribution model, which is a set of functions and subroutines that you define by using the FCMP procedure. For more information, see the section “Defining a Severity Distribution Model with the FCMP Procedure” on page 272.
## Table 6.4  Predefined PROC SEVSELECT Distributions

<table>
<thead>
<tr>
<th>Name</th>
<th>Distribution</th>
<th>Parameters</th>
<th>PDF ((f)) and CDF ((F))</th>
</tr>
</thead>
<tbody>
<tr>
<td>BURR</td>
<td>Burr</td>
<td>(\theta &gt; 0, \alpha &gt; 0,\gamma &gt; 0)</td>
<td>(f(x) = \frac{\alpha x^\gamma}{(1+z)^{(\alpha+1)}}) (F(x) = 1 - \left(\frac{1}{1+z}\right)^\alpha)</td>
</tr>
<tr>
<td>EXP</td>
<td>Exponential</td>
<td>(\theta &gt; 0)</td>
<td>(f(x) = \frac{1}{\theta} e^{-x}) (F(x) = 1 - e^{-x})</td>
</tr>
<tr>
<td>GAMMA</td>
<td>Gamma</td>
<td>(\theta &gt; 0, \alpha &gt; 0)</td>
<td>(f(x) = \frac{\alpha e^{-x}}{\Gamma(\alpha) \gamma(\alpha x)}) (F(x) = \frac{1}{\Gamma(\alpha)} e^{-x})</td>
</tr>
<tr>
<td>GPD</td>
<td>Generalized Pareto</td>
<td>(\theta &gt; 0, \xi &gt; 0)</td>
<td>(f(x) = \frac{1}{\theta} (1 + \xi x)^{-1-1/\xi}) (F(x) = 1 - (1 + \xi x)^{-1/\xi})</td>
</tr>
<tr>
<td>IGAUSS</td>
<td>Inverse Gaussian (Wald)</td>
<td>(\theta &gt; 0, \alpha &gt; 0)</td>
<td>(f(x) = \frac{1}{\theta} \sqrt{\frac{\alpha}{2\pi x^3}} e^{-\frac{(x-1)^2}{2\xi}}) (F(x) = \Phi\left((z-1)\sqrt{\frac{\alpha}{2}}\right) + \Phi\left(-(z+1)\sqrt{\frac{\alpha}{2}}\right) e^{2\alpha})</td>
</tr>
<tr>
<td>LOGN</td>
<td>Lognormal</td>
<td>(\mu) (no bounds), (\sigma &gt; 0)</td>
<td>(f(x) = \frac{1}{x \sigma \sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\log(x)-\mu}{\sigma}\right)^2}) (F(x) = \Phi\left(\frac{\log(x)-\mu}{\sigma}\right))</td>
</tr>
<tr>
<td>PARETO</td>
<td>Pareto</td>
<td>(\theta &gt; 0, \alpha &gt; 0)</td>
<td>(f(x) = \frac{\alpha x^\alpha}{(x+\theta)^{\alpha+1}}) (F(x) = 1 - \left(\frac{\theta}{x+\theta}\right)^\alpha)</td>
</tr>
<tr>
<td>TWEEDEIE</td>
<td>Tweedie**</td>
<td>(p &gt; 1, \mu &gt; 0,\phi &gt; 0)</td>
<td>(f(x) = a(x, \phi) \exp\left[\frac{1}{\phi}\left(\frac{x\mu^{1-p}}{1-p} - \kappa(\mu, p)\right)\right]) (F(x) = \int_0^x f(t)dt)</td>
</tr>
<tr>
<td>STWEEDEIE</td>
<td>Scaled Tweedie**</td>
<td>(\theta &gt; 0, \lambda &gt; 0, 1 &lt; p &lt; 2)</td>
<td>(f(x) = a(x, \theta, \lambda, p) \exp\left(-\frac{x}{\theta} - \lambda\right)) (F(x) = \int_0^x f(t)dt)</td>
</tr>
<tr>
<td>WEIBULL</td>
<td>Weibull</td>
<td>(\theta &gt; 0, \tau &gt; 0)</td>
<td>(f(x) = \frac{1}{\tau^\tau \gamma^\gamma} e^{-\tau x}) (F(x) = 1 - e^{-\tau x})</td>
</tr>
</tbody>
</table>

**For more information, see the section “Tweedie Distributions” on page 243.**

**Notes:**
1. \(z = x/\theta\), wherever \(z\) is used.
2. \(\theta\) denotes the scale parameter for all the distributions. For LOGN, \(\log(\theta) = \mu\).
3. Parameters are listed in the order in which they are defined in the distribution model.
4. \(\gamma(a, b) = \int_0^b t^{a-1} e^{-t} dt\) is the lower incomplete gamma function.
5. \(\Phi(y) = \frac{1}{2} \left(1 + \text{erf}\left(\frac{y}{\sqrt{2}}\right)\right)\) is the standard normal CDF.
Tweedie Distributions

Tweedie distributions are a special case of the exponential dispersion family (Jørgensen 1987) with a property that the variance of the distribution is equal to $\phi \mu^p$, where $\mu$ is the mean of the distribution, $\phi$ is a dispersion parameter, and $p$ is an index parameter as discovered by Tweedie (1984). The distribution is defined for all values of $p$ except for values of $p$ in the open interval $(0, 1)$. Many important known distributions are a special case of Tweedie distributions including normal ($p=0$), Poisson ($p=1$), gamma ($p=2$), and the inverse Gaussian ($p=3$). Apart from these special cases, the probability density function (PDF) of the Tweedie distribution does not have an analytic expression. For $p > 1$, it has the form (Dunn and Smyth 2005),

$$f(x; \mu, \phi, p) = a(x, \phi) \exp \left[ \frac{1}{\phi} \left( \frac{x^{1-p}}{1-p} - \kappa(\mu, p) \right) \right]$$

where $\kappa(\mu, p) = \mu^{2-p} / (2 - p)$ for $p \neq 2$ and $\kappa(\mu, p) = \log(\mu)$ for $p = 2$. The function $a(x, \phi)$ does not have an analytical expression. It is typically evaluated using series expansion methods described in Dunn and Smyth (2005).

For $1 < p < 2$, the Tweedie distribution is a compound Poisson-gamma mixture distribution, which is the distribution of $S$ defined as

$$S = \sum_{i=1}^{N} X_i$$

where $N \sim \text{Poisson}(\lambda)$ and $X_i \sim \text{gamma}(\alpha, \theta)$ are independent and identically distributed gamma random variables with shape parameter $\alpha$ and scale parameter $\theta$. At $X = 0$, the density is a probability mass that is governed by the Poisson distribution, and for values of $X > 0$, it is a mixture of gamma variates with Poisson mixing probability. The parameters $\lambda$, $\alpha$, and $\theta$ are related to the natural parameters $\mu$, $\phi$, and $p$ of the Tweedie distribution as

$$\lambda = \frac{\mu^{2-p}}{\phi(2 - p)}$$
$$\alpha = \frac{2 - p}{p - 1}$$
$$\theta = \phi(p - 1)\mu^{p-1}$$

The mean of a Tweedie distribution is positive for $p > 1$.

Two predefined versions of the Tweedie distribution are provided with the SEVSELECT procedure. The first version, named TWEEDIE and defined for $p > 1$, has the natural parameterization with parameters $\mu$, $\phi$, and $p$. The second version, named STWEEDIE and defined for $1 < p < 2$, is the version with a scale parameter. It corresponds to the compound Poisson-gamma distribution with gamma scale parameter $\theta$, Poisson mean parameter $\lambda$, and the index parameter $p$. The index parameter decides the shape parameter $\alpha$ of the gamma distribution as

$$\alpha = \frac{2 - p}{p - 1}$$

The parameters $\theta$ and $\lambda$ of the STWEEDIE distribution are related to the parameters $\mu$ and $\phi$ of the TWEEDIE distribution as

$$\mu = \lambda \theta \alpha$$
$$\phi = \frac{(\lambda \theta \alpha)^{2-p}}{\lambda(2 - p)} = \frac{\theta}{(p - 1)(\lambda \theta \alpha)^{p-1}}$$
You can fit either version when there are no regression variables. Each version has its own merits. If you fit the TWEEDIE version, you have the direct estimate of the overall mean of the distribution. If you are interested in the most practical range of the index parameter $1 < p < 2$, then you can fit the STWEEDIE version, which provides you direct estimates of the Poisson and gamma components that comprise the distribution (an estimate of the gamma shape parameter $\alpha$ is easily obtained from the estimate of $p$).

If you want to estimate the effect of exogenous (regression) variables on the distribution, then you must use the STWEEDIE version, because PROC SEVSELECT requires a distribution to have a scale parameter in order to estimate regression effects. For more information, see the section “Estimating Regression Effects” on page 256. The gamma scale parameter $\theta$ is the scale parameter of the STWEEDIE distribution. If you are interested in determining the effect of regression variables on the mean of the distribution, you can do so by first fitting the STWEEDIE distribution to determine the effect of the regression variables on the scale parameter $\theta$. Then, you can easily estimate how the mean of the distribution is affected by the regression variables using the relationship $\mu = c \theta$, where $c = \lambda \alpha = \lambda(2 - p)/(p - 1)$. The estimates of the regression parameters remain the same, whereas the estimate of the intercept parameter is adjusted by the estimates of the $\lambda$ and $p$ parameters.

Parameter Initialization for Predefined Distributions

The parameters are initialized by using the method of moments for all the distributions, except for the gamma and the Weibull distributions. For the gamma distribution, approximate maximum likelihood estimates are used. For the Weibull distribution, the method of percentile matching is used.

Given $n$ observations of the severity value $y_i$ ($1 \leq i \leq n$), the estimate of $k$th raw moment is denoted by $m_k$ and computed as

$$m_k = \frac{1}{n} \sum_{i=1}^{n} y_i^k$$

The 100$p$th percentile is denoted by $\pi_p$ ($0 \leq p \leq 1$). By definition, $\pi_p$ satisfies

$$F(\pi_p^-) \leq p \leq F(\pi_p^+)$$

where $F(\pi_p^-) = \lim_{h \downarrow 0} F(\pi_p - h)$. PROC SEVSELECT uses the following practical method of computing $\pi_p$. Let $\hat{F}_n(y)$ denote the empirical distribution function (EDF) estimate at a severity value $y$. Let $y_p^−$ and $y_p^+$ denote two consecutive values in the ascending sequence of $y$ values such that $\hat{F}_n(y_p^−) < p$ and $\hat{F}_n(y_p^+) \geq p$. Then, the estimate $\hat{\pi}_p$ is computed as

$$\hat{\pi}_p = y_p^− + \frac{p - \hat{F}_n(y_p^+)}{\hat{F}_n(y_p^+) - \hat{F}_n(y_p^−)} (y_p^+ - y_p^−)$$

Let $\epsilon$ denote the smallest double-precision floating-point number such that $1 + \epsilon > 1$. This machine precision constant can be obtained by using the CONSTANT function in Base SAS software.

The details of how parameters are initialized for each predefined distribution are as follows:

BURR  
Burr proposed 12 types of families of continuous distributions (Burr 1942; Rodriguez 2005). The predefined BURR distribution in PROC SEVSELECT implements Burr’s type XII
distribution. The parameters are initialized by using the method of moments. The $k$th raw moment of the Burr distribution of type XII is

$$E[X^k] = \frac{\theta^k \Gamma(1 + k/\gamma) \Gamma(\alpha - k/\gamma)}{\Gamma(\alpha)}, \quad -\gamma < k < \alpha \gamma$$

Three moment equations $E[X^k] = m_k$ ($k = 1, 2, 3$) need to be solved for initializing the three parameters of the distribution. In order to get an approximate closed form solution, the second shape parameter $\hat{\gamma}$ is initialized to a value of 2. If $2m_3 - 3m_1 m_2 > 0$, then simplifying and solving the moment equations yields the following feasible set of initial values:

$$\hat{\theta} = \sqrt{\frac{m_2 m_3}{2m_3 - 3m_1 m_2}}, \quad \hat{\alpha} = 1 + \frac{m_3}{2m_3 - 3m_1 m_2}, \quad \hat{\gamma} = 2$$

If $2m_3 - 3m_1 m_2 < \epsilon$, then the parameters are initialized as follows:

$$\hat{\theta} = \sqrt{m_2}, \quad \hat{\alpha} = 2, \quad \hat{\gamma} = 2$$

**EXP**

The parameters are initialized by using the method of moments. The $k$th raw moment of the exponential distribution is

$$E[X^k] = \theta^k \Gamma(k + 1), \quad k > -1$$

Solving $E[X] = m_1$ yields the initial value of $\hat{\theta} = m_1$.

**GAMMA**

The parameter $\alpha$ is initialized by using its approximate maximum likelihood (ML) estimate. For a set of $n$ independent and identically distributed observations $y_i$ ($1 \leq i \leq n$) drawn from a gamma distribution, the log likelihood $l$ is defined as follows:

$$l = \sum_{i=1}^{n} \log \left( y_i^{\alpha-1} e^{-y_i/\theta} / \theta^\alpha \Gamma(\alpha) \right)$$

$$= (\alpha - 1) \sum_{i=1}^{n} \log(y_i) - \frac{1}{\theta} \sum_{i=1}^{n} y_i - n\alpha \log(\theta) - n \log(\Gamma(\alpha))$$

Using a shorter notation of $\sum$ to denote $\sum_{i=1}^{n}$ and solving the equation $\partial l / \partial \theta = 0$ yields the following ML estimate of $\theta$:

$$\hat{\theta} = \frac{\sum_{i=1}^{n} y_i}{n\alpha} = \frac{m_1}{\alpha}$$

Substituting this estimate in the expression of $l$ and simplifying gives

$$l = (\alpha - 1) \sum \log(y_i) - n\alpha - n\alpha \log(m_1) + n\alpha \log(\alpha) - n \log(\Gamma(\alpha))$$

Let $d$ be defined as follows:

$$d = \log(m_1) - \frac{1}{n} \sum \log(y_i)$$

Solving the equation $\partial l / \partial \alpha = 0$ yields the following expression in terms of the digamma function, $\psi(\alpha)$:

$$\log(\alpha) - \psi(\alpha) = d$$
The digamma function can be approximated as follows:

\[ \psi(\alpha) \approx \log(\alpha) - \frac{1}{\alpha} \left( 0.5 + \frac{1}{12\alpha + 2} \right) \]

This approximation is within 1.4% of the true value for all the values of \( \alpha > 0 \) except when \( \alpha \) is arbitrarily close to the positive root of the digamma function (which is approximately 1.461632). Even for the values of \( \alpha \) that are close to the positive root, the absolute error between true and approximate values is still acceptable (\( |\psi(\alpha) - \psi(\alpha)| < 0.005 \) for \( \alpha > 1.07 \)). Solving the equation that arises from this approximation yields the following estimate of \( \alpha \):

\[ \hat{\alpha} = \frac{3 - d + \sqrt{(d - 3)^2 + 24d}}{12d} \]

If this approximate ML estimate is infeasible, then the method of moments is used. The kth raw moment of the gamma distribution is

\[ E[X^k] = \theta^k \frac{\Gamma(\alpha + k)}{\Gamma(\alpha)}, \quad k > -\alpha \]

Solving \( E[X] = m_1 \) and \( E[X^2] = m_2 \) yields the following initial value for \( \alpha \):

\[ \hat{\alpha} = \frac{m_1^2}{m_2 - m_1^2} \]

If \( m_2 - m_1^2 < \epsilon \) (almost zero sample variance), then \( \alpha \) is initialized as follows:

\[ \hat{\alpha} = 1 \]

After computing the estimate of \( \alpha \), the estimate of \( \theta \) is computed as follows:

\[ \hat{\theta} = \frac{m_1}{\hat{\alpha}} \]

Both the maximum likelihood method and the method of moments arrive at the same relationship between \( \hat{\alpha} \) and \( \hat{\theta} \).

GPD

The parameters are initialized by using the method of moments. Notice that for \( \xi > 0 \), the CDF of the generalized Pareto distribution (GPD) is:

\[ F(x) = 1 - \left( 1 + \frac{\xi x}{\theta} \right)^{-1/\xi} = 1 - \left( \frac{\theta / \xi}{x + \theta / \xi} \right)^{1/\xi} \]

This is equivalent to a Pareto distribution with scale parameter \( \theta_1 = \theta / \xi \) and shape parameter \( \alpha = 1/\xi \). Using this relationship, the parameter initialization method used for the PARETO distribution is used to get the following initial values for the parameters of the GPD distribution:

\[ \hat{\theta} = \frac{m_1 m_2}{2(m_2 - m_1^2)}, \quad \hat{\xi} = \frac{m_2 - 2m_1^2}{2(m_2 - m_1^2)} \]

If \( m_2 - m_1^2 < \epsilon \) (almost zero sample variance) or \( m_2 - 2m_1^2 < \epsilon \), then the parameters are initialized as follows:

\[ \hat{\theta} = \frac{m_1}{2}, \quad \hat{\xi} = \frac{1}{2} \]
IGAUSS

The parameters are initialized by using the method of moments. The standard parameterization of the inverse Gaussian distribution (also known as the Wald distribution), in terms of the location parameter \( \mu \) and shape parameter \( \lambda \), is as follows (Klugman, Panjer, and Willmot 1998, p. 583):

\[
\begin{align*}
  f(x) &= \sqrt{\frac{\lambda}{2\pi x^3}} \exp\left(-\frac{\lambda(x - \mu)^2}{2\mu^2 x}\right) \\
  F(x) &= \Phi\left(\frac{x - \mu}{\lambda} \sqrt{\frac{\lambda}{x}}\right) + \Phi\left(\frac{x}{\mu} + 1 \sqrt{\frac{\lambda}{x}}\right) \exp\left(\frac{2\lambda}{\mu}\right)
\end{align*}
\]

For this parameterization, it is known that the mean is \( E[X] = \mu \) and the variance is \( \text{Var}[X] = \mu^3/\lambda \), which yields the second raw moment as \( E[X^2] = \mu^2(1 + \mu/\lambda) \) (computed by using \( E[X^2] = \text{Var}[X] + (E[X])^2 \)).

The predefined IGAUSS distribution in PROC SEVSELECT uses the following alternate parameterization to allow the distribution to have a scale parameter, \( \theta \):

\[
\begin{align*}
  f(x) &= \sqrt{\frac{\alpha \theta}{2\pi x^3}} \exp\left(-\frac{\alpha(x - \theta)^2}{2x\theta}\right) \\
  F(x) &= \Phi\left(\frac{x - \theta}{\alpha \theta} \sqrt{\frac{\alpha \theta}{x}}\right) + \Phi\left(\frac{x}{\theta} + 1 \sqrt{\frac{\alpha \theta}{x}}\right) \exp(2\alpha)
\end{align*}
\]

The parameters \( \theta \) (scale) and \( \alpha \) (shape) of this alternate form are related to the parameters \( \mu \) and \( \lambda \) of the preceding form such that \( \theta = \mu \) and \( \alpha = \lambda/\mu \). Using this relationship, the first and second raw moments of the IGAUSS distribution are

\[
\begin{align*}
  E[X] &= \theta \\
  E[X^2] &= \theta^2 \left(1 + \frac{1}{\alpha}\right)
\end{align*}
\]

Solving \( E[X] = m_1 \) and \( E[X^2] = m_2 \) yields the following initial values:

\[
\hat{\theta} = m_1, \quad \hat{\alpha} = \frac{m_1^2}{m_2 - m_1^2}
\]

If \( m_2 - m_1^2 < \epsilon \) (almost zero sample variance), then the parameters are initialized as follows:

\[
\hat{\theta} = m_1, \quad \hat{\alpha} = 1
\]

LOGN

The parameters are initialized by using the method of moments. The \( k \)th raw moment of the lognormal distribution is

\[
E[X^k] = \exp\left(k\mu + \frac{k^2\sigma^2}{2}\right)
\]

Solving \( E[X] = m_1 \) and \( E[X^2] = m_2 \) yields the following initial values:

\[
\hat{\mu} = 2 \log(m_1) - \frac{\log(m_2)}{2}, \quad \hat{\sigma} = \sqrt{\log(m_2) - 2\log(m_1)}
\]
Chapter 6: The SEVSELECT Procedure

PARETO

The parameters are initialized by using the method of moments. The $k$th raw moment of the Pareto distribution is

$$E[X^k] = \frac{\theta^k \Gamma(k+1) \Gamma(\alpha-k)}{\Gamma(\alpha)}, -1 < k < \alpha$$

Solving $E[X] = m_1$ and $E[X^2] = m_2$ yields the following initial values:

$$\hat{\theta} = \frac{m_1 m_2}{m_2 - 2m_1^2}, \quad \hat{\alpha} = \frac{2(m_2 - m_1^2)}{m_2 - 2m_1^2}$$

If $m_2 - m_1^2 < \epsilon$ (almost zero sample variance) or $m_2 - 2m_1^2 < \epsilon$, then the parameters are initialized as follows:

$$\hat{\theta} = m_1, \quad \hat{\alpha} = 2$$

TWEEDIE

The parameter $p$ is initialized by assuming that the sample is generated from a gamma distribution with shape parameter $\alpha$ and by computing $\hat{p} = \frac{\hat{\alpha} + 2}{\hat{\alpha} + 1}$. The initial value $\hat{\alpha}$ is obtained from using the method previously described for the GAMMA distribution. The parameter $\mu$ is the mean of the distribution. Hence, it is initialized to the sample mean as

$$\hat{\mu} = m_1$$

Variance of a Tweedie distribution is equal to $\phi \mu^p$. Thus, the sample variance is used to initialize the value of $\phi$ as

$$\hat{\phi} = \frac{m_2 - m_1^2}{\hat{\mu} \hat{p}}$$

STWEEDIE

STWEEDIE is a compound Poisson-gamma mixture distribution with mean $\mu = \lambda \theta \alpha$, where $\alpha$ is the shape parameter of the gamma random variables in the mixture and the parameter $p$ is determined solely by $\alpha$. First, the parameter $p$ is initialized by assuming that the sample is generated from a gamma distribution with shape parameter $\alpha$ and by computing $\hat{p} = \frac{\hat{\alpha} + 2}{\hat{\alpha} + 1}$. The initial value $\hat{\alpha}$ is obtained from using the method previously described for the GAMMA distribution. As done for initializing the parameters of the TWEEDIE distribution, the sample mean and variance are used to compute the values $\hat{\mu}$ and $\hat{\phi}$ as

$$\hat{\mu} = m_1$$
$$\hat{\phi} = \frac{m_2 - m_1^2}{\hat{\mu} \hat{p}}$$

Based on the relationship between the parameters of TWEEDIE and STWEEDIE distributions described in the section “Tweedie Distributions” on page 243, values of $\theta$ and $\lambda$ are initialized as

$$\hat{\theta} = \hat{\phi} (\hat{p} - 1) \hat{\mu}^{p-1}$$
$$\hat{\lambda} = \frac{\hat{\mu}}{\hat{\theta} \hat{\alpha}}$$
WEIBULL  The parameters are initialized by using the percentile matching method. Let $q_1$ and $q_3$ denote the estimates of the 25th and 75th percentiles, respectively. Using the formula for the CDF of Weibull distribution, they can be written as

$$1 - \exp(-(q_1/\theta)^r) = 0.25$$
$$1 - \exp(-(q_3/\theta)^r) = 0.75$$

Simplifying and solving these two equations yields the following initial values,

$$\hat{\theta} = \exp \left( \frac{r \log(q_1) - \log(q_3)}{r - 1} \right), \quad \hat{r} = \frac{\log(\log(4))}{\log(q_3) - \log(\hat{\theta})}$$

where $r = \frac{\log(\log(4))}{\log(4/3)}$. These initial values agree with those suggested in Klugman, Panjer, and Willmot (1998).

A summary of the initial values of all the parameters for all the predefined distributions is given in Table 6.5. The table also provides the names of the parameters to use in the INIT= option in the DIST statement if you want to provide a different initial value.
### Table 6.5  Parameter Initialization for Predefined Distributions

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Parameter</th>
<th>Name for INIT Option</th>
<th>Default Initial Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>BURR</td>
<td>$\theta$</td>
<td>theta</td>
<td>$\sqrt{\frac{m_3 m_3}{2m_3 - 3m_1 m_2}}$</td>
</tr>
<tr>
<td></td>
<td>$\alpha$</td>
<td>alpha</td>
<td>$1 + \frac{m_3}{2m_3 - 3m_1 m_2}$</td>
</tr>
<tr>
<td></td>
<td>$\gamma$</td>
<td>gamma</td>
<td>2</td>
</tr>
<tr>
<td>EXP</td>
<td>$\theta$</td>
<td>theta</td>
<td>$m_1/\alpha$</td>
</tr>
<tr>
<td>GAMMA</td>
<td>$\theta$</td>
<td>theta</td>
<td>$m_3 + \sqrt{(d-3)^2 + 24d}$</td>
</tr>
<tr>
<td></td>
<td>$\alpha$</td>
<td>alpha</td>
<td>$\frac{3m_1 m_2}{2m_3}$</td>
</tr>
<tr>
<td>GPD</td>
<td>$\theta$</td>
<td>theta</td>
<td>$(m_2 - m_1^2)/(2m_2 - m_2^2)$</td>
</tr>
<tr>
<td></td>
<td>$\xi$</td>
<td>xi</td>
<td>$(m_2 - m_1^2)/(2m_2 - m_2^2)$</td>
</tr>
<tr>
<td>IGAUSS</td>
<td>$\theta$</td>
<td>theta</td>
<td>$m_1$</td>
</tr>
<tr>
<td></td>
<td>$\alpha$</td>
<td>alpha</td>
<td>$m_1^2/(m_2 - m_1^2)$</td>
</tr>
<tr>
<td>LOGN</td>
<td>$\mu$</td>
<td>mu</td>
<td>$2 \log(m_1) - \log(m_2)/2$</td>
</tr>
<tr>
<td></td>
<td>$\sigma$</td>
<td>sigma</td>
<td>$\sqrt{\log(m_2) - 2 \log(m_1)}$</td>
</tr>
<tr>
<td>PARETO</td>
<td>$\theta$</td>
<td>theta</td>
<td>$m_1 m_2/(m_2 - 2m_1^2)$</td>
</tr>
<tr>
<td></td>
<td>$\alpha$</td>
<td>alpha</td>
<td>$2(m_2 - m_1^2)/(m_2 - 2m_1^2)$</td>
</tr>
<tr>
<td>TWEEDIE</td>
<td>$\mu$</td>
<td>mu</td>
<td>$m_1$</td>
</tr>
<tr>
<td></td>
<td>$\phi$</td>
<td>phi</td>
<td>$(m_2 - m_1^2)/m_1^p$</td>
</tr>
<tr>
<td></td>
<td>$p$</td>
<td>p</td>
<td>$(\alpha + 2)/ (\alpha + 1)$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>where $\alpha = \frac{3-d+\sqrt{(d-3)^2+24d}}{12d}$</td>
</tr>
<tr>
<td>STWEEDIE</td>
<td>$\theta$</td>
<td>theta</td>
<td>$(m_2 - m_1^2)(p-1)/m_1$</td>
</tr>
<tr>
<td></td>
<td>$\lambda$</td>
<td>lambda</td>
<td>$m_1^2/ (\alpha(2m_2 - m_1^2)(p-1))$</td>
</tr>
<tr>
<td></td>
<td>$p$</td>
<td>p</td>
<td>$(\alpha + 2)/ (\alpha + 1)$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>where $\alpha = \frac{3-d+\sqrt{(d-3)^2+24d}}{12d}$</td>
</tr>
<tr>
<td>WEIBULL</td>
<td>$\theta$</td>
<td>theta</td>
<td>$\exp\left(\frac{r \log(q_1) - \log(q_3)}{r - 1}\right)$</td>
</tr>
<tr>
<td></td>
<td>$\tau$</td>
<td>tau</td>
<td>$\log(\log(4))/\log(q_3 - \log(\hat{\theta}))$</td>
</tr>
</tbody>
</table>

Notes:
1. $m_k$ denotes the $k$th raw moment.
2. $d = \log(m_1) - \sum \log(y_i)/n$
3. $q_1$ and $q_3$ denote the 25th and 75th percentiles, respectively.
4. $r = \log(\log(4))/\log(\log(4/3))$
Censoring and Truncation

One of the key features of PROC SEVSELECT is that it enables you to specify whether the severity event’s magnitude is observable and if it is observable, then whether the exact value of the magnitude is known. If an event is unobservable when the magnitude is in certain intervals, then it is referred to as a truncation effect. If the exact magnitude of the event is not known, but it is known to have a value in a certain interval, then it is referred to as a censoring effect.

PROC SEVSELECT allows a severity event to be subject to any combination of the following four censoring and truncation effects:

- **Left-truncation**: An event is said to be left-truncated if it is observed only when \( Y > T_l \), where \( Y \) denotes the random variable for the magnitude and \( T_l \) denotes a random variable for the truncation threshold. You can specify left-truncation using the `LEFTTRUNCATED=` option in the LOSS statement.

- **Right-truncation**: An event is said to be right-truncated if it is observed only when \( Y \leq T_r \), where \( Y \) denotes the random variable for the magnitude and \( T_r \) denotes a random variable for the truncation threshold. You can specify right-truncation using the `RIGHTTRUNCATED=` option in the LOSS statement.

- **Left-censoring**: An event is said to be left-censored if it is known that the magnitude is \( Y \leq C_l \), but the exact value of \( Y \) is not known. \( C_l \) is a random variable for the censoring limit. You can specify left-censoring using the `LEFTCENSORED=` option in the LOSS statement.

- **Right-censoring**: An event is said to be right-censored if it is known that the magnitude is \( Y > C_r \), but the exact value of \( Y \) is not known. \( C_r \) is a random variable for the censoring limit. You can specify right-censoring using the `RIGHTCENSORED=` option in the LOSS statement.

For each effect, you can specify a different threshold or limit for each observation or specify a single threshold or limit that applies to all the observations.

If all four types of effects are present on an event, then the following relationship holds: \( T_l < C_r \leq C_l \leq T_r \). PROC SEVSELECT checks these relationships and writes a warning to the SAS log if any relationship is violated.

If you specify the response variable in the LOSS statement, then PROC SEVSELECT also checks whether each observation satisfies the definitions of the specified censoring and truncation effects. If you specify left-truncation, then PROC SEVSELECT ignores observations where \( Y \leq T_l \), because such observations are not observable by definition. Similarly, if you specify right-truncation, then PROC SEVSELECT ignores observations where \( Y > T_r \). If you specify left-censoring, then PROC SEVSELECT treats an observation with \( Y > C_l \) as uncensored and ignores the value of \( C_l \). The observations with \( Y \leq C_l \) are considered as left-censored, and the value of \( Y \) is ignored. If you specify right-censoring, then PROC SEVSELECT treats an observation with \( Y \leq C_r \) as uncensored and ignores the value of \( C_r \). The observations with \( Y > C_r \) are considered as right-censored, and the value of \( Y \) is ignored. If you specify both left-censoring and right-censoring, it is referred to as interval-censoring. If \( C_r < C_l \) is satisfied for an observation, then it is considered as interval-censored and the value of the response variable is ignored. If \( C_r = C_l \) for an observation, then PROC SEVSELECT assumes that observation to be uncensored. If all the observations in a data table are censored in some form, then the specification of the response variable in the LOSS statement is
optional, because the actual value of the response variable is not required for the purposes of estimating a model.

Specification of censoring and truncation affects the likelihood of the data (see the section “Likelihood Function” on page 253) and how the empirical distribution function (EDF) is estimated (see the section “Empirical Distribution Function Estimation Methods” on page 261).

**Probability of Observability**

For left-truncated data, PROC SEVSELECT also enables you to provide additional information in the form of *probability of observability* by using the `PROBOBSERVED=` option. It is defined as the probability that the underlying severity event gets observed (and recorded) for the specified left-truncation threshold value. For example, if you specify a value of 0.75, then for every 75 observations recorded above a specified threshold, 25 more events have happened with a severity value less than or equal to the specified threshold. Although the exact severity value of those 25 events is not known, PROC SEVSELECT can use the information about the number of those events.

In particular, for each left-truncated observation, PROC SEVSELECT assumes a presence of \((1 - p)/p\) additional observations with \(y_i = t_i\). These additional observations are then used for computing the likelihood (see the section “Probability of Observability and Likelihood” on page 254) and an unconditional estimate of the empirical distribution function (see the section “EDF Estimates and Truncation” on page 266).

**Truncation and Conditional CDF Estimates**

If you specify left-truncation without the probability of observability or if you specify right-truncation, then the EDF estimates that are computed by all methods except the STANDARD method are conditional on the truncation information. For more information, see the section “EDF Estimates and Truncation” on page 266. In such cases, PROC SEVSELECT uses conditional estimates of the CDF for computational or visual comparison to the EDF estimates.

Let \(t_{i \text{min}}^l = \min_i \{t_i^l\}\) be the smallest value of the left-truncation threshold \(t_i^l\) is the left-truncation threshold for observation \(i\) and \(t_{i \text{max}}^r = \max_i \{t_i^r\}\) be the largest value of the right-truncation threshold \(t_i^r\) is the right-truncation threshold for observation \(i\). If \(\hat{F}(y)\) denotes the unconditional estimate of the CDF at \(y\), then the conditional estimate \(\hat{F}_c(y)\) is computed as follows:

- If you do not specify the probability of observability, then the EDF estimates are conditional on the left-truncation information. If an observation is both left-truncated and right-truncated, then

\[
\hat{F}_c(y) = \frac{\hat{F}(y) - \hat{F}(t_{\text{min}}^l)}{\hat{F}(t_{\text{max}}^r) - \hat{F}(t_{\text{min}}^l)}
\]

If an observation is left-truncated but not right-truncated, then

\[
\hat{F}_c(y) = \frac{\hat{F}(y) - \hat{F}(t_{\text{min}}^l)}{1 - \hat{F}(t_{\text{min}}^l)}
\]

If an observation is right-truncated but not left-truncated, then

\[
\hat{F}_c(y) = \frac{\hat{F}(y)}{\hat{F}(t_{\text{max}}^r)}
\]
Parameter Estimation Method

If you specify the probability of observability, then EDF estimates are not conditional on the left-truncation information. If an observation is not right-truncated, then the conditional estimate is the same as the unconditional estimate. If an observation is right-truncated, then the conditional estimate is computed as

$$\hat{F}^c(y) = \frac{\hat{F}(y)}{\hat{F}(t^c_{\text{min}})}$$

If you specify regression effects, then $\hat{F}(y)$, $\hat{F}(t^l_{\text{min}})$, and $\hat{F}(t^c_{\text{max}})$ are all computed from a mixture distribution, as described in the section “CDF and PDF Estimates with Regression Effects” on page 259.

Parameter Estimation Method

PROC SEVSELECT uses the maximum likelihood (ML) method to estimate the parameters of each model. A nonlinear optimization process is used to maximize the log of the likelihood function.

Likelihood Function

Let $f_{\Theta}(x)$ and $F_{\Theta}(x)$ denote the PDF and CDF, respectively, evaluated at $x$ for a set of parameter values $\Theta$. Let $Y$ denote the random response variable, and let $y$ denote its value recorded in an observation in the input data table. Let $T^l$ and $T^r$ denote the random variables for the left-truncation and right-truncation threshold, respectively, and let $t^l$ and $t^r$ denote their values for an observation, respectively. If there is no left-truncation, then $t^l = t^l_{\text{min}}$, where $t^l$ is the smallest value in the support of the distribution; so $F(t^l) = 0$. If there is no right-truncation, then $t^r = t^r_{\text{max}}$, where $t^r$ is the largest value in the support of the distribution; so $F(t^r) = 1$. Let $C^l$ and $C^r$ denote the random variables for the left-censoring and right-censoring limit, respectively, and let $c^l$ and $c^r$ denote their values for an observation, respectively. If there is no left-censoring, then $c^l = c^l_{\text{min}}$; so $F(c^l) = 1$. If there is no right-censoring, then $c^r = c^r_{\text{max}}$; so $F(c^r) = 0$.

The set of input observations can be categorized into the following four subsets within each BY group:

- **$E$** is the set of uncensored and untruncated observations. The likelihood of an observation in $E$ is

$$l_E = \Pr(Y = y) = f_{\Theta}(y)$$

- **$E_t$** is the set of uncensored observations that are truncated. The likelihood of an observation in $E_t$ is

$$l_{E_t} = \Pr(Y = y|t^l < Y \leq t^r) = \frac{f_{\Theta}(y)}{F_{\Theta}(t^r) - F_{\Theta}(t^l)}$$

- **$C$** is the set of censored observations that are not truncated. The likelihood of an observation $C$ is

$$l_C = \Pr(c^r < Y \leq c^l) = F_{\Theta}(c^l) - F_{\Theta}(c^r)$$

- **$C_t$** is the set of censored observations that are truncated. The likelihood of an observation $C_t$ is

$$l_{C_t} = \Pr(c^r < Y \leq c^l|t^l < Y \leq t^r) = \frac{F_{\Theta}(c^l) - F_{\Theta}(c^r)}{F_{\Theta}(t^r) - F_{\Theta}(t^l)}$$
Note that \((E \cup E_t) \cap (C \cup C_t) = \emptyset\). Also, the sets \(E_t\) and \(C_t\) are empty when you do not specify truncation, and the sets \(C\) and \(C_t\) are empty when you do not specify censoring.

Given this, the likelihood of the data \(L\) is as follows:

\[
L = \prod_{E} f_{\Theta}(y) \prod_{E_t, t < t^l} f_{\Theta}(y) / F_{\Theta}(t^l) \prod_{C} F_{\Theta}(c^l) - F_{\Theta}(c^r) \prod_{C_t, t > t^l} F_{\Theta}(c^l) - F_{\Theta}(c^r) / F_{\Theta}(t^l)
\]

The maximum likelihood procedure used by PROC SEVSELECT finds an optimal set of parameter values \(\hat{\Theta}\) that maximizes \(\log(L)\) subject to the boundary constraints on parameter values. For a distribution \(dist\), you can specify such boundary constraints by using the \(dist_{\text{LOWERBOUNDS}}\) and \(dist_{\text{UPPERBOUNDS}}\) subroutines. For more information, see the section “Defining a Severity Distribution Model with the FCMP Procedure” on page 272. Some aspects of the optimization process can be controlled by using the \(\text{NLOPTIONS}\) statement.

**Probability of Observability and Likelihood**

If you specify the probability of observability for the left-truncation, then PROC SEVSELECT uses a modified likelihood function for each truncated observation. If the probability of observability is \(p \in (0.0, 1.0]\), then for each left-truncated observation with truncation threshold \(t^l\), there exist \((1-p)/p\) observations with a response variable value less than or equal to \(t^l\). Each such observation has a probability of \(\Pr(Y \leq t^l) = F_{\Theta}(t^l)\). The right-truncation and censoring information does not apply to these added observations. Thus, following the notation of the section “Likelihood Function” on page 253, the likelihood of the data is as follows:

\[
L = \prod_{E} f_{\Theta}(y) \prod_{E_t, t < t^l} f_{\Theta}(y) / F_{\Theta}(t^l) \prod_{C} F_{\Theta}(c^l) - F_{\Theta}(c^r) \prod_{C_t, t > t^l} F_{\Theta}(c^l) - F_{\Theta}(c^r) / F_{\Theta}(t^l) \left(1-p\right) / p
\]

Note that the likelihood of the observations that are not left-truncated (observations in sets \(E\) and \(C\), and observations in sets \(E_t\) and \(C_t\) for which \(t^l = t^l\)) is not affected.

If you specify a custom objective function, then PROC SEVSELECT accounts for the probability of observability only while computing the empirical distribution function estimate. The parameter estimates are affected only by your custom objective function.

**Estimating Covariance and Standard Errors**

PROC SEVSELECT computes an estimate of the covariance matrix of the parameters by using the asymptotic theory of the maximum likelihood estimators (MLE). If \(N\) denotes the number of observations used for estimating a parameter vector \(\Theta\), then the theory states that as \(N \to \infty\), the distribution of \(\hat{\Theta}\), the estimate of \(\Theta\), converges to a normal distribution with mean \(\Theta\) and covariance \(\hat{C}\) such that \(I(\Theta) \cdot \hat{C} \to 1\), where \(I(\Theta) = -E \left[\nabla^2 \log(L(\Theta))\right]\) is the information matrix for the likelihood of the data, \(L(\Theta)\). The covariance estimate is obtained by using the inverse of the information matrix.
In particular, if $G = \nabla^2 (-\log(L(\theta)))$ denotes the Hessian matrix of the negative of log likelihood, then the covariance estimate is computed as

$$\hat{C} = \frac{N}{d} G^{-1}$$

where $d$ is a denominator that is determined by the VARDEF= option. If VARDEF=N, then $d = N$, which yields the asymptotic covariance estimate. If VARDEF=DF, then $d = N - k$, where $k$ is number of parameters (the model’s degrees of freedom). The VARDEF=DF option is the default, because it attempts to correct the potential bias introduced by the finite sample.

The standard error $s_i$ of the parameter $\theta_i$ is computed as the square root of the $i$th diagonal element of the estimated covariance matrix; that is, $s_i = \sqrt{\hat{C}_{ii}}$.

If you specify a custom objective function, then the covariance matrix of the parameters is still computed by inverting the information matrix, except that the Hessian matrix $G$ is computed as $G = \nabla^2 \log(U(\theta))$, where $U$ denotes your custom objective function that is minimized by the optimizer.

Covariance and standard error estimates might not be available if the Hessian matrix is found to be singular at the end of the optimization process. This can especially happen if the optimization process stops without converging.

---

**Parameter Initialization**

PROC SEVSELECT enables you to initialize parameters of a model in different ways. A model can have two kinds of parameters: distribution parameters and regression parameters.

The distribution parameters can be initialized by using one of the following three methods:

- **INIT= option** You can use the INIT= option in the DIST statement.
- **INEST= option** You can use the INEST= data table in the PROC SEVSELECT statement.
- **PARMINIT subroutine** You can define a dist_PARMINIT subroutine in the distribution model. For more information, see the section “Defining a Severity Distribution Model with the FCMP Procedure” on page 272.

Note that only one of the initialization methods is used. You cannot combine them. They are used in the following order:

- The method that uses the INIT= option takes the highest precedence. If you use the INIT= option to provide an initial value for at least one parameter, then other initialization methods (INEST= or PARMINIT) are not used. If you specify initial values for some but not all the parameters by using the INIT= option, then the uninitialized parameters are initialized to the default value of 0.001.
  
  If you use this option and if you specify the regression effects, then the value of the first distribution parameter must be related to the initial value for the base value of the scale or log-transformed scale parameter. For more information, see the section “Estimating Regression Effects” on page 256.

- The method that uses the INEST= data table takes second precedence. If the INEST= data table contains a nonmissing value for even one distribution parameter, then the PARMINIT method is not used and any uninitialized parameters are initialized to the default value of 0.001.
If none of the distribution parameters are initialized by using the INIT= option or the INEST= data table, but the distribution model defines a PARMINIT subroutine, then PROC SEVSELECT invokes that subroutine with appropriate inputs to initialize the parameters. If the PARMINIT subroutine returns missing values for some parameters, then those parameters are initialized to the default value of 0.001.

If none of the initialization methods are used, each distribution parameter is initialized to the default value of 0.001.

For more information about regression models and initialization of regression parameters, see the section “Estimating Regression Effects” on page 256.

PARMINIT-Based Parameter Initialization Method and Distributed Data

If the input data are distributed across the computational nodes and if you use the PARMINIT subroutine to initialize the distribution parameters, then PROC SEVSELECT invokes that subroutine on each computational node with the data that are local to that node. The EDF estimates that are supplied to the PARMINIT subroutine are also computed using the local data. The initial values of the parameters that are supplied to the optimizer are the average of the local estimates that are computed on each node. This approach works well if the data are distributed randomly across nodes.

Estimating Regression Effects

The SEVSELECT procedure enables you to estimate the influence of regression (exogenous) effects while fitting a distribution if the distribution has a scale parameter or a log-transformed scale parameter.

Let \( x_j, j = 1, \ldots, k \), denote the \( k \) regression effects. Let \( \beta_j \) denote the regression parameter that corresponds to the effect \( x_j \). If you do not specify regression effects, then the model for the response variable \( Y \) is of the form

\[
Y \sim F(\Theta)
\]

where \( F \) is the distribution of \( Y \) with parameters \( \Theta \). This model is usually referred to as the error model. The regression effects are modeled by extending the error model to the following form:

\[
Y \sim \exp(\sum_{j=1}^{k} \beta_j x_j) \cdot F(\Theta)
\]

Under this model, the distribution of \( Y \) is valid and belongs to the same parametric family as \( F \) if and only if \( F \) has a scale parameter. Let \( \theta \) denote the scale parameter and \( \Omega \) denote the set of nonscale distribution parameters of \( F \). Then the model can be rewritten as

\[
Y \sim F(\theta, \Omega)
\]

such that \( \theta \) is modeled by the regression effects as

\[
\theta = \theta_0 \cdot \exp(\sum_{j=1}^{k} \beta_j x_j)
\]
where $\theta_0$ is the base value of the scale parameter. Thus, the scale regression model consists of the following parameters: $\theta_0$, $\Omega$, and $\beta_j (j = 1, \ldots, k)$.

Given this form of the model, distributions without a scale parameter cannot be considered when regression effects are to be modeled. If a distribution does not have a direct scale parameter, then PROC SEVSELECT accepts it only if it has a log-transformed scale parameter—that is, if it has a parameter $p = \log(\theta)$.

**Offset Variable**

You can specify that an offset variable be included in the scale regression model by specifying it in the OFFSET= option of the SCALEMODEL statement. The offset variable is a regressor whose regression coefficient is known to be 1. If $x_o$ denotes the offset variable, then the scale regression model becomes

$$\theta = \theta_0 \cdot \exp(x_o + \sum_{j=1}^{k} \beta_j x_j)$$

The regression coefficient of the offset variable is fixed at 1 and not estimated, so it is not reported in the ParameterEstimates ODS table. However, if you specify the OUTEST= data table, then the regression coefficient is added as a variable to that data table. The value of the offset variable in OUTEST= data table is equal to 1 for the estimates row (\_TYPE_='EST') and is equal to the special missing value .F for the standard error (\_TYPE_='STDERR') and covariance (\_TYPE_='COV') rows.

An offset variable is useful to model the scale parameter per unit of some measure of exposure. For example, in the automobile insurance context, measure of exposure can be the number of car-years insured or the total number of miles driven by a fleet of cars at a rental car company. For worker’s compensation insurance, if you want to model the expected loss per enterprise, then you can use the number of employees or total employee salary as the measure of exposure. For epidemiological data, measure of exposure can be the number of people who are exposed to a certain pathogen when you are modeling the loss associated with an epidemic. In general, if $e$ denotes the value of the exposure measure and if you specify $x_o = \log(e)$ as the offset variable, then you are modeling the influence of other regression effects ($x_j$) on the size of the scale of the distribution per unit of exposure.

Another use for an offset variable is when you have a priori knowledge of the influence of some exogenous variables that cannot be included in the SCALEMODEL statement. You can model the combined influence of such variables as an offset variable in order to correct for the omitted variable bias.

**Parameter Initialization for Regression Models**

The regression parameters are initialized either by using the values that you specify or by the default method.

- If you provide initial values for the regression parameters, then you must provide valid, nonmissing initial values for $\theta_0$ and $\beta_j$ parameters for all $j$.

  You can specify the initial value for $\theta_0$ by using either the INEST= data table or the INIT= option in the DIST statement. If the distribution has a direct scale parameter (no transformation), then the initial value for the first parameter of the distribution is used as an initial value for $\theta_0$. If the distribution has a log-transformed scale parameter, then the initial value for the first parameter of the distribution is used as an initial value for $\log(\theta_0)$.

  If you use the INEST= data table to specify the initial values for $\beta_j$, then the INEST= data table must be an unmodified version of a OUTEST= data table that you have created in a previous PROC SEVSELECT step with the same set of regression effects.
If you do not specify valid initial values for $\theta_0$ or $\beta_j$ parameters for all $j$, then PROC SEVSELECT initializes those parameters by using the following method:

Let a random variable $Y$ be distributed as $F(\theta, \Omega)$, where $\theta$ is the scale parameter. By the definition of the scale parameter, a random variable $W = Y / \theta$ is distributed as $G(\Omega)$ such that $G(\Omega) = F(1, \Omega)$. Given a random error term $e$ that is generated from a distribution $G(\Omega)$, a value $y$ from the distribution of $Y$ can be generated as

$$y = \theta \cdot e$$

Taking the logarithm of both sides and using the relationship of $\theta$ with the regression effects yields

$$\log(y) = \log(\theta_0) + \sum_{j=1}^{k} \beta_j x_j + \log(e)$$

PROC SEVSELECT makes use of the preceding relationship to initialize parameters of a regression model with distribution $\text{dist}$ as follows:

1. The following linear regression problem is solved to obtain initial estimates of $\beta_0$ and $\beta_j$:

$$\log(y) = \beta_0 + \sum_{j=1}^{k} \beta_j x_j$$

The estimates of $\beta_j$ ($j = 1, \ldots, k$) in the solution of this regression problem are used to initialize the respective regression parameters of the model. The estimate of $\beta_0$ is later used to initialize the value of $\theta_0$.

The results of this regression are also used to detect whether any regression parameters are linearly dependent on the other regression parameters. If any such parameters are found, then a warning is written to the SAS log and the corresponding parameter is eliminated from further analysis. The estimates for linearly dependent regression parameters are denoted by the special missing value .R in the OUTEST= data table and in any displayed output.

2. Let $s_0$ denote the initial value of the scale parameter.

If the distribution model of $\text{dist}$ does not contain the $\text{dist}$_PARMINIT subroutine, then $s_0$ and all the nonscale distribution parameters are initialized to the default value of 0.001.

However, it is strongly recommended that each distribution’s model contain the $\text{dist}$_PARMINIT subroutine. For more information, see the section “Defining a Severity Distribution Model with the FCMP Procedure” on page 272. If that subroutine is defined, then $s_0$ is initialized as follows:

Each input value $y_i$ of the response variable is transformed to its scale-normalized version $w_i$ as

$$w_i = \frac{y_i}{\exp(\beta_0 + \sum_{j=1}^{k} \beta_j x_{ij})}$$

where $x_{ij}$ denotes the value of $j$th regression effect in the $i$th input observation. These $w_i$ values are used to compute the input arguments for the $\text{dist}$_PARMINIT subroutine. The values that are computed by the subroutine for nonscale parameters are used as their respective initial values.

If the distribution has an untransformed scale parameter, then $s_0$ is set to the value of the scale parameter that is computed by the subroutine. If the distribution has a log-transformed scale parameter $P$, then $s_0$ is computed as $s_0 = \exp(l_0)$, where $l_0$ is the value of $P$ computed by the subroutine.
3. The value of $\theta_0$ is initialized as

$$\theta_0 = s_0 \cdot \exp(\beta_0)$$

**Reporting Estimates of Regression Parameters**

When you request estimates to be written to the output (either ODS displayed output or in the OUTEST= data table), the estimate of the base value of the first distribution parameter is reported. If the first parameter is the log-transformed scale parameter, then the estimate of $\log(\theta_0)$ is reported; otherwise, the estimate of $\theta_0$ is reported. The transform of the first parameter of a distribution dist is controlled by the dist_SCALETRANSFORM function that is defined for it.

**CDF and PDF Estimates with Regression Effects**

When regression effects are estimated, the estimate of the scale parameter depends on the values of the regressors and the estimates of the regression parameters. This dependency results in a potentially different distribution for each observation. To make estimates of the cumulative distribution function (CDF) and probability density function (PDF) comparable across distributions and comparable to the empirical distribution function (EDF), PROC SEVSELECT computes and reports the CDF and PDF estimates from a representative distribution. The representative distribution is a mixture of a certain number of distributions, where each distribution differs only in the value of the scale parameter. You can specify the number of distributions in the mixture and how their scale values are chosen by using the DFMIXTURE= option in the SCALEMODEL statement.

Let $N$ denote the number of observations that are used for estimation, $K$ denote the number of components in the mixture distribution, $s_k$ denote the scale parameter of the $k$th mixture component, and $d_k$ denote the weight associated with $k$th mixture component.

Let $f(y; s_k, \hat{\Omega})$ and $F(y; s_k, \hat{\Omega})$ denote the PDF and CDF, respectively, of the $k$th component distribution, where $\hat{\Omega}$ denotes the set of estimates of all parameters of the distribution other than the scale parameter. Then, the PDF and CDF estimates, $f^*(y)$ and $F^*(y)$, respectively, of the mixture distribution at $y$ are computed as

$$f^*(y) = \frac{1}{D} \sum_{k=1}^{K} d_k f(y; s_k, \hat{\Omega})$$

$$F^*(y) = \frac{1}{D} \sum_{k=1}^{K} d_k F(y; s_k, \hat{\Omega})$$

where $D$ is the normalization factor ($D = \sum_{k=1}^{K} d_k$).

PROC SEVSELECT uses the $F^*(y)$ values to compute the EDF-based statistics of fit.

The scale values $s_k$ for the $K$ mixture components are derived from the set $\{\hat{\lambda}_i\}$ ($i = 1, \ldots, N$) of $N$ linear predictor values, where $\hat{\lambda}_i$ denotes the estimate of the linear predictor due to observation $i$. It is computed as

$$\hat{\lambda}_i = \log(\hat{\theta}_0) + \sum_{j=1}^{k} \hat{\beta}_j x_{ij}$$

where $\hat{\theta}_0$ is an estimate of the base value of the scale parameter, $\hat{\beta}_j$ are the estimates of regression coefficients, and $x_{ij}$ is the value of $j$th regression effect in observation $i$. 
Let $w_i$ denote the weight of observation $i$. If you specify the WEIGHT statement, then the weight is equal to the value of the specified weight variable for the corresponding observation in the DATA= data table; otherwise, the weight is set to 1.

You can specify one of the following method-names in the DFMIXTURE= option in the SCALEMODEL statement to specify the method of choosing $K$ and the corresponding $s_k$ and $d_k$ values:

**FULL**

In this method, there are as many mixture components as the number of observations that are used for estimation. In other words, $K = N$, $s_k = \hat{\theta}_k$, and $d_k = w_k$ ($k = 1, \ldots, N$). This is the slowest method, because it requires $O(N)$ computations to compute the mixture CDF $F^*(y_i)$ or the mixture PDF $f^*(y_i)$ of one observation. For $N$ observations, the computational complexity in terms of number of CDF or PDF evaluations is $O(N^2)$. Even for moderately large values of $N$, the time that is taken to compute the mixture CDF and PDF can significantly exceed the time that is taken to estimate the model parameters. So it is recommended that you use the FULL method only for small data tables.

**MEAN**

In this method, the mixture contains only one distribution, whose scale value is determined by the mean of the linear predictor values that are implied by all the observations. In other words, $s_1$ is computed as

$$s_1 = \exp\left(\frac{1}{N} \sum_{i=1}^{N} \hat{\lambda}_i\right)$$

The component’s weight $d_1$ is set to 1.

This method is the fastest because it requires only one CDF or PDF evaluation per observation. The computational complexity is $O(N)$ for $N$ observations.

If you do not specify the DFMIXTURE= option in the SCALEMODEL statement, then this is the default method.

**QUANTILE**

In this method, a certain number of quantiles are chosen from the set of all linear predictor values. If you specify a value of $q$ for the K= option when specifying this method, then $K = q - 1$ and $s_k$ ($k = 1, \ldots, K$) is computed as $s_k = \exp(\hat{\lambda}_k)$, where $\hat{\lambda}_k$ is the $k$th $q$-quantile from the set $\{\hat{\lambda}_i\}$ ($i = 1, \ldots, N$). The weight of each of the components ($d_k$) is assumed to be 1 for this method.

The default value of $q$ is 2, which implies a one-point mixture that has a distribution whose scale value is equal to the median scale value.

For this method, PROC SEVSELECT needs to sort the $N$ linear predictor values in the set $\{\hat{\lambda}_i\}$; the sorting requires $O(N \log(N))$ computations. Then, computing the mixture estimate of one observation requires $(q - 1)$ CDF or PDF evaluations. Hence, the computational complexity of this method is $O(qN) + O(N \log(N))$ for computing a mixture CDF or PDF of $N$ observations. For $q \ll N$, the QUANTILE method is significantly faster than the FULL method.

**RANDOM**

In this method, a uniform random sample of observations is chosen, and the mixture contains the distributions that are implied by those observations. If you specify a value of $r$ for the K= option when specifying this method, then the size of the sample is $r$. Hence, $K = r$. If $l_j$ denotes the index of $j$th observation in the sample ($j = 1, \ldots, r$), such that $1 \leq l_j \leq N$, then the scale of $k$th component distribution in the mixture is $s_k = \exp(\hat{\lambda}_{l_k})$. The weight of each of the components ($d_k$) is assumed to be 1 for this method.
You can also specify the seed to be used for generating the random sample by using the SEED= option for this method. The same sample of observations is used for all models. Computing a mixture estimate of one observation requires \( r \) CDF or PDF evaluations. Hence, the computational complexity of this method is \( O(rN) \) for computing a mixture CDF or PDF of \( N \) observations. For \( r \ll N \), the RANDOM method is significantly faster than the FULL method.

**Empirical Distribution Function Estimation Methods**

The empirical distribution function (EDF) is a nonparametric estimate of the cumulative distribution function (CDF) of the distribution. PROC SEVSELECT computes EDF estimates for two purposes: to send the estimates to a distribution’s PARMINIT subroutine in order to initialize the distribution parameters, and to compute the EDF-based statistics of fit.

To reduce the time that it takes to compute the EDF estimates, you can use the INITSAMPLE option to specify that only a fraction of the input data be used. If you do not specify the INITSAMPLE option and the data table has more than 10,000 valid observations, then a uniform random sample of at most 10,000 observations is used for EDF estimation.

In the distributed mode of execution, in which data are distributed across the grid nodes, the EDF estimates are computed on each node by using the portion of the input data that is located on that node. These local EDF estimates are an approximation of the global EDF estimates, which would be computed by using the entire input data table. PROC SEVSELECT does not compute global EDF estimates. Let \( X \) denote a quantity that depends on the EDF estimates. \( X \) can be either an EDF-based initial value of a distribution parameter or an EDF-based statistic of fit. PROC SEVSELECT estimates \( X \) as follows: First, each grid node \( k \) computes an estimate \( X_k \) by using the local EDF estimates that are computed on that node. Then, the estimate \( \hat{X} \) of \( X \) is computed as an average of all the \( X_k \) values; that is, \( \hat{X} = \sum_{i=1}^{K} X_k \), where \( K \) denotes the total number of nodes where the data reside.

This section describes the methods that are used for computing EDF estimates.

**Notation**

Let there be a set of \( N \) observations, each containing a quintuplet of values \((y_i, t^l_i, t^r_i, c^r_i, c^l_i)\), \( i = 1, \ldots, N \), where \( y_i \) is the value of the response variable, \( t^l_i \) is the value of the left-truncation threshold, \( t^r_i \) is the value of the right-truncation threshold, \( c^r_i \) is the value of the right-censoring limit, and \( c^l_i \) is the value of the left-censoring limit.

If an observation is not left-truncated, then \( t^l_i = t^l \), where \( t^l \) is the smallest value in the support of the distribution; so \( F(t^l) = 0 \). If an observation is not right-truncated, then \( t^r_i = t^r \), where \( t^r \) is the largest value in the support of the distribution; so \( F(t^r) = 1 \). If an observation is not right-censored, then \( c^r_i = t^r \); so \( F(c^r) = 0 \). If an observation is not left-censored, then \( c^l_i = t^l \); so \( F(c^l) = 1 \).

Let \( w_i \) denote the weight associated with \( i \)th observation. If you specify the WEIGHT statement, then \( w_i \) is the normalized value of the weight variable; otherwise, it is set to 1. The weights are normalized such that they sum up to \( N \).

An indicator function \( I[e] \) takes a value of 1 or 0 if the expression \( e \) is true or false, respectively.
Estimation Methods

If the response variable is subject to both left-censoring and right-censoring effects and if you explicitly specify the `EMPIRICALCDF=TURNBULL` option, then PROC SEVSELECT uses the Turnbull’s method. This section describes methods other than Turnbull’s method. For Turnbull’s method, see the next section “Turnbull’s EDF Estimation Method” on page 264.

The method descriptions assume that all observations are either uncensored or right-censored; that is, each observation is of the form \((y_i, t_i^l, t_i^r, \tau_i, \tau_h)\) or \((y_i, t_i^l, c_i^r, \tau_h)\).

If all observations are either uncensored or left-censored, then each observation is of the form \((y_i, t_i^l, t_i^r, \tau_l, \tau_i)\). It is converted to an observation \((-y_i, -t_i^l, -t_i^r, -c_i^r, \tau_i)\); that is, the signs of all the response variable values are reversed, the new left-truncation threshold is equal to the negative of the original right-truncation threshold, the new right-truncation threshold is equal to the negative of the original left-truncation threshold, and the negative of the original left-censoring limit becomes the new right-censoring limit. With this transformation, each observation is either uncensored or right-censored. The methods described for handling uncensored or right-censored data are now applicable.

Further, a set of uncensored or right-censored observations can be converted to a set of observations of the form \((y_i, t_i^l, t_i^r, \tau_i, \delta_i)\), where \(\delta_i\) is the indicator of right-censoring. \(\delta_i = 0\) indicates a right-censored observation, in which case \(y_i\) is assumed to record the right-censoring limit \(c_i^r\). \(\delta_i = 1\) indicates an uncensored observation, and \(y_i\) records the exact observed value. In other words, \(\delta_i = I[Y \leq C^r]\) and \(y_i = \min(y_i, c_i^r)\).

Given this notation, the EDF is estimated as

\[
F_n(y) = \begin{cases} 
0 & \text{if } y < y^{(1)} \\
\hat{F}_n(y^{(k)}) & \text{if } y^{(k)} \leq y < y^{(k+1)}, k = 1, \ldots, N - 1 \\
\hat{F}_n(y^{(N)}) & \text{if } y^{(N)} \leq y 
\end{cases}
\]

where \(y^{(k)}\) denotes the \(k\)th-order statistic of the set \(\{y_i\}\) and \(\hat{F}_n(y^{(k)})\) is the estimate computed at that value. The definition of \(\hat{F}_n\) depends on the estimation method. You can specify a particular method or let PROC SEVSELECT choose an appropriate method by using the `EMPIRICALCDF=` option in the PROC SEVSELECT statement. Each method computes \(\hat{F}_n\) as follows:

- **NOTURNBULL** This is the default method. First, censored observations, if any, are processed as follows:
  - An observation that is left-censored but not right-censored is converted to an uncensored observation \((y_i^u, t_i^l, t_i^r, \tau^l, \tau_h)\), where \(y_i^u = c_i^l/2\).
  - An observation that is both left-censored and right-censored is converted to an uncensored observation \((y_i^u, t_i^l, t_i^r, \tau^l, \tau_h)\), where \(y_i^u = (c_i^r + c_i^l)/2\).
  - An observation that is right-censored but not left-censored is left unchanged.

If the processed set of observations contains any truncated or right-censored observations, the KAPLANMEIER method is used. Otherwise, the STANDARD method is used.

The observations are modified only for the purpose of computing the EDF estimates. The original censoring information is used by the parameter estimation process.
STANDARD This method is the standard way of computing EDF. The EDF estimate at observation \( i \) is computed as follows:

\[
\hat{F}_n(y_i) = \frac{1}{N} \sum_{j=1}^{N} w_j \cdot I[y_j \leq y_i]
\]

If you do not specify any censoring or truncation information, then this method is chosen. If you explicitly specify this method, then PROC SEVSELECT ignores any censoring and truncation information that you specify in the LOSS statement.

The standard error of \( \hat{F}_n(y_i) \) is computed by using the normal approximation method:

\[
\hat{\sigma}_n(y_i) = \sqrt{\frac{\hat{F}_n(y_i)(1 - \hat{F}_n(y_i))}{N}}
\]

KAPLANMEIER The Kaplan-Meier (KM) estimator, also known as the product-limit estimator, was first introduced by Kaplan and Meier (1958) for censored data. Lynden-Bell (1971) derived a similar estimator for left-truncated data. PROC SEVSELECT uses the definition that combines both censoring and truncation information (Klein and Moeschberger 1997; Lai and Ying 1991).

The EDF estimate at observation \( i \) is computed as

\[
\hat{F}_n(y_i) = 1 - \prod_{\tau \leq y_i} \left( 1 - \frac{n(\tau)}{R_n(\tau)} \right)
\]

where \( n(\tau) \) and \( R_n(\tau) \) are defined as follows:

- \( n(\tau) = \sum_{k=1}^{N} w_k \cdot I[y_k = \tau \text{ and } \tau \leq t_k^r \text{ and } \delta_k = 1] \), which is the number of uncensored observations (\( \delta_k = 1 \)) for which the response variable value is equal to \( \tau \) and \( \tau \) is observable according to the right-truncation threshold of that observation (\( \tau \leq t_k^r \)).

- \( R_n(\tau) = \sum_{k=1}^{N} w_k \cdot I[y_k \geq \tau > t_k^l] \), which is the size (cardinality) of the risk set at \( \tau \). The term risk set has its origins in survival analysis; it contains the events that are at risk of failure at a particular time, \( \tau \). In other words, it contains the events that have survived up to time \( \tau \) and might fail at or after \( \tau \). For PROC SEVSELECT, time is equivalent to the magnitude of the event and failure is equivalent to an uncensored and observable event, where observable means it satisfies the truncation thresholds.

This method is chosen when you specify at least one form of censoring or truncation.

The standard error of \( \hat{F}_n(y_i) \) is computed by using Greenwood’s formula (Greenwood 1926):

\[
\hat{\sigma}_n(y_i) = \sqrt{\frac{(1 - \hat{F}_n(y_i))^2 \cdot \sum_{\tau \leq y_i} \left( \frac{n(\tau)}{R_n(\tau)(R_n(\tau) - n(\tau))} \right)}{\sum_{\tau \leq y_i} \left( \frac{n(\tau)}{R_n(\tau)(R_n(\tau) - n(\tau))} \right)}}
\]

MODIFIEDKM The product-limit estimator used by the KAPLANMEIER method does not work well if the risk set size becomes very small. For right-censored data, the size can become
small towards the right tail. For left-truncated data, the size can become small at the left tail and can remain so for the entire range of data. This was demonstrated by Lai and Ying (1991). They proposed a modification to the estimator that ignores the effects due to small risk set sizes.

The EDF estimate at observation $i$ is computed as

$$\hat{F}_n(y_i) = 1 - \prod_{\tau \leq y_i} \left( 1 - \frac{n(\tau)}{R_n(\tau)} \cdot I[R_n(\tau) \geq c N^\alpha] \right)$$

where the definitions of $n(\tau)$ and $R_n(\tau)$ are identical to those used for the KAPLAN-MEIER method described previously.

You can specify the values of $c$ and $\alpha$ by using the C= and ALPHA= options. If you do not specify a value for $c$, the default value used is $c = 1$. If you do not specify a value for $\alpha$, the default value used is $\alpha = 0.5$.

As an alternative, you can also specify an absolute lower bound, say $L$, on the risk set size by using the RSLB= option, in which case $I[R_n(\tau) \geq c N^\alpha] \cdot I[R_n(\tau) \geq L]$ in the definition.

The standard error of $\hat{F}_n(y_i)$ is computed by using Greenwood’s formula (Greenwood 1926):

$$\hat{\sigma}_n(y_i) = \sqrt{(1 - \hat{F}_n(y_i))^2 \cdot \sum_{\tau \leq y_i} \left( \frac{n(\tau)}{R_n(\tau)(R_n(\tau) - n(\tau))} \cdot I[R_n(\tau) \geq c N^\alpha] \right)}$$

**Turnbull’s EDF Estimation Method**

If the response variable is subject to both left-censoring and right-censoring effects and if you explicitly specify the EMPIRICALCDF=TURNBULL option, then the SEVSELECT procedure uses a method proposed by Turnbull (1976) to compute the nonparametric estimates of the cumulative distribution function. The original Turnbull’s method is modified using the suggestions made by Frydman (1994) when truncation effects are present.

Let the input data consist of $N$ observations in the form of quintuplets of values $(y_i, t_i^l, t_i^r, c_i^r, c_i^l), i = 1, \ldots, N$ with notation described in the section “Notation” on page 261. For each observation, let $A_i = (c_i^r, c_i^l)$ be the censoring interval; that is, the response variable value is known to lie in the interval $A_i$, but the exact value is not known. If an observation is uncensored, then $A_i = (y_i - \epsilon, y_i]$ for any arbitrarily small value of $\epsilon > 0$. If an observation is censored, then the value $y_i$ is ignored. Similarly, for each observation, let $B_i = (t_i^l, t_i^r]$ be the truncation interval; that is, the observation is drawn from a truncated (conditional) distribution $F(y, B_i) = P(Y \leq y | Y \in B_i)$.

Two sets, $L$ and $R$, are formed using $A_i$ and $B_i$ as follows:

$$L = \{c_i^r, 1 \leq i \leq N\} \cup \{t_i^r, 1 \leq i \leq N\}$$
$$R = \{c_i^l, 1 \leq i \leq N\} \cup \{t_i^l, 1 \leq i \leq N\}$$

The sets $L$ and $R$ represent the left endpoints and right endpoints, respectively. A set of disjoint intervals $C_j = [q_j, p_j], 1 \leq j \leq M$ is formed such that $q_j \in L$ and $p_j \in R$ and $q_j < p_j$ and $p_j < q_{j+1}$. The value of $M$ is dependent on the nature of censoring and truncation intervals in the input data. Turnbull (1976)
showed that the maximum likelihood estimate (MLE) of the EDF can increase only inside intervals \( C_j \). In other words, the MLE estimate is constant in the interval \( p_j; q_j \). The likelihood is independent of the behavior of \( F_n \) inside any of the intervals \( C_j \). Let \( s_j \) denote the increase in \( F_n \) inside an interval \( C_j \). Then, the EDF estimate is as follows:

\[
F_n(y) = \begin{cases} 
0 & \text{if } y < q_1 \\
\sum_{k=1}^{j} s_k & \text{if } p_j < y < q_{j+1}, 1 \leq j \leq M - 1 \\
1 & \text{if } y > p_M
\end{cases}
\]

PROC SEVSELECT computes the estimates \( F_n(p_j) = F_n(q_{j+1}^{-}) = \sum_{k=1}^{j} s_k \) at points \( p_j \) and \( q_{j+1}^{-} \) and computes \( F_n(q_1^{-}) = 0 \) at point \( q_1 \), where \( F_n(x) \) denotes the limiting estimate at a point that is infinitesimally larger than \( x \) when approaching \( x \) from values larger than \( x \) and where \( F_n(x^{-}) \) denotes the limiting estimate at a point that is infinitesimally smaller than \( x \) when approaching \( x \) from values smaller than \( x \).

PROC SEVSELECT uses the expectation-maximization (EM) algorithm proposed by Turnbull (1976), who referred to the algorithm as the self-consistency algorithm. By default, the algorithm runs until one of the following criteria is met:

- **Relative-error criterion:** The maximum relative error between the two consecutive estimates of \( s_j \) falls below a threshold \( \epsilon \). If \( l \) indicates an index of the current iteration, then this can be formally stated as

\[
\arg \max_{1 \leq j \leq M} \left\{ \frac{|s_j^l - s_j^{l-1}|}{s_j^{l-1}} \right\} \leq \epsilon
\]

You can control the value of \( \epsilon \) by specifying the **EPS=** suboption of the **EDF=TURNBULL** option in the PROC SEVSELECT statement. The default value is 1.0E–8.

- **Maximum-iteration criterion:** The number of iterations exceeds an upper limit that you specify for the **MAXITER=** suboption of the **EDF=TURNBULL** option in the PROC SEVSELECT statement. The default number of maximum iterations is 500.

The self-consistent estimates obtained in this manner might not be maximum likelihood estimates. Gentleman and Geyer (1994) suggested the use of the Kuhn-Tucker conditions for the maximum likelihood problem to ensure that the estimates are MLE. If you specify the **ENSUREMLE** suboption of the **EDF=TURNBULL** option in the PROC SEVSELECT statement, then PROC SEVSELECT computes the Kuhn-Tucker conditions at the end of each iteration to determine whether the estimates \( \{s_j\} \) are MLE. If you do not specify any truncation effects, then the Kuhn-Tucker conditions derived by Gentleman and Geyer (1994) are used. If you specify any truncation effects, then PROC SEVSELECT uses modified Kuhn-Tucker conditions that account for the truncation effects. An integral part of checking the conditions is to determine whether an estimate \( s_j \) is zero or whether an estimate of the Lagrange multiplier or the reduced gradient associated with the estimate \( s_j \) is zero. PROC SEVSELECT declares these values to be zero if they are less than or equal to a threshold \( \delta \). You can control the value of \( \delta \) by specifying the **ZEROPROB=** suboption of the **EDF=TURNBULL** option in the PROC SEVSELECT statement. The default value is 1.0E–8. The algorithm continues until the Kuhn-Tucker conditions are satisfied or the number of iterations exceeds the upper limit.

The relative-error criterion stated previously is not used when you specify the **ENSUREMLE** option.

The standard errors for Turnbull’s EDF estimates are computed by using the asymptotic theory of the maximum likelihood estimators (MLE), even though the final estimates might not be MLE. Turnbull’s
Supplying EDF Estimates to Functions and Subroutines

EDF Estimates and Truncation

Denote the (unknown) distribution functions of the left-truncation threshold variable \( H \). The parameter initialization subroutines in distribution models and some predefined utility functions require \( \hat{F}_n(y) \), the EDF estimates. For more information, see the sections “Defining a Severity Distribution Model with the FCMP Procedure” on page 272 and “Predefined Utility Functions” on page 285.

If you specify truncation, then the estimate \( \hat{F}_n(y) \) that is computed by any method other than the STANDARD method is a conditional estimate. In other words, \( \hat{F}_n(y) = \text{Pr}(Y \leq y | \tau_G < Y \leq \tau_H) \), where \( G \) and \( H \) denote the (unknown) distribution functions of the left-truncation threshold variable \( T^l \) and the right-truncation threshold variable \( T^r \), respectively, \( \tau_G \) denotes the smallest left-truncation threshold with a nonzero cumulative probability, and \( \tau_H \) denotes the largest right-truncation threshold with a nonzero cumulative probability. Formally, \( \tau_G = \inf\{s : G(s) > 0\} \) and \( \tau_H = \sup\{s : H(s) > 0\} \).

If you specify truncation with the probability of observability \( p \), then PROC SEVSELECT uses the additional information provided by \( p \) to compute an estimate of the EDF that is not conditional on the left-truncation information. In particular, for each left-truncated observation \( i \) with response variable value \( y_i \) and truncation threshold \( t_i^l \), an observation \( j \) is added with weight \( w_j = (1 - p)/p \) and \( y_j = t_i^l \). Each added observation is assumed to be uncensored and untruncated. Then, your specified EDF method is used by assuming no left-truncation. The EDF estimate that is obtained using this method is not conditional on the left-truncation information. For the KAPLANMEIER and MODIFIEDKM methods with uncensored or right-censored data, definitions of \( n(\tau) \) and \( R_n(\tau) \) are modified to account for the added observations. If \( N^a \) denotes the total number of observations including the added observations, then \( n(\tau) \) is defined as \( n(\tau) = \sum_{k=1}^{N^a} w_k I[y_k = \tau \text{ and } \tau \leq t_i^l \text{ and } \delta_k = 1] \), and \( R_n(\tau) \) is defined as \( R_n(\tau) = \sum_{k=1}^{N^a} w_k I[y_k \geq \tau] \). In the definition of \( R_n(\tau) \), the left-truncation information is not used, because it was used along with \( p \) to add the observations.

If the original data are a combination of left- and right-censored data and if you specify the EMPIRICALCDF=TURBULL option, then Turnbull’s method is applied to the appended set that contains no left-truncated observations.

Supplying EDF Estimates to Functions and Subroutines

The parameter initialization subroutines in distribution models and some predefined utility functions require EDF estimates. For more information, see the sections “Defining a Severity Distribution Model with the FCMP Procedure” on page 272 and “Predefined Utility Functions” on page 285.
PROC SEVSELECT supplies the EDF estimates to these subroutines and functions by using two arrays, \( x \) and \( F \), the dimension of each array, and a type of the EDF estimates. The type identifies how the EDF estimates are computed and stored. They are as follows:

**Type 1** specifies that EDF estimates are computed using the STANDARD method; that is, the data that are used for estimation are neither censored nor truncated.

**Type 2** specifies that EDF estimates are computed using either the KAPLANMEIER or the MODIFIEDKM method; that is, the data that are used for estimation are subject to truncation and one type of censoring (left or right, but not both).

**Type 3** specifies that EDF estimates are computed using the TURNBULL method; that is, the data that are used for estimation are subject to both left- and right-censoring. The data might or might not be truncated.

For Types 1 and 2, the EDF estimates are stored in arrays \( x \) and \( F \) of dimension \( N \) such that the following holds,

\[
F_n(y) = \begin{cases} 
0 & \text{if } y < x[1] \\
F[k] & \text{if } x[k] \leq y < x[k+1], k = 1, \ldots, N - 1 \\
F[N] & \text{if } x[N] \leq y
\end{cases}
\]

where \([k]\) denotes \( k \)th element of the array (\([1]\) denotes the first element of the array).

For Type 3, the EDF estimates are stored in arrays \( x \) and \( F \) of dimension \( N \) such that the following holds:

\[
F_n(y) = \begin{cases} 
0 & \text{if } y < x[1] \\
\text{undefined} & \text{if } x[2k - 1] \leq y < x[2k], k = 1, \ldots, (N - 1)/2 \\
F[2k] = F[2k + 1] & \text{if } x[2k] \leq y < x[2k + 1], k = 1, \ldots, (N - 1)/2 \\
F[N] & \text{if } x[N] \leq y
\end{cases}
\]

Although the behavior of EDF is theoretically undefined for the interval \([x[2k - 1], x[2k]]\), for computational purposes, all predefined functions and subroutines assume that the EDF increases linearly from \( F[2k - 1] \) to \( F[2k] \) in that interval if \( x[2k - 1] < x[2k] \). If \( x[2k - 1] = x[2k] \), which can happen when the EDF is estimated from a combination of uncensored and interval-censored data, the predefined functions and subroutines assume that \( F_n(x[2k - 1]) = F_n(x[2k]) = F[2k] \).

**Statistics of Fit**

PROC SEVSELECT computes and reports various statistics of fit to indicate how well the estimated model fits the data. The statistics belong to two categories: likelihood-based statistics and EDF-based statistics. Neg2LogLike, AIC, AICC, and BIC are likelihood-based statistics, and KS, AD, and CvM are EDF-based statistics.

In the distributed mode of execution, in which data are distributed across the grid nodes, the EDF estimates are computed by using the local data. The EDF-based statistics are computed by using these local EDF estimates. The reported value of each EDF-based statistic is an average of the values of the statistic that are computed by all the grid nodes where the data reside. Also, for large data tables, in both single-machine and distributed modes of execution, the EDF estimates are computed by using a fraction of the input data that is governed by either the INITSAMPLE option or the default sample size. Because of this nature of computing
the EDF estimates, the EDF-based statistics of fit are an approximation of the values that would have been computed if the entire input data table were used for computing the EDF estimates. So the values that are reported for EDF-based statistics should be used only for comparing different models. The reported values should not be interpreted as true estimates of the corresponding statistics.

The likelihood-based statistics are reported for the entire input data in both single-machine and distributed modes of execution.

The following subsections provide definitions of each category of statistics.

**Likelihood-Based Statistics of Fit**

Let \( y_i, i = 1, \ldots, N \), denote the response variable values. Let \( L \) be the likelihood as defined in the section “Likelihood Function” on page 253. Let \( p \) denote the number of model parameters that are estimated. Note that \( p = p_d + (k - k_r) \), where \( p_d \) is the number of distribution parameters, \( k \) is the number of all regression parameters, and \( k_r \) is the number of regression parameters that are found to be linearly dependent (redundant) on other regression parameters. Given this notation, the likelihood-based statistics are defined as follows:

- **Neg2LogLike**
  The log likelihood is reported as
  \[
  \text{Neg2LogLike} = -2 \log(L)
  \]
  The multiplying factor \(-2\) makes it easy to compare it to the other likelihood-based statistics. A model that has a smaller value of Neg2LogLike is deemed better.

- **AIC**
  Akaike’s information criterion (AIC) is defined as
  \[
  \text{AIC} = -2 \log(L) + 2p
  \]
  A model that has a smaller AIC value is deemed better.

- **AICC**
  The corrected Akaike’s information criterion (AICC) is defined as
  \[
  \text{AICC} = -2 \log(L) + \frac{2Np}{N - p - 1}
  \]
  A model that has a smaller AICC value is deemed better. It corrects the finite-sample bias that AIC has when \( N \) is small compared to \( p \). AICC is related to AIC as
  \[
  \text{AICC} = \text{AIC} + \frac{2p(p + 1)}{N - p - 1}
  \]
  As \( N \) becomes large compared to \( p \), AICC converges to AIC. AICC is usually recommended over AIC as a model selection criterion.

- **BIC**
  The Schwarz Bayesian information criterion (BIC) is defined as
  \[
  \text{BIC} = -2 \log(L) + p \log(N)
  \]
  A model that has a smaller BIC value is deemed better.
EDF-Based Statistics

This class of statistics is based on the difference between the estimate of the cumulative distribution function (CDF) and the estimate of the empirical distribution function (EDF). A model that has a smaller value of the chosen EDF-based statistic is deemed better.

Let \( y_i, i = 1, \ldots, N \), denote the sample of \( N \) values of the response variable. Let \( w_i \) denote the normalized weight of the \( i \)th observation. If \( w_i^o \) denotes the original, unnormalized weight of the \( i \)th observation, then
\[
 w_i = N w_i^o / (\sum_{i=1}^{N} w_i^o).
\]
Let \( N_u \) denote the number of observations with unique (nonduplicate) values of the response variable. Let \( W_i = \sum_{j=1}^{N} w_j I[y_j = y_i] \) denote the total weight of observations with a value \( y_i \), where \( I \) is an indicator function. Let \( r_i = \sum_{j=1}^{N} w_j I[y_j \leq y_i] \) denote the total weight of observations with a value less than or equal to \( y_i \). Let \( W = \sum_{i=1}^{N_u} W_i \) denote the total weight of all observations. Use of normalized weights implies that \( W = N \).

Let \( F_n(y_i) \) denote the EDF estimate that is computed by using the method that you specify in the EMPIRICALCDF= option. Let \( Z_i = \hat{F}(y_i) \) denote the estimate of the CDF. Let \( F_n(Z_i) \) denote the EDF estimate of \( Z_i \) values that are computed using the same method that is used to compute the EDF of \( y_i \) values. Using the probability integral transformation, if \( F(y) \) is the true distribution of the random variable \( Y \), then the random variable \( Z = F(Y) \) is uniformly distributed between 0 and 1 (D’Agostino and Stephens 1986, Ch. 4). Thus, comparing \( F_n(y_i) \) with \( \hat{F}(y_i) \) is equivalent to comparing \( F_n(Z_i) \) with \( \hat{F}(Z_i) = Z_i \) (uniform distribution).

Note the following two points regarding which CDF estimates are used for computing the test statistics:

- If you specify regression effects, then the CDF estimates \( Z_i \) that are used for computing the EDF test statistics are from a mixture distribution. For more information, see the section “CDF and PDF Estimates with Regression Effects” on page 259.

- If the EDF estimates are conditional because of the truncation information, then each unconditional estimate \( Z_i \) is converted to a conditional estimate using the method described in the section “Truncation and Conditional CDF Estimates” on page 252.

In the following, it is assumed that \( Z_i \) denotes an appropriate estimate of the CDF if you specify any truncation or regression effects. Given this, the EDF-based statistics of fit are defined as follows:

KS

The Kolmogorov-Smirnov (KS) statistic computes the largest vertical distance between the CDF and the EDF. It is formally defined as follows:

\[
 KS = \sup_y |F_n(y) - F(y)|
\]

If the STANDARD method is used to compute the EDF, then the following formula is used:

\[
 D^+ = \max_i \left( \frac{r_i}{W} - Z_i \right)
\]
\[
 D^- = \max_i \left( Z_i - \frac{r_i-1}{W} \right)
\]
\[
 KS = \sqrt{W} \max(D^+, D^-) + \frac{0.19}{\sqrt{W}}
\]

Note that \( r_0 \) is assumed to be 0.
If the method used to compute the EDF is any method other than the STANDARD method, then the following formula is used:
\[
D^+ = \max_i \left( F_n(Z_i) - Z_i \right), \text{ if } F_n(Z_i) \geq Z_i \\
D^- = \max_i \left( Z_i - F_n(Z_i) \right), \text{ if } F_n(Z_i) < Z_i \\
KS = \sqrt{W} \max(D^+, D^-) + \frac{0.19}{\sqrt{W}}
\]

The Anderson-Darling (AD) statistic is a quadratic EDF statistic that is proportional to the expected value of the weighted squared difference between the EDF and CDF. It is formally defined as follows:
\[
AD = N \int_{-\infty}^{\infty} \frac{(F_n(y) - F(y))^2}{F(y)(1 - F(y))} dF(y)
\]

If the STANDARD method is used to compute the EDF, then the following formula is used:
\[
AD = -W - \frac{1}{W} \sum_{i=1}^{N_n} W_i \left[ (2r_i - 1) \log(Z_i) + (2W + 1 - 2r_i) \log(1 - Z_i) \right]
\]

If the method used to compute the EDF is any method other than the STANDARD method, then the statistic can be computed by using the following two pieces of information:

- If the EDF estimates are computed using the KAPLANMEIER or MODIFIEDKM methods, then EDF is a step function such that the estimate \( F_n(z) \) is a constant equal to \( F_n(Z_{i-1}) \) in interval \([Z_{i-1}, Z_i]\). If the EDF estimates are computed using the TURNBULL method, then there are two types of intervals: one in which the EDF curve is constant and the other in which the EDF curve is theoretically undefined. For computational purposes, it is assumed that the EDF curve is linear for the latter type of the interval. For each method, the EDF estimate \( F_n(y) \) at \( y \) can be written as
  \[
  F_n(z) = F_n(Z_{i-1}) + S_i(z - Z_{i-1}), \text{ for } z \in [Z_{i-1}, Z_i]
  \]
  where \( S_i \) is the slope of the line defined as
  \[
  S_i = \frac{F_n(Z_i) - F_n(Z_{i-1})}{Z_i - Z_{i-1}}
  \]
  For the KAPLANMEIER or MODIFIEDKM method, \( S_i = 0 \) in each interval.

- Using the probability integral transform \( z = F(y) \), the formula simplifies to
  \[
  AD = N \int_{-\infty}^{\infty} \frac{(F_n(z) - z)^2}{z(1 - z)} dz
  \]
  The computation formula can then be derived from the approximation,
  \[
  AD = N \sum_{i=1}^{K+1} \int_{Z_{i-1}}^{Z_i} \frac{(F_n(z) - z)^2}{z(1 - z)} dz
  = N \sum_{i=1}^{K+1} \int_{Z_{i-1}}^{Z_i} \frac{(F_n(Z_{i-1}) + S_i(z - Z_{i-1}) - z)^2}{z(1 - z)} dz
  = N \sum_{i=1}^{K+1} \int_{Z_{i-1}}^{Z_i} \frac{(P_i - Q_i z)^2}{z(1 - z)} dz
  \]
where \( P_i = F_n(Z_{i-1}) - S_i Z_{i-1}, Q_i = 1 - S_i, \) and \( K \) is the number of points at which the EDF estimate are computed. For the TURNBULL method, \( K = 2k \) for some \( k \).

Assuming \( Z_0 = 0, Z_K+1 = 1, F_n(0) = 0, \) and \( F_n(Z_K) = 1 \) yields the computation formula,

\[
AD = - N(Z_1 + \log(1 - Z_1) + \log(Z_K) + (1 - Z_K)) + N \sum_{i=2}^{K} \left[ P_i^2 A_i - (Q_i - P_i)^2 B_i - Q_i^2 C_i \right]
\]

where \( A_i = \log(Z_i) - \log(Z_{i-1}), B_i = \log(1 - Z_i) - \log(1 - Z_{i-1}), \) and \( C_i = Z_i - Z_{i-1} \).

If EDF estimates are computed using the KAPLANMEIER or MODIFIEDKM method, then \( P_i = F_n(Z_{i-1}) \) and \( Q_i = 1, \) which simplifies the formula as

\[
AD = - N(1 + \log(1 - Z_1) + \log(Z_K)) + N \sum_{i=2}^{K} \left[ F_n(Z_{i-1})^2 A_i - (1 - F_n(Z_{i-1}))^2 B_i \right]
\]

**CvM**

The Cramér–von Mises (CvM) statistic is a quadratic EDF statistic that is proportional to the expected value of the squared difference between the EDF and CDF. It is formally defined as follows:

\[
C_{\text{V M}} = N \int_{-\infty}^{\infty} (F_n(y) - F(y))^2 dF(y)
\]

If the STANDARD method is used to compute the EDF, then the following formula is used:

\[
C_{\text{V M}} = \frac{1}{12W} + \sum_{i=1}^{N_u} W_i \left( Z_i - \frac{(2r_i - 1)}{2W} \right)^2
\]

If the method used to compute the EDF is any method other than the STANDARD method, then the statistic can be computed by using the following two pieces of information:

- As described previously for the AD statistic, the EDF estimates are assumed to be piecewise linear such that the estimate \( F_n(y) \) at \( y \) is

\[
F_n(z) = F_n(Z_{i-1}) + S_i (z - Z_{i-1}), \text{ for } z \in [Z_{i-1}, Z_i]
\]

where \( S_i \) is the slope of the line defined as

\[
S_i = \frac{F_n(Z_i) - F_n(Z_{i-1})}{Z_i - Z_{i-1}}
\]

For the KAPLANMEIER or MODIFIEDKM method, \( S_i = 0 \) in each interval.

- Using the probability integral transform \( z = F(y) \), the formula simplifies to

\[
C_{\text{V M}} = N \int_{-\infty}^{\infty} (F_n(z) - z)^2 dz
\]
The computation formula can then be derived from the following approximation,

\[
\text{CvM} = N \sum_{i=1}^{K+1} \int_{Z_{i-1}}^{Z_i} (F_n(z) - z)^2 \, dz
\]

where

\[
F_n(z) = \frac{1}{N} \sum_{j=1}^{N} I(Z_j \leq z)
\]

and

\[
I(A) = \begin{cases} 
1 & \text{if } A \text{ is true,} \\
0 & \text{otherwise.}
\end{cases}
\]

Assuming \(Z_0 = 0\), \(Z_{K+1} = 1\), and \(F_n(0) = 0\) yields the following computation formula,

\[
\text{CvM} = N \left( \frac{Z_3}{3} + \sum_{i=2}^{K+1} \left( P_i^2 A_i - P_i Q_i B_i - \frac{Q_i^2}{3} C_i \right) \right)
\]

where

\[
A_i = Z_i - Z_{i-1}, \quad B_i = Z_i^2 - Z_{i-1}^2, \quad C_i = Z_i^3 - Z_{i-1}^3.
\]

If EDF estimates are computed using the KAPLANMEIER or MODIFIEDKM method, then \(P_i = F_n(Z_{i-1})\) and \(Q_i = 1\), which simplifies the formula as

\[
\text{CvM} = \frac{N}{3} + N \sum_{i=2}^{K+1} \left[ F_n(Z_{i-1})^2 (Z_i - Z_{i-1}) - F_n(Z_{i-1}) (Z_i^2 - Z_{i-1}^2) \right]
\]

which is similar to the formula proposed by Koziol and Green (1976).

---

**Defining a Severity Distribution Model with the FCMP Procedure**

A severity distribution model consists of a set of functions and subroutines that are defined using the FCMP procedure. The FCMP procedure is part of Base SAS software. Each function or subroutine must be named as \(<\text{distribution-name}>_{-<\text{keyword}>}\), where \text{distribution-name} is the identifying short name of the distribution and \text{keyword} identifies one of the functions or subroutines. The total length of the name should not exceed 32. Each function or subroutine must have a specific signature, which consists of the number of arguments, sequence and types of arguments, and return value type. The summary of all the recognized function and subroutine names and their expected behavior is given in Table 6.6.

Consider the following points when you define a distribution model:

- When you define a function or subroutine requiring parameter arguments, the names and order of those arguments must be the same. Arguments other than the parameter arguments can have any name, but they must satisfy the requirements on their type and order.
When the SEVSELECT procedure invokes any function or subroutine, it provides the necessary input values according to the specified signature, and expects the function or subroutine to prepare the output and return it according to the specification of the return values in the signature.

You can use most of the SAS programming statements and SAS functions that you can use in a DATA step for defining the FCMP functions and subroutines. However, there are a few differences in the capabilities of the DATA step and the FCMP procedure. To learn more, see the documentation of the FCMP procedure in the Base SAS Procedures Guide.

You must specify either the PDF or the LOGPDF function. Similarly, you must specify either the CDF or the LOGCDF function. All other functions are optional, except when necessary for correct definition of the distribution. It is strongly recommended that you define the PARMINIT subroutine to provide a good set of initial values for the parameters. The information that PROC SEVSELECT provides to the PARMINIT subroutine enables you to use popular initialization approaches based on the method of moments and the method of percentile matching, but you can implement any algorithm to initialize the parameters by using the values of the response variable and the estimate of its empirical distribution function.

The LOWERBOUNDS subroutines should be defined if the lower bound on at least one distribution parameter is different from the default lower bound of 0. If you define a LOWERBOUNDS subroutine but do not set a lower bound for some parameter inside the subroutine, then that parameter is assumed to have no lower bound (or a lower bound of \(-\infty\)). Hence, it is recommended that you explicitly return the lower bound for each parameter when you define the LOWERBOUNDS subroutine.

The UPPERBOUNDS subroutines should be defined if the upper bound on at least one distribution parameter is different from the default upper bound of \(\infty\). If you define an UPPERBOUNDS subroutine but do not set an upper bound for some parameter inside the subroutine, then that parameter is assumed to have no upper bound (or a upper bound of \(\infty\)). Hence, it is recommended that you explicitly return the upper bound for each parameter when you define the UPPERBOUNDS subroutine.

If you want to use the distribution in a model with regression effects, then make sure that the first parameter of the distribution is the scale parameter itself or a log-transformed scale parameter. If the first parameter is a log-transformed scale parameter, then you must define the SCALETRANSFORM function.

In general, it is not necessary to define the gradient and Hessian functions, because the SEVSELECT procedure uses an internal system to evaluate the required derivatives. The internal system typically computes the derivatives analytically. But it might not be able to do so if your function definitions use other functions that it cannot differentiate analytically. In such cases, derivatives are approximated using a finite difference method and a note is written to the SAS log to indicate the components that are differentiated using such approximations. PROC SEVSELECT does reasonably well with these finite difference approximations. But, if you know of a way to compute the derivatives of such components analytically, then you should define the gradient and Hessian functions.
In order to use your distribution with PROC SEVSELECT, you need to record the FCMP library that contains the functions and subroutines for your distribution and other FCMP libraries that contain FCMP functions or subroutines used within your distribution’s functions and subroutines. Specify all those libraries in the CMPLIB= system option by using the OPTIONS global statement. For more information about the OPTIONS statement, see SAS Viya Statements: Reference. For more information about the CMPLIB= system option, see SAS Viya System Options: Reference.

Each predefined distribution mentioned in the section “Predefined Distributions” on page 241 has a distribution model associated with it. The functions and subroutines of all those models are available in the SasHelp.Svrtdist library. The order of the parameters in the signatures of the functions and subroutines is the same as listed in Table 6.4. You do not need to use the CMPLIB= option in order to use the predefined distributions with PROC SEVSELECT. However, if you need to use the functions or subroutines of the predefined distributions in SAS statements other than the PROC SEVSELECT step (such as in a DATA step), then specify the SasHelp.Svrtdist library in the CMPLIB= system option by using the OPTIONS global statement prior to using them.

Table 6.6 shows functions and subroutines that define a distribution model, and subsections after the table provide more detail. The functions are listed in alphabetical order of the keyword suffix.
Table 6.6  List of Functions and Subroutines That Define a Distribution Model

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Required</th>
<th>Expected to Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>dist_CDF</td>
<td>Function</td>
<td>YES¹</td>
<td>Cumulative distribution function value</td>
</tr>
<tr>
<td>dist_CDFGRADIENT</td>
<td>Subroutine</td>
<td>NO</td>
<td>Gradient of the CDF</td>
</tr>
<tr>
<td>dist_CDFHESSIAN</td>
<td>Subroutine</td>
<td>NO</td>
<td>Hessian of the CDF</td>
</tr>
<tr>
<td>dist_CONSTANTPARM</td>
<td>Subroutine</td>
<td>NO</td>
<td>Constant parameters</td>
</tr>
<tr>
<td>dist_DESCRIPTION</td>
<td>Function</td>
<td>NO</td>
<td>Description of the distribution</td>
</tr>
<tr>
<td>dist_LOGCDF</td>
<td>Function</td>
<td>YES¹</td>
<td>Log of cumulative distribution function value</td>
</tr>
<tr>
<td>dist_LOGCDFGRADIENT</td>
<td>Subroutine</td>
<td>NO</td>
<td>Gradient of the LOGCDF</td>
</tr>
<tr>
<td>dist_LOGCDFHESSIAN</td>
<td>Subroutine</td>
<td>NO</td>
<td>Hessian of the LOGCDF</td>
</tr>
<tr>
<td>dist_LOGPDF</td>
<td>Function</td>
<td>YES²</td>
<td>Log of probability density function value</td>
</tr>
<tr>
<td>dist_LOGPDFGRADIENT</td>
<td>Subroutine</td>
<td>NO</td>
<td>Gradient of the LOGPDF</td>
</tr>
<tr>
<td>dist_LOGPDFHESSIAN</td>
<td>Subroutine</td>
<td>NO</td>
<td>Hessian of the LOGPDF</td>
</tr>
<tr>
<td>dist_LOGSDF</td>
<td>Function</td>
<td>NO</td>
<td>Log of survival function value</td>
</tr>
<tr>
<td>dist_LOGSDFGRADIENT</td>
<td>Subroutine</td>
<td>NO</td>
<td>Gradient of the LOGSDF</td>
</tr>
<tr>
<td>dist_LOGSDFHESSIAN</td>
<td>Subroutine</td>
<td>NO</td>
<td>Hessian of the LOGSDF</td>
</tr>
<tr>
<td>dist_LOWERBOUNDS</td>
<td>Subroutine</td>
<td>NO</td>
<td>Lower bounds on parameters</td>
</tr>
<tr>
<td>dist_PARMINIT</td>
<td>Subroutine</td>
<td>NO</td>
<td>Initial values for parameters</td>
</tr>
<tr>
<td>dist_PDF</td>
<td>Function</td>
<td>YES²</td>
<td>Probability density function value</td>
</tr>
<tr>
<td>dist_PDFGRADIENT</td>
<td>Subroutine</td>
<td>NO</td>
<td>Gradient of the PDF</td>
</tr>
<tr>
<td>dist_PDFHESSIAN</td>
<td>Subroutine</td>
<td>NO</td>
<td>Hessian of the PDF</td>
</tr>
<tr>
<td>dist_QUANTILE</td>
<td>Function</td>
<td>NO</td>
<td>Quantile for a particular CDF value</td>
</tr>
<tr>
<td>dist_SCALETRANSFORM</td>
<td>Function</td>
<td>NO</td>
<td>Type of relationship between the first distribution parameter and the scale parameter</td>
</tr>
<tr>
<td>dist_SDF</td>
<td>Function</td>
<td>NO</td>
<td>Survival function value</td>
</tr>
<tr>
<td>dist_SDFGRADIENT</td>
<td>Subroutine</td>
<td>NO</td>
<td>Gradient of the SDF</td>
</tr>
<tr>
<td>dist_SDFHESSIAN</td>
<td>Subroutine</td>
<td>NO</td>
<td>Hessian of the SDF</td>
</tr>
<tr>
<td>dist_UPPERBOUNDS</td>
<td>Subroutine</td>
<td>NO</td>
<td>Upper bounds on parameters</td>
</tr>
</tbody>
</table>

Notes:
1. Either the dist_CDF or the dist_LOGCDF function must be defined.
2. Either the dist_PDF or the dist_LOGPDF function must be defined.

The signature syntax and semantics of each function or subroutine are as follows:

**dist_CDF**

defines a function that returns the value of the cumulative distribution function (CDF) of the distribution at the specified values of the random variable and distribution parameters.

- **Type**: Function
• **Required**: YES
• **Number of arguments**: \( m + 1 \), where \( m \) is the number of distribution parameters
• **Sequence and type of arguments**:
  - \( x \): Numeric value of the random variable at which the CDF value should be evaluated
  - \( p_1 \): Numeric value of the first parameter
  - \( p_2 \): Numeric value of the second parameter
  - \( \ldots \)
  - \( p_m \): Numeric value of the \( m \)th parameter

• **Return value**: Numeric value that contains the CDF value \( F(x; p_1, p_2, \ldots, p_m) \)

If you want to consider this distribution as a candidate distribution when you estimate a response variable model with regression effects, then the first parameter of this distribution must be a scale parameter or log-transformed scale parameter. In other words, if the distribution has a scale parameter, then the following equation must be satisfied:

\[
F(x; p_1, p_2, \ldots, p_m) = F\left( \frac{x}{p_1}; 1, p_2, \ldots, p_m \right)
\]

If the distribution has a log-transformed scale parameter, then the following equation must be satisfied:

\[
F(x; p_1, p_2, \ldots, p_m) = F\left( \frac{x}{\exp(p_1)}; 0, p_2, \ldots, p_m \right)
\]

Here is a sample structure of the function for a distribution named ‘FOO’:

```c
function FOO_CDF(x, P1, P2);
    /* Code to compute CDF by using x, P1, and P2 */
    F = <computed CDF>;
    return (F);
endsub;
```

dist_CONSTANTPARM
defines a subroutine that specifies constant parameters. A parameter is constant if it is required for defining a distribution but is not subject to optimization in PROC SEVSELECT. Constant parameters are required to be part of the model in order to compute the PDF or the CDF of the distribution. Typically, values of these parameters are known a priori or estimated using some means other than the maximum likelihood method used by PROC SEVSELECT. You can estimate them inside the dist_PARMINIT subroutine. Once initialized, the parameters remain constant in the context of PROC SEVSELECT; that is, they retain their initial value. PROC SEVSELECT estimates only the nonconstant parameters.

• **Type**: Subroutine
• **Required**: NO
• **Number of arguments**: \( k \), where \( k \) is the number of constant parameters
• **Sequence and type of arguments**:
constant parameter 1  Name of the first constant parameter

...  

constant parameter k  Name of the kth constant parameter

• Return value: None

Here is a sample structure of the subroutine for a distribution named ‘FOO’ that has P3 and P5 as its constant parameters, assuming that distribution has at least three parameters:

```plaintext
subroutine FOO_CONSTANTPARM(p5, p3);
endsub;
```

Note the following points when you specify the constant parameters:

• At least one distribution parameter must be free to be optimized; that is, if a distribution has total \( m \) parameters, then \( k \) must be strictly less than \( m \).
• If you want to use this distribution for modeling regression effects, then the first parameter must not be a constant parameter.
• The order of arguments in the signature of this subroutine does not matter as long as each argument’s name matches the name of one of the parameters that are defined in the signature of the \texttt{dist\_PDF} function.
• The constant parameters must be specified in signatures of all the functions and subroutines that accept distribution parameters as their arguments.
• You must provide a nonmissing initial value for each constant parameter by using one of the supported parameter initialization methods.

\texttt{dist\_DESCRIPTION}

defines a function that returns a description of the distribution.

• Type: Function
• Required: NO
• Number of arguments: None
• Sequence and type of arguments: Not applicable
• Return value: Character value containing a description of the distribution

Here is a sample structure of the function for a distribution named ‘FOO’:

```plaintext
function FOO_DESCRIPTION() $48;
   length desc $48;
   desc = "A model for a continuous distribution named foo";
   return (desc);
endsub;
```
**dist_LOGcore**

defines a function that returns the natural logarithm of the specified `core` function of the distribution at the specified values of the random variable and distribution parameters. The `core` keyword can be PDF, CDF, or SDF.

- **Type**: Function
- **Required**: YES only if `core` is PDF or CDF and you have not defined that `core` function; otherwise, NO
- **Number of arguments**: $m + 1$, where $m$ is the number of distribution parameters
- **Sequence and type of arguments**:
  - $x$ Numeric value of the random variable at which the natural logarithm of the `core` function should be evaluated
  - $p_1$ Numeric value of the first parameter
  - $p_2$ Numeric value of the second parameter
  - ...  
  - $p_m$ Numeric value of the $m$th parameter
- **Return value**: Numeric value that contains the natural logarithm of the `core` function

Here is a sample structure of the function for the core function PDF of a distribution named ‘FOO’:

```plaintext
function FOO_LOGPDF(x, P1, P2);
    /* Code to compute LOGPDF by using x, P1, and P2 */
    l = <computed LOGPDF>;
    return (l);
endsub;
```

**dist_LOWERBOUNDS**

defines a subroutine that returns lower bounds for the parameters of the distribution. If this subroutine is not defined for a particular distribution, then the SEVSELECT procedure assumes a lower bound of 0 for each parameter. If a lower bound of $l_i$ is returned for a parameter $p_i$, then the SEVSELECT procedure assumes that $l_i < p_i$ (strict inequality). If a missing value is returned for some parameter, then the SEVSELECT procedure assumes that there is no lower bound for that parameter (equivalent to a lower bound of $-\infty$).

- **Type**: Subroutine
- **Required**: NO
- **Number of arguments**: $m$, where $m$ is the number of distribution parameters
- **Sequence and type of arguments**:
  - $p_1$ Output argument that returns the lower bound on the first parameter. You must specify this in the OUTARGS statement inside the subroutine’s definition.
  - $p_2$ Output argument that returns the lower bound on the second parameter. You must specify this in the OUTARGS statement inside the subroutine’s definition.
  - ...
Output argument that returns the lower bound on the \( m \)th parameter. You must specify this in the OUTARGS statement inside the subroutine’s definition.

- **Return value**: The results, lower bounds on parameter values, should be returned in the parameter arguments of the subroutine.

Here is a sample structure of the subroutine for a distribution named ‘FOO’:

```
subroutine FOO_LOWERBOUNDS(p1, p2);
  outargs p1, p2;
  p1 = <lower bound for P1>;
  p2 = <lower bound for P2>;
endsub;
```

**dist_PARMINIT**

defines a subroutine that returns the initial values for the distribution’s parameters given an empirical distribution function (EDF) estimate.

- **Type**: Subroutine
- **Required**: NO
- **Number of arguments**: \( m + 4 \), where \( m \) is the number of distribution parameters
- **Sequence and type of arguments**:
  - \( \text{dim} \): Input numeric value that contains the dimension of the \( x \), \( nx \), and \( F \) array arguments.
  - \( x[*] \): Input numeric array of dimension \( \text{dim} \) that contains values of the random variables at which the EDF estimate is available. It can be assumed that \( x \) contains values in an increasing order. In other words, if \( i < j \), then \( x[i] < x[j] \).
  - \( nx[*] \): Input numeric array of dimension \( \text{dim} \). Each \( nx[i] \) contains the number of observations in the original data that have the value \( x[i] \).
  - \( F[*] \): Input numeric array of dimension \( \text{dim} \). Each \( F[i] \) contains the EDF estimate for \( x[i] \). This estimate is computed by the SEVSELECT procedure based on the options that you specify in the LOSS statement and the \text{EMPIRICALCDF=} \) option.
  - \( F\text{type} \): Input numeric value that contains the type of the EDF estimate that is stored in \( x \) and \( F \). For definitions of types, see the section “Supplying EDF Estimates to Functions and Subroutines” on page 266.
  - \( p1 \): Output argument that returns the initial value of the first parameter. You must specify this in the OUTARGS statement inside the subroutine’s definition.
  - \( p2 \): Output argument that returns the initial value of the second parameter. You must specify this in the OUTARGS statement inside the subroutine’s definition.
  - \( \ldots \)
  - \( pm \): Output argument that returns the initial value of the \( m \)th parameter. You must specify this in the OUTARGS statement inside the subroutine’s definition.

- **Return value**: The results, initial values of the parameters, should be returned in the parameter arguments of the subroutine.

Here is a sample structure of the subroutine for a distribution named ‘FOO’:
Chapter 6: The SEVSELECT Procedure

```c
subroutine FOO_PARMINIT(dim, x(*), nx(*), F(*), Ftype, p1, p2);
  outargs p1, p2;

  /* Code to initialize values of P1 and P2 by using 
     dim, x, nx, and F */
  p1 = <initial value for p1>;
  p2 = <initial value for p2>;
endsub;
```

dist_PDF

defines a function that returns the value of the probability density function (PDF) of the distribution at the specified values of the random variable and distribution parameters.

- **Type**: Function
- **Required**: YES
- **Number of arguments**: \(m + 1\), where \(m\) is the number of distribution parameters
- **Sequence and type of arguments**:
  - \(x\) Numeric value of the random variable at which the PDF value should be evaluated
  - \(p1\) Numeric value of the first parameter
  - \(p2\) Numeric value of the second parameter
  - \(\ldots\)
  - \(pm\) Numeric value of the \(m\)th parameter
- **Return value**: Numeric value that contains the PDF value \(f(x; p_1, p_2, \ldots, p_m)\)

If you want to consider this distribution as a candidate distribution when you estimate a response variable model with regression effects, then the first parameter of this distribution must be a scale parameter or log-transformed scale parameter. In other words, if the distribution has a scale parameter, then the following equation must be satisfied:

\[
f(x; p_1, p_2, \ldots, p_m) = \frac{1}{p_1} f\left(\frac{x}{p_1}; 1, p_2, \ldots, p_m\right)
\]

If the distribution has a log-transformed scale parameter, then the following equation must be satisfied:

\[
f(x; p_1, p_2, \ldots, p_m) = \frac{1}{\exp(p_1)} f\left(\frac{x}{\exp(p_1)}; 0, p_2, \ldots, p_m\right)
\]

Here is a sample structure of the function for a distribution named ‘FOO’:

```c
function FOO_PDF(x, P1, P2);
  /* Code to compute PDF by using x, P1, and P2 */
  f = <computed PDF>;
  return (f);
endsub;
```
Dist_QUANTILE defines a function that returns the quantile of the distribution at the specified value of the CDF for the specified values of distribution parameters.

- **Type:** Function
- **Required:** NO
- **Number of arguments:** \( m + 1 \), where \( m \) is the number of distribution parameters
- **Sequence and type of arguments:**
  - \( \text{cdf} \)  Numeric value of the cumulative distribution function (CDF) for which the quantile should be evaluated
  - \( p_1 \)  Numeric value of the first parameter
  - \( p_2 \)  Numeric value of the second parameter
  - ... 
  - \( p_m \)  Numeric value of the \( m \)th parameter
- **Return value:** Numeric value that contains the quantile \( F^{-1}(\text{cdf}; p_1, p_2, \ldots, p_m) \)

Here is a sample structure of the function for a distribution named ‘FOO’:

```plaintext
function FOO_QUANTILE(c, P1, P2);
    /* Code to compute quantile by using c, P1, and P2 */
    Q = <computed quantile>;
    return (Q);
endsub;
```

dist_SCALETRANSFORM defines a function that returns a keyword to identify the transform that needs to be applied to the scale parameter to convert it to the first parameter of the distribution.

If you want to use this distribution for modeling regression effects, then the first parameter of this distribution must be a scale parameter. However, for some distributions, a typical or convenient parameterization might not have a scale parameter, but one of the parameters can be a simple transform of the scale parameter. As an example, consider a typical parameterization of the lognormal distribution with two parameters, location \( \mu \) and shape \( \sigma \), for which the PDF is defined as follows:

\[
 f(x; \mu, \sigma) = \frac{1}{x\sigma \sqrt{2\pi}} e^{-\frac{1}{2} \left( \frac{\log(x)-\mu}{\sigma} \right)^2}
\]

You can reparameterize this distribution to contain a parameter \( \theta \) instead of the parameter \( \mu \) such that \( \mu = \log(\theta) \). The parameter \( \theta \) would then be a scale parameter. However, if you want to specify the distribution in terms of \( \mu \) and \( \sigma \) (which is a more recognized form of the lognormal distribution) and still allow it as a candidate distribution for estimating regression effects, then instead of writing another distribution with parameters \( \theta \) and \( \sigma \), you can simply define the distribution with \( \mu \) as the first parameter and specify that it is the logarithm of the scale parameter.
Chapter 6: The SEVSELECT Procedure

- **Type**: Function
- **Required**: NO
- **Number of arguments**: None
- **Sequence and type of arguments**: Not applicable
- **Return value**: Character value that contains one of the following keywords:
  - **LOG**: specifies that the first parameter is the logarithm of the scale parameter.
  - **IDENTITY**: specifies that the first parameter is a scale parameter without any transformation.

If you do not specify this function, then the IDENTITY transform is assumed.

Here is a sample structure of the function for a distribution named ‘FOO’:

```plaintext
function FOO_SCALETRANSFORM() $8;
    length xform $8;
    xform = "IDENTITY";
    return (xform);
endsub;
```

**dist_SDF**

defines a function that returns the value of the survival distribution function (SDF) of the distribution at the specified values of the random variable and distribution parameters.

- **Type**: Function
- **Required**: NO
- **Number of arguments**: $m \cdot 1$, where $m$ is the number of distribution parameters
- **Sequence and type of arguments**:
  - $x$ Numeric value of the random variable at which the SDF value should be evaluated
  - $p_1$ Numeric value of the first parameter
  - $p_2$ Numeric value of the second parameter
  - ...
  - $p_m$ Numeric value of the $m$th parameter
- **Return value**: Numeric value that contains the SDF value $S(x; p_1, p_2, \ldots, p_m)$

If you want to consider this distribution as a candidate distribution when estimating a response variable model with regression effects, then the first parameter of this distribution must be a scale parameter or log-transformed scale parameter. In other words, if the distribution has a scale parameter, then the following equation must be satisfied:

$$S(x; p_1, p_2, \ldots, p_m) = S\left(\frac{x}{p_1}; 1, p_2, \ldots, p_m\right)$$

If the distribution has a log-transformed scale parameter, then the following equation must be satisfied:

$$S(x; p_1, p_2, \ldots, p_m) = S\left(\frac{x}{\exp(p_1)}; 0, p_2, \ldots, p_m\right)$$

Here is a sample structure of the function for a distribution named ‘FOO’:
function FOO_SDF(x, P1, P2);
    /* Code to compute SDF by using x, P1, and P2 */
    S = <computed SDF>;
    return (S);
endsub;

dist_UPPERBOUNDS
defines a subroutine that returns upper bounds for the parameters of the distribution. If this subroutine
is not defined for a particular distribution, then the SEVSELECT procedure assumes that there is no
upper bound for any of the parameters. If an upper bound of \( u_i \) is returned for a parameter \( p_i \), then
the SEVSELECT procedure assumes that \( p_i < u_i \) (strict inequality). If a missing value is returned
for some parameter, then the SEVSELECT procedure assumes that there is no upper bound for that
parameter (equivalent to an upper bound of \( \infty \)).

- **Type**: Subroutine
- **Required**: NO
- **Number of arguments**: \( m \), where \( m \) is the number of distribution parameters
- **Sequence and type of arguments**:
  
  - \( p_1 \) Output argument that returns the upper bound on the first parameter. You must
    specify this in the OUTARGS statement inside the subroutine’s definition.
  - \( p_2 \) Output argument that returns the upper bound on the second parameter. You must
    specify this in the OUTARGS statement inside the subroutine’s definition.
  - \( \ldots \)
  - \( p_m \) Output argument that returns the upper bound on the \( m \)th parameter. You must
    specify this in the OUTARGS statement inside the subroutine’s definition.

- **Return value**: The results, upper bounds on parameter values, should be returned in the parameter
arguments of the subroutine.

Here is a sample structure of the subroutine for a distribution named ‘FOO’:

```
subroutine FOO_UPPERBOUNDS(p1, p2);
    outargs p1, p2;

    p1 = <upper bound for P1>;
    p2 = <upper bound for P2>;
endsub;
```

dist_coreGRADIENT
defines a subroutine that returns the gradient vector of the specified core function of the distribution at
the specified values of the random variable and distribution parameters. The core keyword can be PDF,
CDF, SDF, LOGPDF, LOGCDF, or LOGSDF.

- **Type**: Subroutine
- **Required**: NO
Chapter 6: The SEVSELECT Procedure

- **Number of arguments:** \( m + 2 \), where \( m \) is the number of distribution parameters

- **Sequence and type of arguments:**

  \( x \)  
  Numeric value of the random variable at which the gradient should be evaluated

  \( p_1 \)  
  Numeric value of the first parameter

  \( p_2 \)  
  Numeric value of the second parameter

  \[ \ldots \]

  \( p_m \)  
  Numeric value of the \( m \)th parameter

  \( \text{grad}[*] \)  
  Output numeric array of size \( m \) that contains the gradient vector evaluated at the specified values. If \( h \) denotes the value of the core function, then the expected order of the values in the array is as follows:

  \[
  \frac{\partial h}{\partial p_1} \quad \frac{\partial h}{\partial p_2} \quad \ldots \quad \frac{\partial h}{\partial p_m}
  \]

- **Return value:** Numeric array that contains the gradient evaluated at \( x \) for the parameter values \((p_1, p_2, \ldots, p_m)\)

Here is a sample structure of the function for the core function CDF of a distribution named ‘FOO’:

```plaintext
subroutine FOO_CDFGRADIENT(x, P1, P2, grad(*));
  outargs grad;
  /* Code to compute gradient by using x, P1, and P2 */
  grad[1] = <partial derivative of CDF w.r.t. P1 evaluated at x, P1, P2>;
  grad[2] = <partial derivative of CDF w.r.t. P2 evaluated at x, P1, P2>;
endsub;
```

**dist_coreHESSIAN**

defines a subroutine that returns the Hessian matrix of the specified core function of the distribution at the specified values of the random variable and distribution parameters. The core keyword can be PDF, CDF, SDF, LOGPDF, LOGCDF, or LOGSDF.

- **Type:** Subroutine

- **Required:** NO

- **Number of arguments:** \( m + 2 \), where \( m \) is the number of distribution parameters

- **Sequence and type of arguments:**

  \( x \)  
  Numeric value of the random variable at which the Hessian matrix should be evaluated

  \( p_1 \)  
  Numeric value of the first parameter

  \( p_2 \)  
  Numeric value of the second parameter

  \[ \ldots \]

  \( p_m \)  
  Numeric value of the \( m \)th parameter

  \( \text{hess}[*] \)  
  Output numeric array of size \( m(m + 1)/2 \) that contains the lower triangular portion of the Hessian matrix in a packed vector form, evaluated at the specified values. If \( h \) denotes the value of the core function, then the expected order of the values in the array is as follows:

  \[
  \frac{\partial^2 h}{\partial p_1^2} \quad \frac{\partial^2 h}{\partial p_1 \partial p_2} \quad \frac{\partial^2 h}{\partial p_1 \partial p_3} \quad \ldots \quad \frac{\partial^2 h}{\partial p_1 \partial p_m} \quad \frac{\partial^2 h}{\partial p_2 \partial p_3} \quad \ldots \quad \frac{\partial^2 h}{\partial p_m^2}
  \]
• **Return value:** Numeric array that contains the lower triangular portion of the Hessian matrix evaluated at \( x \) for the parameter values \((p_1, p_2, \ldots, p_m)\)

Here is a sample structure of the subroutine for the core function LOGSDF of a distribution named ‘FOO’:

```plaintext
subroutine FOO_LOGSDFHESSIAN(x, P1, P2, hess{*});
  outargs hess;

  /* Code to compute Hessian by using x, P1, and P2 */
  hess[1] = <second order partial derivative of LOGSDF w.r.t. P1 evaluated at x, P1, P2>;
  hess[2] = <second order partial derivative of LOGSDF w.r.t. P1 and P2 evaluated at x, P1, P2>;
  hess[3] = <second order partial derivative of LOGSDF w.r.t. P2 evaluated at x, P1, P2>;
endsub;
```

---

**Predefined Utility Functions**

The following predefined utility functions are provided with the SEVSELECT procedure and are available in the Sashelp.Svrtdist library:

**SVRTUTIL_EDF**

This function computes the empirical distribution function (EDF) estimate at the specified value of the random variable given the EDF estimate for a sample.

- **Type:** Function
- **Signature:** SVRTUTIL_EDF(y, n, x{*}, F{*}, Ftype)
- **Argument description:**
  
<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>Value of the random variable at which the EDF estimate is desired</td>
</tr>
<tr>
<td>n</td>
<td>Dimension of the ( x ) and ( F ) input arrays</td>
</tr>
<tr>
<td>x{*}</td>
<td>Input numeric array of dimension ( n ) that contains values of the random variable observed in the sample. These values are sorted in nondecreasing order.</td>
</tr>
<tr>
<td>F{*}</td>
<td>Input numeric array of dimension ( n ) in which each ( F[i] ) contains the EDF estimate for ( x[i] ). These values must be sorted in nondecreasing order.</td>
</tr>
<tr>
<td>Ftype</td>
<td>Type of the empirical estimate that is stored in the ( x ) and ( F ) arrays. For definitions of types, see the section “Supplying EDF Estimates to Functions and Subroutines” on page 266.</td>
</tr>
</tbody>
</table>

- **Return value:** The EDF estimate at \( y \)

The type of the sample EDF estimate determines how the EDF estimate at \( y \) is computed. For more information, see the section “Supplying EDF Estimates to Functions and Subroutines” on page 266.
SVRTUTIL_EMPLIMMOMENT

This function computes the empirical estimate of the limited moment of specified order for the specified upper limit, given the EDF estimate for a sample.

- **Type**: Function
- **Signature**: SVRTUTIL_EMPLIMMOMENT(k, u, n, x{*}, F{*}, Ftype)
- **Argument description**:
  - k: Order of the desired empirical limited moment
  - u: Upper limit on the value of the random variable to be used in the computation of the desired empirical limited moment
  - n: Dimension of the x and F input arrays
  - x{*}: Input numeric array of dimension n that contains values of the random variable observed in the sample. These values are sorted in nondecreasing order.
  - F{*}: Input numeric array of dimension n in which each F[i] contains the EDF estimate for x[i]. These values must be sorted in nondecreasing order.
  - Ftype: Type of the empirical estimate that is stored in the x and F arrays. For definitions of types, see the section “Supplying EDF Estimates to Functions and Subroutines” on page 266.
- **Return value**: The desired empirical limited moment

The empirical limited moment is computed by using the empirical estimate of the CDF. If \( F_n(x) \) denotes the EDF at x, then the empirical limited moment of order k with upper limit u is defined as

\[
E_n[(X \wedge u)^k] = k \int_0^u (1 - F_n(x))x^{k-1}dx
\]

The SVRTUTIL_EMPLIMMOMENT function uses the piecewise linear nature of \( F_n(x) \) as described in the section “Supplying EDF Estimates to Functions and Subroutines” on page 266 to compute the integration.

SVRTUTIL_HILLCUTOFF

This function computes an estimate of the value where the right tail of a distribution is expected to begin. The function implements the algorithm described in Danielsson et al. 2001. The description of the algorithm uses the following notation:

- \( n \): Number of observations in the original sample
- \( B \): Number of bootstrap samples to draw
- \( m_1 \): Size of the bootstrap sample in the first step of the algorithm (\( m_1 < n \))
- \( x_{(i)}^{j,m} \): ith-order statistic of jth bootstrap sample of size m (\( 1 \leq i \leq m, 1 \leq j \leq B \))
- \( x_{(i)} \): ith-order statistic of the original sample (\( 1 \leq i \leq n \))

Given the input sample x and values of B and \( m_1 \), the steps of the algorithm are as follows:

1. Take B bootstrap samples of size \( m_1 \) from the original sample.
2. Find the integer \( k_1 \) that minimizes the bootstrap estimate of the mean squared error:
\[
k_1 = \arg \min_{1 \leq k < m_1} Q(m_1, k)
\]

3. Take \( B \) bootstrap samples of size \( m_2 = m_1^2/n \) from the original sample.

4. Find the integer \( k_2 \) that minimizes the bootstrap estimate of the mean squared error:
\[
k_2 = \arg \min_{1 \leq k < m_2} Q(m_2, k)
\]

5. Compute the integer \( k_{opt} \), which is used for computing the cutoff point:
\[
k_{opt} = \frac{k_2^2}{k_1} \left( \frac{\log(k_1)}{2 \log(m_1) - \log(k_1)} \right)^{2-2 \log(k_1)/\log(m_1)}
\]

6. Set the cutoff point equal to \( x_{k_{opt}+1} \).

The bootstrap estimate of the mean squared error is computed as
\[
Q(m, k) = \frac{1}{B} \sum_{j=1}^{B} \text{MSE}_j(m, k)
\]

The mean squared error of \( j \)th bootstrap sample is computed as
\[
\text{MSE}_j(m, k) = \left( M_j(m, k) - 2(\gamma_j(m, k))^2 \right)^2
\]
where \( M_j(m, k) \) is a control variate proposed by Danielsson et al. 2001,
\[
M_j(m, k) = \frac{1}{k} \sum_{i=1}^{k} \left( \log(x_{j, m, m}^{i, m})(m-i+1) - \log(x_{j, m, m}^{i, m}(m-k)) \right)^2
\]
and \( \gamma_j(m, k) \) is the Hill’s estimator of the tail index (Hill 1975),
\[
\gamma_j(m, k) = \frac{1}{k} \sum_{i=1}^{k} \log(x_{j, m, m}^{i, m}(m-i+1)) - \log(x_{j, m, m}^{i, m}(m-k))
\]

This algorithm has two tuning parameters, \( B \) and \( m_1 \). The number of bootstrap samples \( B \) is chosen based on the availability of computational resources. The optimal value of \( m_1 \) is chosen such that the following ratio, \( R(m_1) \), is minimized:
\[
R(m_1) = \frac{(Q(m_1, k_1))^2}{Q(m_2, k_2)}
\]

The SVRTUTIL_HILLCUTOFF utility function implements the preceding algorithm. It uses the grid search method to compute the optimal value of \( m_1 \).

- **Type:** Function
- **Signature:** SVRTUTIL_HILLCUTOFF(n, x{*}, b, s, status)
- **Argument description:**
n  Dimension of the array x
x{*}  Input numeric array of dimension n that contains the sample
b  Number of bootstrap samples used to estimate the mean squared error. If b is less than 10, then a default value of 50 is used.
s  Approximate number of steps used to search the optimal value of \( m_1 \) in the range \([n^{0.75}, n-1]\). If s is less than or equal to 1, then a default value of 10 is used.
status  Output argument that contains the status of the algorithm. If the algorithm succeeds in computing a valid cutoff point, then status is set to 0. If the algorithm fails, then status is set to 1.

- **Return value**: The cutoff value where the right tail is estimated to start. If the size of the input sample is inadequate \((n \leq 5)\), then a missing value is returned and status is set to a missing value. If the algorithm fails to estimate a valid cutoff value \((status = 1)\), then the fifth-largest value in the input sample is returned.

SVRTUTIL_PERCENTILE

This function computes the specified empirical percentile given the EDF estimates.

- **Type**: Function
- **Signature**: SVRTUTIL_PERCENTILE(p, n, x{*}, F{*}, Ftype)
- **Argument description**:
  - p  Desired percentile. The value must be in the interval \((0,1)\). The function returns the 100\(p\)th percentile.
  - n  Dimension of the \(x\) and \(F\) input arrays
  - x{*}  Input numeric array of dimension \(n\) that contains values of the random variable observed in the sample. These values are sorted in nondecreasing order.
  - F{*}  Input numeric array of dimension \(n\) in which each \(F[i]\) contains the EDF estimate for \(x[i]\). These values must be sorted in nondecreasing order.
  - Ftype  Type of the empirical estimate that is stored in the \(x\) and \(F\) arrays. For definitions of types, see the section “Supplying EDF Estimates to Functions and Subroutines” on page 266.
- **Return value**: The 100\(p\)th percentile of the input sample

The method used to compute the percentile depends on the type of the EDF estimate (Ftype argument).

- **Ftype = 1**: Smoothed empirical estimates are computed using the method described in Klugman, Panjer, and Willmot (1998). Let \([x]\) denote the greatest integer less than or equal to \(x\). Define \(g = [p(n + 1)]\) and \(h = p(n + 1) - g\). Then the empirical percentile \(\hat{\pi}_p\) is defined as
  \[
  \hat{\pi}_p = (1 - h)x[g] + hx[g + 1]
  \]
  This method does not work if \(p < 1/(n + 1)\) or \(p > n/(n + 1)\). If \(p < 1/(n + 1)\), then the function returns \(\hat{\pi}_p = x[1]/2\), which assumes that the EDF is 0 in the interval \([0, x[1]]\). If \(p > n/(n + 1)\), then \(\hat{\pi}_p = x[n]\).
Predefined Utility Functions

Ftype = 2
If \( p < F[1] \), then \( \hat{\pi}_p = x[1]/2 \), which assumes that the EDF is 0 in the interval [0, \( x[1] \)]. If \( |p - F[i]| < \epsilon \) for some value of \( i \) and \( i < n \), then \( \hat{\pi}_p \) is computed as

\[
\hat{\pi}_p = \frac{x[i] + x[i + 1]}{2}
\]

where \( \epsilon \) is a machine-precision constant as returned by the SAS function CONSTANT('MACEPS'). If \( F[i - 1] < p < F[i] \), then \( \hat{\pi}_p \) is computed as

\[
\hat{\pi}_p = x[i]
\]

If \( p \geq F[n] \), then \( \hat{\pi}_p = x[n] \).

Ftype = 3
If \( p < F[1] \), then \( \hat{\pi}_p = x[1]/2 \), which assumes that the EDF is 0 in the interval [0, \( x[1] \)]. If \( |p - F[i]| < \epsilon \) for some value of \( i \) and \( i < n \), then \( \hat{\pi}_p \) is computed as

\[
\hat{\pi}_p = \frac{x[i] + x[i + 1]}{2}
\]

where \( \epsilon \) is a machine-precision constant as returned by the SAS function CONSTANT('MACEPS'). If \( F[i - 1] < p < F[i] \), then \( \hat{\pi}_p \) is computed as

\[
\hat{\pi}_p = x[i - 1] + (p - F[i - 1]) \frac{x[i] - x[i - 1]}{F[i] - F[i - 1]}
\]

If \( p \geq F[n] \), then \( \hat{\pi}_p = x[n] \).

SVRTUTIL_RAWMOMENTS
This subroutine computes the raw moments of a sample.

- **Type**: Subroutine
- **Signature**: SVRTUTIL_RAWMOMENTS(n, x{*}, nx{*}, nRaw, raw{*})
- **Argument description**:
  - n: Dimension of the x and nx input arrays
  - x{*}: Input numeric array of dimension n that contains distinct values of the random variable that are observed in the sample
  - nx{*}: Input numeric array of dimension n in which each nx[i] contains the number of observations in the sample that have the value x[i]
  - nRaw: Desired number of raw moments. The output array raw contains the first nRaw raw moments.
  - raw{*}: Output array of raw moments. The kth element in the array (raw{k}) contains the kth raw moment, where 1 \( \leq k \leq \) nRaw.
- **Return value**: Numeric array raw that contains the first nRaw raw moments. The array contains missing values if the sample has no observations (that is, if all the values in the nx array add up to zero).

SVRTUTIL_SORT
This function sorts an array of numeric values in an ascending or descending order.

- **Type**: Subroutine
- **Signature**: SVRTUTIL_SORT(n, x{*}, flag)
- **Argument description**:
Chapter 6: The SEVSELECT Procedure

n
Dimension of the input array x

x{*} Numeric array that contains the values to be sorted at input. The subroutine uses the same array to return the sorted values.

flag A numeric value that controls the sort order. If flag is 0, then the values are sorted in an ascending order. If flag has any value other than 0, then the values are sorted in descending order.

Return value: Numeric array x, which is sorted in place (that is, the sorted array is stored in the same storage area occupied by the input array x)

You can use the following predefined functions when you use the FCMP procedure to define functions and subroutines. They are summarized here for your information. For more information, see the FCMP procedure documentation in Base SAS Procedures Guide.

INVCDF
This function computes the quantile from any continuous probability distribution by numerically inverting the CDF of that distribution. You need to specify the CDF function of the distribution, the values of its parameters, and the cumulative probability to compute the quantile.

LIMMOMENT
This function computes the limited moment of order k with upper limit u for any continuous probability distribution. The limited moment is defined as

\[ E[(X \wedge u)^k] = \int_0^u x^k f(x)dx + \int_u^\infty u^k f(x)dx \]
\[ = \int_0^u x^k f(x)dx + u^k (1 - F(u)) \]

where \( f(x) \) and \( F(x) \) denote the PDF and the CDF of the distribution, respectively. The LIMMOMENT function uses the following alternate definition, which can be derived using integration-by-parts:

\[ E[(X \wedge u)^k] = k \int_0^u (1 - F(x))x^{k-1}dx \]

You need to specify the CDF function of the distribution, the values of its parameters, and the values of k and u to compute the limited moment.

Input Data Tables

PROC SEVSELECT accepts DATA= and INEST= data tables as input data tables. This section details the information that they are expected to contain.

DATA= Data Table

The DATA= data table is expected to contain the values of the analysis variables that you specify in the CLASS, EFFECT, LOSS, and SCALEMODEL statements.

If you specify the BY statement, then the DATA= data table must contain all the BY variables that you specify in the BY statement. However, it is not required to be sorted by the BY variables.
INEST\(\) Data Table

The INEST\(\) data table is expected to contain the initial values of the parameters for the parameter estimation process.

If you specify the BY statement, then the INEST\(\) data table must contain all the BY variables that you specify in the BY statement. The data table is not required to be sorted by BY variables, and it is not required to contain all the BY groups present in the DATA\(=\) data table. For the BY groups that are not present in the INEST\(=\) data table, the default parameter initialization method is used.

The INEST\(=\) data table must be created by using the OUTEST\(=\) option in a PROC SEVSELECT step that uses the same BY and SCALEMODEL statements as the PROC SEVSELECT step where you specify the INEST\(=\) option.

Output Data Tables

PROC SEVSELECT produces the OUTEST\(=\) data table when requested by the OUTEST\(=\) option in the PROC SEVSELECT statement. It also produces the OUT\(=\) data table when you specify the OUTPUT statement. The data tables and their contents are described in the following sections.

OUT\(=\) Data Table

The OUT\(=\) data table that you specify in the OUTPUT statement records the estimates of the scoring functions and quantiles that you specify in the OUTPUT statement.

For each distribution that you specify in the DIST statement, the OUT\(=\) data table contains one variable for each scoring function that you specify in the FUNCTIONS\(=\) option and one variable for each quantile that you specify in the QUANTILES\(=\) option. The prefix of the variable’s name is \(<\text{distribution-name}>_\)\(\), whereas the suffix of the variable’s name is determined by the information that you specify in the respective option or by the default method that PROC SEVSELECT uses. For more information about variable names, see the description of the OUTPUT statement.

The OUT\(=\) data table also contains the variables that you specify in the COPYVARS\(=\) option. If you specify the BY statement and if you want PROC SEVSELECT to copy the BY variables from the DATA\(=\) data table to the OUT\(=\) data table, then you must specify them in the COPYVARS\(=\) option.

The number of observations in the OUT\(=\) data table depends on the options that you specify in the OUTPUT statement and whether or not you specify the SCALEMODEL statement.

If either of the following conditions is met, then the number of observations in the OUT\(=\) data table is equal to the number of observations in the DATA\(=\) data table:

- You specify the SCALEMODEL statement.
- You specify the FUNCTIONS\(=\) option in the OUTPUT statement such that at least one scoring function does not have a constant, nonmissing argument.

If neither of the preceding conditions is met, then the number of observations in the OUT\(=\) data table is equal to the number of BY groups, which is equal to 1 if you do not specify the BY statement.
OUTEST= Data Table

The OUTEST= data table records the estimates of the model parameters. It also contains estimates of their standard errors and optionally their covariance structure. If you specify BY variables, then the data are organized in BY groups and the data table contains variables that you specify in the BY statement.

If you do not specify the COVOUT option, then the data table contains the following variables:

- **MODEL**: identifying name of the distribution model. The observation contains information about this distribution.
- **TYPE**: type of the estimates reported in this observation. It can take one of the following two values:
  - EST: point estimates of model parameters
  - STDERR: standard error estimates of model parameters
- **STATUS**: status of the reported estimates. The possible values are listed in the section “**STATUS** Variable Values” on page 294.

<Parameter 1> . . . <Parameter M>

*M* variables, named after the parameters of all candidate distributions, that contain estimates of the respective parameters. *M* is the cardinality of the union of parameter name sets from all candidate distributions. In an observation, estimates are populated only for parameters that correspond to the distribution that is indicated by the **MODEL** variable. If **TYPE** is EST, then the estimates are missing if the model does not converge. If **TYPE** is STDERR, then the estimates are missing if covariance estimates cannot be obtained.

If you specify regression effects, then the estimate that is reported for the first parameter of each distribution is the estimate of the base value of the scale or log-transformed scale parameter. For more information, see the section “Estimating Regression Effects” on page 256.

<Regression Parameter 1> . . . <Regression Parameter K>

If your effect specification in the SCALEMODEL statement results in *K* regression parameters and if you do not specify the SELECTOUT option, then the OUTEST= data table contains *K* regression parameter variables.

The name of each variable is formed by using the name of the effect and the levels of the CLASS variables that the effect might contain. If the effect name or level names are too long, then the variable name is constructed by using a part of the effect name and parts of the levels of the CLASS variables. The label of the variable is more descriptive than the name of the variable. The variables contain different types of estimates for their respective regression parameters.

If **TYPE** is EST, then a variable contains the estimate of the regression coefficient \( \hat{\beta}_j \) of the respective regression parameter, unless one of the following conditions applies:

- If the model does not converge, then it contains an ordinary missing value.
- If the regression parameter is linearly dependent on other regression parameters for a particular BY group and
  - if you specify the ZEROEST option, then it contains a value of 0.
if you do not specify the ZEROEST option, then it contains the special missing value .R.

- If you specify the SELECTION statement and if the regression parameter is not part of the selected model for a particular BY group, then
  - if you specify the ZEROEST option, then it contains a value of 0.
  - if you do not specify the ZEROEST option, then it contains the special missing value .N.

If _TYPE_ is STDERR, then a variable contains the estimate of the standard error of the respective regression parameter, unless one of the following conditions applies:

- If covariance estimates cannot be obtained, then it contains an ordinary missing value.
- If the regression parameter is linearly dependent on other regression parameters for a particular BY group, then it contains the special missing value .R.
- If you specify the SELECTION statement and if the regression parameter is not part of the selected model for a particular BY group, then it contains the special missing value .N.

<Offset Variable>

If you specify an OFFSET= variable in the SCALEMODEL statement, then the OUTEST= data table contains a variable that is named after the offset variable. If _TYPE_ is EST, then the value of this variable is 1. If _TYPE_ is STDERR, then the value of this variable is the special missing value .F.

If you specify the COVOUT option in the PROC SEVSELECT statement, then the OUTEST= data table contains additional observations that contain the estimates of the covariance structure. In addition to the variables listed and described previously, the data table contains the following variables that are either new or have a modified description:

_TYPE_ type of the estimates reported in this observation. For observations that contain rows of the covariance structure, the value is COV.

_STATUS_ status of the reported estimates. For observations that contain rows of the covariance structure, the status is 0 if covariance estimation was successful. If estimation fails, the status is 1 and a single observation is reported with _TYPE_=COV and missing values for all the parameter variables.

_NAME_ name of the parameter for the row of covariance matrix that is reported in the current observation.

If _TYPE_ is COV, then a variable contains the estimate of the covariance between the respective parameter and the parameter that is named in the _NAME_ variable, unless one of the following conditions applies:

- If covariance estimates cannot be obtained, then it contains an ordinary missing value.
- If the variable corresponds to a regression parameter and
  - if the regression parameter is linearly dependent on other regression parameters for a particular BY group, then it contains the special missing value .R.
– if you specify the SELECTION statement and if the regression parameter is not in the selected model for a particular BY group, then it contains the special missing value .N.

If you specify the SELECTION statement and the SELECTOUT option, then the OUTEST= data table contains only the regression parameters that correspond to the final selected effects. For the selected parameters, the values are written according to the rules stated previously for _TYPE_ values of EST, STDERR, and COV.

If you specify the BY and CLASS statements and if the names and values of some CLASS variables are long, then it is possible for a regression parameter with the same name to have different labels in different BY groups. In such cases, the OUTEST= data table contains rows with _TYPE_='COV' and an additional variable called _LABEL_. You can use the values of _NAME_ and _LABEL_ variables in such rows to obtain the label for multilabel regression parameters in each BY group. When the OUTEST= data table does not contain the _LABEL_ variable, then the label of a regression parameter is the label of the variable that contains the estimate of that regression parameter.

_STATUS_ Variable Values

The _STATUS_ variable in the OUTEST= data table contains a value that indicates the status of the parameter estimation process for the respective distribution model. The variable can take the following values in the OUTEST= data table for _TYPE_=EST observations:

- 0  The parameter estimation process converged for this model.
- 301 The parameter estimation process might not have converged for this model because there is no improvement in the objective function value. This might indicate that the initial values of the parameters are optimal, or you can try different convergence criteria in the NLOPTIONS statement.
- 302 The parameter estimation process might not have converged for this model because the number of iterations exceeded the maximum allowed value. You can try setting a larger value for the MAXITER= options in the NLOPTIONS statement.
- 303 The parameter estimation process might not have converged for this model because the number of objective function evaluations exceeded the maximum allowed value. You can try setting a larger value for the MAXFUNC= options in the NLOPTIONS statement.
- 304 The parameter estimation process might not have converged for this model because the time taken by the process exceeded the maximum allowed value. You can try setting a larger value for the MAXTIME= option in the NLOPTIONS statement.
- 400 The parameter estimation process did not converge for this model.

The _STATUS_ variable can take the following values in the OUTEST= data table for _TYPE_=STDERR and _TYPE_=COV observations:

- 0  The covariance and standard error estimates are available and valid.
- 1  The covariance and standard error estimates are not available, because the process of computing covariance estimates failed.
Displayed Output

The SEVSELECT procedure optionally produces displayed output by using the Output Delivery System (ODS). All output is controlled by the PRINT= option in the PROC SEVSELECT statement. Table 6.7 relates the ODS tables to PRINT= options.

Table 6.7  ODS Tables Produced in PROC SEVSELECT

<table>
<thead>
<tr>
<th>ODS Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>AllFitStatistics</td>
<td>Statistics of fit for all the distribution models</td>
<td>PRINT=ALLFITSTATS</td>
</tr>
<tr>
<td>BSplineDetails</td>
<td>Basis details for B-spline effects</td>
<td>PRINT=ALL and EFFECT SPLINE statement</td>
</tr>
<tr>
<td>ClassLevels</td>
<td>Level information for classification variables</td>
<td>Default for CLASS statement</td>
</tr>
<tr>
<td>CollectionLevelInfo</td>
<td>Level details for collection effects</td>
<td>PRINT=ALL and EFFECT COLLECTION statement</td>
</tr>
<tr>
<td>ConvergenceStatus</td>
<td>Convergence status of parameter estimation process</td>
<td>PRINT=CONVSTATUS</td>
</tr>
<tr>
<td>DescStats</td>
<td>Descriptive statistics for the response variable</td>
<td>PRINT=DESCSTATS</td>
</tr>
<tr>
<td>Distribution</td>
<td>Information about number of parameters in a distribution’s model</td>
<td>Default</td>
</tr>
<tr>
<td>DistributionInfo</td>
<td>Detailed information about all distributions</td>
<td>PRINT=DISTINFO</td>
</tr>
<tr>
<td>EntryCandidates</td>
<td>Details about candidates for entry into the model</td>
<td>PRINT=ALL and SELECTION statement</td>
</tr>
<tr>
<td>EstimationDetails</td>
<td>Details of the estimation process for all the distribution models</td>
<td>PRINT=ESTIMATIONDETAILS</td>
</tr>
<tr>
<td>FitStatistics</td>
<td>Statistics of fit</td>
<td>PRINT=STATISTICS</td>
</tr>
<tr>
<td>InitialValues</td>
<td>Initial parameter values and bounds</td>
<td>PRINT=INITIALVALUES</td>
</tr>
<tr>
<td>IterationHistory</td>
<td>Optimization iteration history</td>
<td>PRINT=NLOHISTORY</td>
</tr>
<tr>
<td>MMLevelInfo</td>
<td>Level information for multimember effects</td>
<td>PRINT=ALL and EFFECT MULTIMEMBER statement</td>
</tr>
<tr>
<td>ModelSelection</td>
<td>Model selection summary</td>
<td>PRINT=SELECTION</td>
</tr>
<tr>
<td>OptimizationSummary</td>
<td>Optimization summary</td>
<td>PRINT=NLOSUMMARY</td>
</tr>
<tr>
<td>ParameterEstimates</td>
<td>Final parameter estimates</td>
<td>PRINT=ESTIMATES</td>
</tr>
<tr>
<td>PolyDetails</td>
<td>Number of variables and columns, polynomial degree, and standardization method</td>
<td>PRINT=ALL and EFFECT POLY statement</td>
</tr>
<tr>
<td>PolyScaling</td>
<td>Centering and scaling details</td>
<td>PRINT=ALL and EFFECT POLY statement</td>
</tr>
<tr>
<td>RegDescStats</td>
<td>Descriptive statistics for the regression effects that do not contain a CLASS variable</td>
<td>PRINT=DESCSTATS</td>
</tr>
</tbody>
</table>
### Table 6.7 continued

<table>
<thead>
<tr>
<th>ODS Table Name</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>RemovalCandidates</td>
<td>Details about candidates for removal from the model</td>
<td>PRINT=ALL and SELECTION statement</td>
</tr>
<tr>
<td>SelectedEffects</td>
<td>List of effects selected for the model</td>
<td>Default for SELECTION statement</td>
</tr>
<tr>
<td>SelectionEstimationDetails</td>
<td>Details of the estimation process for all selection steps of a distribution’s model</td>
<td>PRINT=ESTIMATIONDETAILS and SELECTION statement</td>
</tr>
<tr>
<td>SelectionInfo</td>
<td>Information about the settings for model selection</td>
<td>Default for SELECTION statement</td>
</tr>
<tr>
<td>SelectionReason</td>
<td>Reason why the particular model was selected</td>
<td>Default for SELECTION statement</td>
</tr>
<tr>
<td>SelectionSummary</td>
<td>Summary information about model selection steps</td>
<td>Default for SELECTION statement</td>
</tr>
<tr>
<td>SplineKnots</td>
<td>Knot and boundary knot values</td>
<td>PRINT=ALL and EFFECT SPLINE statement</td>
</tr>
<tr>
<td>StopReason</td>
<td>Reason for termination of model selection</td>
<td>Default for SELECTION statement</td>
</tr>
<tr>
<td>TPFSplineDetails</td>
<td>Truncated power function (TPF) spline basis details</td>
<td>PRINT=ALL and EFFECT SPLINE statement</td>
</tr>
<tr>
<td>Timing</td>
<td>Timing information for various computational stages of the procedure</td>
<td>PRINT=ALL</td>
</tr>
</tbody>
</table>

If you do not specify the PRINT= option, then by default PROC SEVSELECT produces the ModelSelection, ConvergenceStatus, OptimizationSummary, FitStatistics, and ParameterEstimates ODS tables.

If you specify the CLASS statement and if you do not specify the NOCLPRINT option, then by default PROC SEVSELECT produces the ClassLevels table.

If you specify the SELECTION statement, then by default PROC SEVSELECT produces the SelectedEffects, SelectionInfo, SelectionReason, SelectionSummary, and StopReason ODS tables.

The following list presents further details of the contents of some of the tables:

**AllFitStatistics (PRINT=ALLFITSTATS)**

- displays the comparison of all the statistics of fit for all the models in one table. The table does not include the models whose parameter estimation process does not converge. If no models converge, then this table is not produced. If the table contains more than one model, then the best model according to each statistic is indicated by an asterisk (*) in that statistic’s column.

**DistributionInfo (PRINT=DISTINFO)**

- displays the information about all the candidate distribution. It includes the name, the description, the number of distribution parameters, and whether the distribution is valid for the specified modeling task.
FitStatistics (PRINT=STATISTICS)

displays the statistics of fit for each model. The statistics of fit are not displayed for models whose parameter estimation process does not converge.

ModelSelection (PRINT=SELECTION)

displays the model selection table. The table shows the convergence status of each candidate model and the value of the selection criterion along with an indication of the selected model.

ParameterEstimates (PRINT=ESTIMATES)

displays the estimates of parameters of the fitted model. The estimates are not displayed for models whose parameter estimation process does not converge.

---

**Examples: SEVSELECT Procedure**

**Example 6.1: Defining a Model for the Gaussian Distribution with a Scale Parameter**

If you want to estimate the influence of regression effects, then the model needs to be parameterized to have a scale parameter. Although this might not be always possible, it is possible for certain distributions. This example illustrates it for the Gaussian distribution. To obtain a parameterization with scale parameter, you replace the location parameter $\mu$ with another parameter, $\alpha = \mu / \sigma$, and define the PDF ($f$) and the CDF ($F$) as follows:

\[
  f(x; \sigma, \alpha) = \frac{1}{\sigma \sqrt{2\pi}} \exp\left( -\frac{1}{2} \left( \frac{x}{\sigma} - \alpha \right)^2 \right)
\]

\[
  F(x; \sigma, \alpha) = \frac{1}{2} \left( 1 + \text{erf} \left( \frac{1}{\sqrt{2}} \left( \frac{x}{\sigma} - \alpha \right) \right) \right)
\]

Then, you can verify that $\sigma$ is the scale parameter, because both of the following equalities are true:

\[
  f(x; \sigma, \alpha) = \frac{1}{\sigma} f\left( \frac{x}{\sigma}; 1, \alpha \right)
\]

\[
  F(x; \sigma, \alpha) = F\left( \frac{x}{\sigma}; 1, \alpha \right)
\]

**NOTE:** The Gaussian distribution is not a commonly used severity distribution. It is used in this example primarily to illustrate the concept of parameterizing a distribution such that it has a scale parameter. Although the distribution has a support over the entire real line, you can fit the distribution with PROC SEVSELECT only if the input sample contains nonnegative values.

The following statements use the alternate parameterization to define a new model named NORMAL_S. The definition is stored in the Work.Sevexmpl library.
An important point to note is that the scale parameter $\Sigma$ is the first distribution parameter (after the ‘x’ argument) listed in the signatures of NORMAL_S_PDF and NORMAL_S_CDF functions. $\Sigma$ is also the first distribution parameter listed in the signatures of other subroutines. This is required by PROC SEVSELECT, so that it can identify which is the scale parameter. When you specify regression effects, PROC SEVSELECT checks whether the first parameter of each candidate distribution is a scale parameter (or a log-transformed scale parameter if dist_SCALETRANSFORM subroutine is defined for the distribution with LOG as the transform). If it is not, then an appropriate message is written the SAS log and that distribution is not fitted.

Let the following DATA step statements simulate a sample from the normal distribution where the parameter $\sigma$ is affected by the regressors as follows:

$$\sigma = \exp(1 + 0.5 \times X1 + 0.75 \times X3 - 2 \times X4 + X5)$$

The sample is simulated such that the regressor $X2$ is linearly dependent on regressors $X1$ and $X3$.

```sas
/*--- Simulate a normal distribution's sample affected by regressors ---*/ data testnorm_reg(keep=y x1-x5 Sigma);
array x(*) x1-x5;
array b(6) _TEMPORARY_ (1 0.5 . 0.75 -2 1);
call streaminit(34567);
```
Example 6.1: Defining a Model for the Gaussian Distribution with a Scale Parameter

label y='Normal Response Influenced by Regressors';

do n = 1 to 100;
    /* simulate regressors */
    do i = 1 to dim(x);
        x(i) = rand('UNIFORM');
    end;
    /* make x2 linearly dependent on x1 */
    x(2) = 5 * x(1);

    /* compute log of the scale parameter */
    logSigma = b(1);
    do i = 1 to dim(x);
        if (i ne 2) then
            logSigma = logSigma + b(i+1) * x(i);
    end;
    Sigma = exp(logSigma);
    y = rand('NORMAL', 25, Sigma);
    output;
end;
run;

The following DATA step loads the Work.Testnorm_reg data set into a data table in your CAS session that is associated with the mycas CAS engine libref. The DATA step assumes that your CAS engine libref is named mycas, but you can substitute any appropriately defined CAS engine libref.

data mycas.testnorm_reg;
    set testnorm_reg;
run;

The following statements use PROC SEVSELECT to fit the NORMAL_S distribution model along with some of the predefined distributions to the simulated sample:

    /*--- Set the search path for functions defined with PROC FCMP ---*/
    options cmplib=(work.sevexmpl);

    /*-------- Fit models with PROC SEVSELECT --------*/
    proc sevselect data=mycas.testnorm_reg print=all;
        loss y;
        scalemodel x1-x5;
        dist Normal_s burr logn pareto weibull;
    run;

The “Model Selection” table in Output 6.1.1 indicates that all the models, except the Burr distribution model, have converged. Also, only three models, Normal_s, Burr, and Weibull, seem to have a good fit for the data. The table that compares all the fit statistics indicates that Normal_s model is the best according to the likelihood-based statistics; however, the Burr model is the best according to the EDF-based statistics.
Chapter 6: The SEVSELECT Procedure

Output 6.1.1 Summary of Results for Fitting the Normal Distribution with Regressors

The SEVSELECT Procedure

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Converged</th>
<th>Likelihood</th>
<th>Selected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal_s</td>
<td>Yes</td>
<td>603.95786</td>
<td>Yes</td>
</tr>
<tr>
<td>Burr</td>
<td>Maybe</td>
<td>612.81684</td>
<td>No</td>
</tr>
<tr>
<td>Logn</td>
<td>Yes</td>
<td>749.20125</td>
<td>No</td>
</tr>
<tr>
<td>Pareto</td>
<td>Yes</td>
<td>841.07018</td>
<td>No</td>
</tr>
<tr>
<td>Weibull</td>
<td>Yes</td>
<td>612.77496</td>
<td>No</td>
</tr>
</tbody>
</table>

This prompts you to further evaluate why the model with Burr distribution has not converged. The initial values, convergence status, and the optimization summary for the Burr distribution are shown in Output 6.1.2. The initial values table indicates that the regressor $X_2$ is redundant, which is expected. More importantly, the convergence status indicates that it requires more than 50 iterations. PROC SEVSELECT enables you to change several settings of the optimizer by using the NLOPTIONS statement. In this case, you can increase the limit of 50 on the iterations, change the convergence criterion, or change the technique to something other than the default trust-region technique.

Output 6.1.2 Details of the Fitted Burr Distribution Model

The SEVSELECT Procedure

<table>
<thead>
<tr>
<th>Distribution</th>
<th>-2 Log Likelihood</th>
<th>AIC</th>
<th>AICC</th>
<th>SBC</th>
<th>KS</th>
<th>AD</th>
<th>CvM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal_s</td>
<td>603.95786*</td>
<td>615.95786*</td>
<td>616.86108*</td>
<td>631.58888*</td>
<td>1.10894</td>
<td>1.95971*</td>
<td>0.29270</td>
</tr>
<tr>
<td>Burr</td>
<td>612.81684</td>
<td>626.81684</td>
<td>628.03423</td>
<td>645.05303</td>
<td>1.10250*</td>
<td>2.07305</td>
<td>0.27370*</td>
</tr>
<tr>
<td>Logn</td>
<td>749.20125</td>
<td>761.20125</td>
<td>762.10448</td>
<td>776.83227</td>
<td>1.89200</td>
<td>6.14785</td>
<td>1.05923</td>
</tr>
<tr>
<td>Pareto</td>
<td>841.07018</td>
<td>853.07018</td>
<td>853.97341</td>
<td>868.70121</td>
<td>2.85767</td>
<td>10.67436</td>
<td>2.30709</td>
</tr>
<tr>
<td>Weibull</td>
<td>612.77496</td>
<td>624.77496</td>
<td>625.67819</td>
<td>640.40598</td>
<td>1.10274</td>
<td>2.07458</td>
<td>0.27388</td>
</tr>
</tbody>
</table>

Asterisk (*) denotes the best model in the column.

This prompts you to further evaluate why the model with Burr distribution has not converged. The initial values, convergence status, and the optimization summary for the Burr distribution are shown in Output 6.1.2. The initial values table indicates that the regressor $X_2$ is redundant, which is expected. More importantly, the convergence status indicates that it requires more than 50 iterations. PROC SEVSELECT enables you to change several settings of the optimizer by using the NLOPTIONS statement. In this case, you can increase the limit of 50 on the iterations, change the convergence criterion, or change the technique to something other than the default trust-region technique.

Output 6.1.2 Details of the Fitted Burr Distribution Model

The SEVSELECT Procedure

Burr Distribution

<table>
<thead>
<tr>
<th>Model Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distribution</td>
</tr>
<tr>
<td>Description</td>
</tr>
<tr>
<td>Distribution Parameters</td>
</tr>
<tr>
<td>Regression Parameters</td>
</tr>
</tbody>
</table>
Example 6.1: Defining a Model for the Gaussian Distribution with a Scale Parameter

The following PROC SEVSELECT step uses the NLOPTIONS statement to change the convergence criterion and the limits on the iterations and function evaluations, exclude the lognormal and Pareto distributions that have been confirmed previously to fit the data poorly, and exclude the redundant regressor $X_2$ from the model:

```sas
/*--- Refit and compare models with higher limit on iterations ---*/
proc sevselect data=mycas.testnorm_reg print=all;
  loss y;
  scalemodel x1 x3-x5;
  dist Normal_s burr weibull;
  nloptions absfconv=2.0e-5 maxiter=100 maxfunc=500;
run;
```

The results shown in Output 6.1.3 indicate that the Burr distribution has now converged and that the Burr and Weibull distributions have an almost identical fit for the data. The NORMAL_S distribution is still the best distribution according to the likelihood-based criteria.

```
Output 6.1.3  Summary of Results after Changing Maximum Number of Iterations

The SEVSELECT Procedure

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Converged</th>
<th>-2 Log Likelihood</th>
<th>Selected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal_s</td>
<td>Yes</td>
<td>603.95786</td>
<td>Yes</td>
</tr>
<tr>
<td>Burr</td>
<td>Yes</td>
<td>612.79276</td>
<td>No</td>
</tr>
<tr>
<td>Weibull</td>
<td>Yes</td>
<td>612.77496</td>
<td>No</td>
</tr>
</tbody>
</table>
```
Output 6.1.3 continued

<table>
<thead>
<tr>
<th>Distribution</th>
<th>-2 Log Likelihood</th>
<th>AIC</th>
<th>AICC</th>
<th>SBC</th>
<th>KS</th>
<th>AD</th>
<th>CvM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>603.95786*</td>
<td>615.95786*</td>
<td>616.86108*</td>
<td>631.58888*</td>
<td>1.10894</td>
<td>1.95971*</td>
<td>0.29270</td>
</tr>
<tr>
<td>Burr</td>
<td>612.79276</td>
<td>626.79276</td>
<td>628.01015</td>
<td>645.02895</td>
<td>1.10264*</td>
<td>2.07393</td>
<td>0.27380*</td>
</tr>
<tr>
<td>Weibull</td>
<td>612.77496</td>
<td>624.77496</td>
<td>625.67819</td>
<td>640.40596</td>
<td>1.10274</td>
<td>2.07458</td>
<td>0.27388</td>
</tr>
</tbody>
</table>

Asterisk (*) denotes the best model in the column.

Example 6.2: Defining a Model for Mixed-Tail Distributions

In some applications, a few severity values tend to be extreme as compared to the typical values. The extreme values represent the worst-case scenarios and cannot be discarded as outliers. Instead, their distribution must be modeled to prepare for their occurrences. In such cases, it is often useful to fit one distribution to the non-extreme values and another distribution to the extreme values. The mixed-tail distribution mixes two distributions: one for the body region, which contains the non-extreme values, and another for the tail region, which contains the extreme values. The tail distribution is usually a generalized Pareto distribution (GPD), because it is usually good for modeling the conditional excess severity above a threshold. The body distribution can be any distribution. The following definitions are used in describing a generic formulation of a mixed-tail distribution:

\[
g(x) \quad \text{PDF of the body distribution} \\
G(x) \quad \text{CDF of the body distribution} \\
h(x) \quad \text{PDF of the tail distribution} \\
H(x) \quad \text{CDF of the tail distribution} \\
\theta \quad \text{scale parameter for the body distribution} \\
\Omega \quad \text{set of nonscale parameters for the body distribution} \\
\xi \quad \text{shape parameter for the GPD tail distribution} \\
x_r \quad \text{normalized value of the response variable at which the tail starts} \\
p_n \quad \text{mixing probability}
\]

Given these notations, the PDF \( f(x) \) and the CDF \( F(x) \) of the mixed-tail distribution are defined as

\[
f(x) = \begin{cases} 
\frac{p_n}{G(x_r)} g(x) & \text{if } x \leq x_b \\
(1 - p_n) h(x - x_b) & \text{if } x > x_b 
\end{cases}
\]

\[
F(x) = \begin{cases} 
\frac{p_n}{G(x_r)} G(x) & \text{if } x \leq x_b \\
p_n + (1 - p_n) H(x - x_b) & \text{if } x > x_b 
\end{cases}
\]

where \( x_b = \theta x_r \) is the value of the response variable at which the tail starts.

These definitions indicate the following:

- The body distribution is conditional on \( X \leq x_b \), where \( X \) denotes the random response variable.
- The tail distribution is the generalized Pareto distribution of the \( (X - x_b) \) values.
Example 6.2: Defining a Model for Mixed-Tail Distributions

The probability that a response variable value belongs to the body is \( p_n \). Consequently the probability that the value belongs to the tail is \((1 - p_n)\).

The parameters of this distribution are \( \theta, \Omega, \xi, x_r, \) and \( p_n \). The scale of the GPD tail distribution \( \theta_t \) is computed as

\[
\theta_t = \frac{G(x_b; \theta, \Omega)}{g(x_b; \theta, \Omega)} \left( \frac{1 - p_n}{p_n} \right) = \frac{G(x_r; \theta = 1, \Omega)}{g(x_r; \theta = 1, \Omega)} \left( \frac{1 - p_n}{p_n} \right)
\]

The parameter \( x_r \) is usually initialized using a tail index estimation algorithm. One such algorithm is Hill’s algorithm (Danielsson et al. 2001), which is implemented by the predefined utility function SVRTUTIL_HILLCUTOFF available to you in the Sashelp.Svrdist library. The algorithm and the utility function are described in detail in the section “Predefined Utility Functions” on page 285. The function computes an estimate of \( x_b \), which can be used to compute an initial estimate of \( x_r \) as \( x_r = x_b / \hat{\theta} \), where \( \hat{\theta} \) is the estimate of the scale parameter of the body distribution.

The parameter \( p_n \) is usually determined by the domain expert based on the fraction of losses that are expected to belong to the tail.

The following SAS statements define the LOGNGPD distribution model for a mixed-tail distribution with the lognormal distribution as the body distribution and GPD as the tail distribution:

```sas
/*------- Define lognormal Body-GPD tail mixed distribution -------*/
proc fcmp library=sashelp.svrdist outlib=work.sevexmpl.models;
   function LOGNGPD_DESCRIPTION() $256;
      length desc $256;
      desc1 = "Lognormal Body-GPD Tail Distribution.";
      desc2 = " Mu, Sigma, Xi, and Xr are free parameters.";
      desc3 = " Pn is a constant parameter.";
      desc = desc1 || desc2 || desc3;
      return(desc);
   endsub;

   function LOGNGPD_SCALETRANSFORM() $3;
      length xform $3;
      xform = "LOG";
      return (xform);
   endsub;

   subroutine LOGNGPD_CONSTANTPARM(Pn);
   endsub;

   function LOGNGPD_PDF(x, Mu, Sigma, Xi, Xr, Pn);
      cutoff = exp(Mu) * Xr;
      p = CDF('LOGN', cutoff, Mu, Sigma);
      if (x < cutoff + constant('MACEPS')) then do;
         return (((Pn/p)*PDF('LOGN', x, Mu, Sigma)));
      end;
      else do;
         gpd_scale = p*((1-Pn)/Pn)/PDF('LOGN', cutoff, Mu, Sigma);
         h = (1+Xi*(x-cutoff)/gpd_scale)**(-1-(1/Xi))/gpd_scale;
         return (((1-Pn)*h));
   end;
end;
```
Chapter 6: The SEVSELECT Procedure

function LOGNGPD_CDF(x, Mu, Sigma, Xi, Xr, Pn);
cutoff = exp(Mu) * Xr;
p = CDF('LOGN', cutoff, Mu, Sigma);
if (x < cutoff + constant('MACEPS')) then do;
    return ((Pn/p)*CDF('LOGN', x, Mu, Sigma));
end;
else do;
    gpd_scale = p*((1-Pn)/Pn)/PDF('LOGN', cutoff, Mu, Sigma);
    H = 1 - (1 + Xi*((x-cutoff)/gpd_scale))**(-1/Xi);
    return (Pn + (1-Pn)*H);
end;
endsub;

subroutine LOGNGPD_PARMINIT(dim, x[*], nx[*], F[*], Ftype, Mu, Sigma, Xi, Xr, Pn);
outargs Mu, Sigma, Xi, Xr, Pn;
array xe[1] / nosymbols;
array nxe[1] / nosymbols;
eps = constant('MACEPS');

Pn = 0.8; /* Set mixing probability */
_status_ = .;
call streaminit(56789);
Xb = svrtutil_hillcutoff(dim, x, 100, 25, _status_);
if (missing(_status_) or _status_ = 1) then
    Xb = svrtutil_percentile(Pn, dim, x, F, Ftype);

/* Initialize lognormal parameters */
call logn_parminit(dim, x, nx, F, Ftype, Mu, Sigma);
if (not(missing(Mu))) then
    Xr = Xb/exp(Mu);
else
    Xr = .;

/* Prepare arrays for excess values */
i = 1;
do while (i <= dim and x[i] < Xb+eps);
    i = i + 1;
end;
dime = dim-i+1;
if (dime > 0) then do;
call dynamic_array(xe, dime);
call dynamic_array(nxe, dime);
j = 1;
do while(i <= dim);
    xe[j] = x[i] - Xb;
nxe[j] = nx[i];
    i = i + 1;
    j = j + 1;
end;
Example 6.2: Defining a Model for Mixed-Tail Distributions

*/ Initialize GPD's shape parameter using excess values */
call gpd_parminit(dime, xe, nxe, F, Ftype, theta_gpd, Xi);
end;
else do;
   Xi = .;
end;
endsub;

subroutine LOGNGPD_LOWERBOUNDS(Mu,Sigma,Xi,Xr,Pn);
outargs Mu,Sigma,Xi,Xr,Pn;
   Mu = .; /* Mu has no lower bound */
   Sigma = 0; /* Sigma > 0 */
   Xi = 0; /* Xi > 0 */
   Xr = 0; /* Xr > 0 */
endsub;
quit;

Note the following points about the LOGNGPD definition:

- In this example, the parameter $p_n$ is not estimated with the maximum likelihood method used by PROC SEVSELECT, so you need to specify it as a constant parameter by defining the dist_CONSTANTPARAM subroutine. The signature of the LOGNGPD_CONSTANTPARAM subroutine lists only the constant parameter $Pn$.

- The LOGNGPD_PARMINIT subroutine initializes the parameter $x_r$ by first using the SVRTUTIL_HILLCUTOFF utility function to compute an estimate of the cutoff point $\hat{x}_b$ and then computing $x_r = \hat{x}_b/e^{\hat{\theta}}$. If SVRTUTIL_HILLCUTOFF fails to compute a valid estimate, then the SVRTUTIL_PERCENTILE utility function is used to set $\hat{x}_b$ to the $p_n$th percentile of the data. The parameter $p_n$ is fixed to 0.8.

- The Sashelp.Svrtdist library is specified with the LIBRARY= option in the PROC FCMP statement to enable the LOGNGPD_PARMINIT subroutine to use the predefined utility functions (SVRTUTIL_HILLCUTOFF and SVRTUTIL_PERCENTILE) and parameter initialization subroutines (LOGN_PARMINIT and GPD_PARMINIT).

- The LOGNGPD_LOWERBOUNDS subroutine defines the lower bounds for all parameters. This subroutine is required because the parameter $Mu$ has a non-default lower bound. The bounds for $Sigma$ and $Xi$ must be specified. If they are not specified, they are returned as missing values, which PROC SEVSELECT interprets as having no lower bound. You do not need to specify any bounds for the constant parameter $Pn$, because it is not subject to optimization.

The following DATA step statements simulate a sample from a mixed-tail distribution with a lognormal body and GPD tail. The parameter $p_n$ is fixed to 0.8, the same value used in the LOGNGPD_PARMINIT subroutine defined previously.

```plaintext
/*----- Simulate a sample for the mixed-tail distribution -----*/
data testmixdist(keep=y label='Lognormal Body-GPD Tail Sample');
   call streaminit(45678);
   label y='Response Variable';
   N = 1000;
   Mu = 1.5;
```
Sigma = 0.25;
Xi = 0.7;
Pn = 0.8;

/* Generate data for the lognormal body */
Nbody = N*Pn;
do i=1 to Nbody;
   y = exp(Mu) * rand('LOGNORMAL') ** Sigma;
   output;
end;

/* Generate data for the GPD tail */
cutoff = quantile('LOGNORMAL', Pn, Mu, Sigma);
gpd_scale = (1-Pn) / pdf('LOGNORMAL', cutoff, Mu, Sigma);
do i=Nbody+1 to N;
   y = cutoff + ((1-rand('UNIFORM')) ** (-Xi) - 1) * gpd_scale / Xi;
   output;
end;
run;

The following DATA step loads the Work.Testmixdist data set into a data table in your CAS session that is associated with the mycas CAS engine libref. The DATA step assumes that your CAS engine libref is named mycas, but you can substitute any appropriately defined CAS engine libref.

data mycas.testmixdist;
set testmixdist;
run;

The following statements use PROC SEVSELECT to fit the LOGNGPD distribution model to the simulated sample. They also fit three other predefined distributions (BURR, LOGN, and GPD). The final parameter estimates are written to the mycas.Parmest data table.

/*--- Set the search path for functions defined with PROC FCMP ---*/
options cmplib=(work.sevexmpl);

/****** Fit LOGNGPD model with PROC SEVSELECT ******/
proc sevselect data=mycas.testmixdist print=all outest=mycas.parmest;
   loss y;
   dist logngpd burr logn gpd;
run;

Some of the results that PROC SEVSELECT produces are shown in Output 6.2.1 and Output 6.2.2. The “Model Selection” table in Output 6.2.1 indicates that all models converged. The “All Fit Statistics” table in Output 6.2.1 shows that the model with LOGNGPD distribution has the best fit according to all the fit statistics. The Burr distribution model is the closest contender to the LOGNGPD model, but the GPD distribution model fits the data very poorly.
Example 6.2: Defining a Model for Mixed-Tail Distributions

Output 6.2.1  Summary of Fitting Mixed-Tail Distribution

The SEVSELECT Procedure

<table>
<thead>
<tr>
<th>Model Selection</th>
<th>-2 Log</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distribution</td>
<td>Converged</td>
</tr>
<tr>
<td>Logngpd</td>
<td>Yes</td>
</tr>
<tr>
<td>Burr</td>
<td>Yes</td>
</tr>
<tr>
<td>Logn</td>
<td>Yes</td>
</tr>
<tr>
<td>Gpd</td>
<td>Yes</td>
</tr>
</tbody>
</table>

All Fit Statistics

<table>
<thead>
<tr>
<th>Distribution</th>
<th>-2 Log</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logngpd</td>
<td>3640*</td>
</tr>
<tr>
<td>Burr</td>
<td>3687</td>
</tr>
<tr>
<td>Logn</td>
<td>3862</td>
</tr>
<tr>
<td>Gpd</td>
<td>5344</td>
</tr>
</tbody>
</table>

Output 6.2.2  Detailed Results for the LOGNGPD Distribution

The detailed results for the LOGNGPD distribution are shown in Output 6.2.2. The initial values table shows the fixed value of the \( Pn \) parameter that the LOGNGPD_PARMINIT subroutine sets. The table uses the bounds columns to indicate that it is a constant parameter. The last table in the figure shows the final parameter estimates. The estimates of all free parameters are significantly different from 0. As expected, the final estimate of the constant parameter \( Pn \) has not changed from its initial value.

Convergence Status

Convergence criterion (GCONV=1E-8) satisfied.
The following SAS statements use the parameter estimates to compute the value where the tail region is estimated to start \((x_b = e^{\theta_t} \times_r)\) and the scale of the GPD tail distribution \((\theta_t = \frac{G(x_b)}{g(x_b)} \frac{(1-P_n)}{P_n})\):

```sas
/*-------- Compute tail cutoff and tail distribution's scale --------*/
data xb_thetat(keep=x_b theta_t);
  set mycas.parmest(where=(MODEL_='Logngpd' and _TYPE_='EST'));
  x_b = exp(Mu) * Xr;
  theta_t = (CDF('LOGN',x_b,Mu,Sigma)/PDF('LOGN',x_b,Mu,Sigma)) * ((1-Pn)/Pn);
run;
proc print data=xb_thetat noobs;
run;
```

### Output 6.2.3

Start of the Tail and Scale of the GPD Tail Distribution

<table>
<thead>
<tr>
<th>x_b</th>
<th>theta_t</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.01665</td>
<td>1.00545</td>
</tr>
</tbody>
</table>

The computed values of \(x_b\) and \(\theta_t\) are shown as \(x_b\) and \(\theta_t\) in Output 6.2.3. Equipped with this additional derived information, you can now interpret the results of fitting the mixed-tail distribution as follows:

- The tail starts at \(y \approx 6.02\). Optimizing the scale-normalized relative cutoff \((x_r)\) in addition to optimizing the scale of the body region \((\theta = e^{\mu})\) gives you more flexibility in optimizing the absolute cutoff \((x_b)\). If \(Xr\) is declared as a constant parameter, then \(x_b\) is optimized by virtue of optimizing the scale of the body region \((\theta = e^{\mu})\), and you must rely on Hill’s tail index estimator to yield an initial estimate of \(x_b\) that is close to an optimal estimate. By keeping \(Xr\) as a free parameter, you account for the possibility that Hill’s estimator can yield a suboptimal estimate.

- The values \(y \leq 6.02\) follow the lognormal distribution with parameters \(\mu \approx 1.614\) and \(\sigma \approx 0.317\). These parameter estimates are reasonably close to the parameters of the body distribution that is used for simulating the sample.
Example 6.3: Fitting a Scaled Tweedie Model with Regressors

The Tweedie distribution is often used in the insurance industry to explain the influence of regression effects on the distribution of losses. PROC SEVSELECT provides a predefined scaled Tweedie distribution (STWEEDIE) that enables you to model the influence of regression effects on the scale parameter. The scale regression model has its own advantages such as the ability to easily account for inflation effects. This example illustrates how that model can be used to evaluate the influence of regression effects on the mean of the Tweedie distribution, which is useful in problems such as rate-making and pure premium modeling.

Assume a Tweedie process, whose mean \( \mu \) is affected by \( k \) regression effects \( x_j, j = 1, \ldots, k \), as follows,

\[
\mu = \mu_0 \exp \left( \sum_{j=1}^{k} \beta_j x_j \right)
\]

where \( \mu_0 \) represents the base value of the mean (you can think of \( \mu_0 \) as \( \exp(\beta_0) \), where \( \beta_0 \) is the intercept). This model for the mean is identical to the popular generalized linear model for the mean with a logarithmic link function.

More interestingly, it parallels the model used by PROC SEVSELECT for the scale parameter \( \theta \),

\[
\theta = \theta_0 \exp \left( \sum_{j=1}^{k} \beta_j x_j \right)
\]

where \( \theta_0 \) represents the base value of the scale parameter. As described in the section “Tweedie Distributions” on page 243, for the parameter range \( p \in (1, 2) \), the mean of the Tweedie distribution is given by

\[
\mu = \theta \lambda^{\frac{2-p}{p-1}}
\]

where \( \lambda \) is the Poisson mean parameter of the scaled Tweedie distribution. This relationship enables you to use the scale regression model to infer the influence of regression effects on the mean of the distribution.

Let the data set Work.Test_Sevtw contain a sample generated from a Tweedie distribution with dispersion parameter \( \phi = 0.5 \), index parameter \( p = 1.75 \), and the mean parameter that is affected by three regression variables \( x1, x2, \) and \( x3 \) as follows:

\[
\mu = 5 \exp(0.25 x1 - x2 + 3 x3)
\]

Thus, the population values of regression parameters are \( \mu_0 = 5, \beta_1 = 0.25, \beta_2 = -1, \) and \( \beta_3 = 3. \)

You following DATA step generates a sample of the Tweedie distribution:
/*--- Simulate a Tweedie sample that is affected by regressors ---*/
%let samplesize=300;
data test_sevtw(keep=y x1-x3
   label='A Tweedie Sample Affected by Regressors');
array x{*} x1-x3;
array b{4} _TEMPORARY_ (5 0.25 -1 3);
call streaminit(45678);
label y='Response Influenced by Regressors';
Phi = 0.5;
P = 1.75;
do n = 1 to &samplesize;
   Mu = 0;
   do i = 1 to dim(x);
      x(i) = rand('UNIFORM');
      Mu = Mu + b(i+1) * x(i);
   end;
   Mu = b(1) * exp(Mu); /* b(1) is base value of mean */
   cdf = rand('UNIFORM');
   y = quantile('TWEEDIE', cdf, P, Mu, Phi);
   output;
end;
run;

The following DATA step loads the Work.Test_sevtw data set into a data table in your CAS session that is associated with the mycas CAS engine libref. The DATA step assumes that your CAS engine libref is named mycas, but you can substitute any appropriately defined CAS engine libref.

data mycas.test_sevtw;
   set test_sevtw;
run;

The following PROC SEVSELECT step uses the sample in mycas.Test_Sevtw data table to estimate the parameters of the scale regression model for the predefined scaled Tweedie distribution (STWEEDIE) with the dual quasi-Newton (QUANEW) optimization technique:

/*--- Fit the scale parameter version of the Tweedie distribution ---*/
proc sevselect data=mycas.test_sevtw outest=mycas.estw covout print=all;
   loss y;
   scalemodel x1-x3;
   dist stweedie;
   nloptions tech=quanew;
run;

The dual quasi-Newton technique is used because it requires only the first-order derivatives of the objective function, and it is harder to compute reasonably accurate estimates of the second-order derivatives of Tweedie distribution’s PDF with respect to the parameters.

Some of the key results that PROC SEVSELECT prepares are shown in Output 6.3.1 and Output 6.3.2. The model information and the convergence results are shown in Output 6.3.1.
Output 6.3.1  Convergence Results for the STWEEDIE Model with Regressors

**The SEVSELECT Procedure**

**Stweedie Distribution**

<table>
<thead>
<tr>
<th>Model Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distribution</td>
</tr>
<tr>
<td>Description</td>
</tr>
<tr>
<td>Distribution Parameters</td>
</tr>
<tr>
<td>Regression Parameters</td>
</tr>
</tbody>
</table>

**Convergence Status**

Convergence criterion (FCONV=2.220446E-16) satisfied.

**Optimization Summary**

<table>
<thead>
<tr>
<th>Optimization Technique</th>
<th>Dual Quasi-Newton</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterations</td>
<td>39</td>
</tr>
<tr>
<td>Function Calls</td>
<td>186</td>
</tr>
<tr>
<td>Log Likelihood</td>
<td>-1044.321462</td>
</tr>
</tbody>
</table>

The final parameter estimates of the STWEEDIE regression model are shown in Output 6.3.2. The estimate that is reported for the parameter \( \theta \) is the estimate of the base value \( \theta_0 \). The estimates of regression coefficients \( \beta_1, \beta_2, \) and \( \beta_3 \) are indicated by the rows of \( x_1, x_2, \) and \( x_3 \), respectively.

Output 6.3.2  Parameter Estimates for the STWEEDIE Model with Regressors

<table>
<thead>
<tr>
<th>Parameter Estimates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter</td>
</tr>
<tr>
<td>Theta</td>
</tr>
<tr>
<td>Lambda</td>
</tr>
<tr>
<td>P</td>
</tr>
<tr>
<td>x1</td>
</tr>
<tr>
<td>x2</td>
</tr>
<tr>
<td>x3</td>
</tr>
</tbody>
</table>

If your goal is to explain the influence of regression effects on the scale parameter, then the output displayed in Output 6.3.2 is sufficient. But, if you want to compute the influence of regression effects on the mean of the distribution, then you need to do some postprocessing. Using the relationship between \( \mu \) and \( \theta \), \( \mu \) can be written in terms of the parameters of the STWEEDIE model as

\[
\mu = \theta_0 \exp \left( \sum_{j=1}^{k} \beta_j x_j \right) \lambda^{\frac{2-p}{p-1}}
\]

This shows that the parameters \( \beta_j \) are identical for the mean and the scale model, and the base value \( \mu_0 \) of the mean model is

\[
\mu_0 = \theta_0 \lambda^{\frac{2-p}{p-1}}
\]
The estimate of $\mu_0$ and the standard error associated with it can be computed by using the property of the functions of maximum likelihood estimators (MLE). If $g(\Omega)$ represents a totally differentiable function of parameters $\Omega$, then the MLE of $g$ has an asymptotic normal distribution with mean $g(\hat{\Omega})$ and covariance $C = (\partial g/\partial \Omega)' \Sigma (\partial g)$, where $\hat{\Omega}$ is the MLE of $\Omega$, $\Sigma$ is the estimate of covariance matrix of $\Omega$, and $\partial g$ is the gradient vector of $g$ with respect to $\Omega$ evaluated at $\hat{\Omega}$. For $\mu_0$, the function is $g(\Omega) = \theta_0 \lambda (2 - p)/(p - 1)$. The gradient vector is

$$
\partial g = \begin{pmatrix}
\frac{\partial g}{\partial \theta_0} & \frac{\partial g}{\partial \lambda} & \frac{\partial g}{\partial p} & \frac{\partial g}{\partial \beta_1} & \cdots & \frac{\partial g}{\partial \beta_k}
\end{pmatrix}
= \begin{pmatrix}
\frac{\mu_0}{\theta_0} & \frac{\mu_0}{\lambda} & \frac{-\mu_0}{(p - 1)(2 - p)} & 0 & \cdots & 0
\end{pmatrix}
$$

You can write a DATA step that implements these computations by using the parameter and covariance estimates that the PROC SEVSELECT step prepares. The following statements show one such implementation:

```plaintext
/*--- Bring estimates data to the client ---*/
data estw;
  set mycas.estw;
run;

/*--- Compute the estimate, standard error, and p-value of the mean ---*/
data Mu0(keep=Parameter Estimate Stderr T probT);
  Parameter='Mu0';
  label Estimate='Estimate' Stderr='Standard;Error' T='t Value' probT='Approx;Pr > |t|';
  array cov{3,3} _temporary_;
  array parms{3} _temporary_;
  array parmrow{3} theta lambda p;
  set estw(where=(_MODEL_='Stweedie' and
     (_type_='COV' or _type_='EST'))) end=eof;
  n = _n_-1;
  if (n = 0) then do;
    do i=1 to 3;
      parms[i] = parmrow[i];
    end;
  end;
  else do;
    do i=1 to n;
      cov[n,i] = parmrow[i];
    end;
  end;
  if (eof or n = 3) then do;
    mu0 = parms[1]*parms[2]*(2-parms[3])/(parms[3]-1);
    Estimate = mu0;
    a = mu0/parms[1]; /* dMu0_T0 */
    b = mu0/parms[2]; /* dMu0_L */
    c = -mu0/((parms[3]-1)*(2-parms[3])); /* dMu0_P */
    varMu0 = a * (a*cov[1,1]+b*cov[2,1]+c*cov[3,1]) +
             b * (a*cov[2,1]+b*cov[2,2]+c*cov[3,2]) +
             c * (a*cov[3,1]+b*cov[3,2]+c*cov[3,3]);
  end;
```
\[
\text{Stderr} = \sqrt{\text{varMu0}};
\]
\[
\text{df} = \&\text{samplesize}-6;
\]
\[
T = \frac{\mu0}{\text{Stderr}};
\]
\[
\text{probT} = (1-\text{probt}(T, \text{df}))*2;
\]
\[
\text{output};
\]
\[
\text{stop};
\]
\[
\text{end};
\]
\[
\text{run};
\]

The estimates of \(\mu_0\) that the preceding DATA step prepares are shown in Output 6.3.3. These estimates and the estimates of \(\beta\) as shown in Output 6.3.2 are reasonably close (that is, within one or two standard errors) to the parameters of the population from which the sample in \text{mycas.Test_Sevtw} data table was drawn.

**Output 6.3.3** Estimate of the Base Value Mu0 of the Mean Parameter

| Parameter | Estimate | Standard Error | t Value | Approx Pr > |t| |
|-----------|----------|----------------|---------|-------------|---|
| Mu0       | 4.47299  | 0.42284        | 10.5785 | 0           |

Another outcome of using the scaled Tweedie distribution to model the influence of regression effects is that the regression effects also influence the variance \(V\) of the Tweedie distribution. The variance is related to the mean as \(V = \phi \mu^p\), where \(\phi\) is the dispersion parameter. Using the relationship between the parameters \text{TWEEDIE} and \text{STWEEDIE} distributions as described in the section “Tweedie Distributions” on page 243, the regression model for the dispersion parameter is

\[
\log(\phi) = (2 - p) \log(\mu) - \log(\lambda(2 - p))
= ((2 - p) \log(\mu_0) - \log(\lambda(2 - p))) + (2 - p) \sum_{j=1}^{k} \beta_j x_j
\]

Subsequently, the regression model for the variance is

\[
\log(V) = 2 \log(\mu) - \log(\lambda(2 - p))
= (2 \log(\mu_0) - \log(\lambda(2 - p))) + 2 \sum_{j=1}^{k} \beta_j x_j
\]

In summary, PROC SEVSELECT enables you to estimate regression effects on various parameters and statistics of the Tweedie model.

---

**Example 6.4: Fitting Distributions to Interval-Censored Data**

In some applications, the data available for modeling might not be exact. A commonly encountered scenario is the use of grouped data from an external agency, which for several reasons, including privacy, does not provide information about individual loss events. The losses are grouped into disjoint bins, and you know
only the range and number of values in each bin. Each group is essentially interval-censored, because you
know that a loss magnitude is in certain interval, but you do not know the exact magnitude. This example
illustrates how you can use PROC SEVSELECT to model such data.

The following DATA step generates sample grouped data for dental insurance claims, which is taken from
Klugman, Panjer, and Willmot (1998):

```sas
/* Grouped dental insurance claims data (Klugman, Panjer, and Willmot 1998) */
data gdental;
  input lowerbd upperbd count @@;
datalines;
  0 25 30 25 50 31 50 100 57 100 150 42 150 250 65 250 500 84
  500 1000 45 1000 1500 10 1500 2500 11 2500 4000 3;
run;
```

To analyze small data sets such as this, you might want to restrict the number of workers that are used by
PROC SEVSELECT by submitting the following statements:

```sas
cas mysess terminate;
cas mysess sessopts=(nworkers=1);
libname mycas cas sessref=mysess;
```

The first CAS statement terminates the current CAS session named `mysess`. It also clears the `mycas` libref
that was associated with the session. The second CAS statement creates another session that uses only
one worker of the server. The LIBNAME statement associates the `mycas` libref with the new session. For
more information about the CAS statement and the LIBNAME statement, see SAS Cloud Analytic Services:
Language Reference.

If you want to load a SAS data set into a data table on the CAS server, then it is recommended that you
restrict the number of workers before you load the data set. If you restrict the number of workers after you
load the data, then PROC SEVSELECT needs to redistribute the data table so that it resides on the restricted
number of workers; this can cause unnecessary deterioration of performance.

The following DATA step loads the `Work.Gdental` data set into a data table in your CAS session that is
associated with the `mycas` CAS engine libref. The DATA step assumes that your CAS engine libref is named
`mycas`, but you can substitute any appropriately defined CAS engine libref.

```sas
data mycas.gdental;
  set gdental;
run;
```

The following PROC SEVSELECT step fits all the predefined distributions to the data in the `mycas.Gdental`
data table:

```sas
/* Fit all predefined distributions */
proc sevselect data=mycas.gdental edf=turnbull print=all criterion=aicc;
  loss / rc=lowerbd lc=upperbd;
  weight count;
  dist _predef_
run;
```

The EDF= option in the PROC SEVSELECT statement specifies that the Turnbull’s method be used for EDF
estimation. The LOSS statement specifies the left and right boundaries of each group as the right-censoring
and left-censoring limits, respectively. The variable `count` records the number of losses in each group and
is specified in the WEIGHT statement. Note that no response variable is specified in the LOSS statement, which is allowed as long as each observation in the input data table is censored.

Some of the key results that PROC SEVSELECT produces are shown in Output 6.4.1. According to the “Model Selection” table in Output 6.4.1, all distribution models have converged. The “All Fit Statistics” table in Output 6.4.1 indicates that the exponential distribution (EXP) has the best fit for data according to a majority of the likelihood-based statistics and that the Burr distribution (BURR) has the best fit according to all the EDF-based statistics.

Output 6.4.1  Statistics of Fit for Interval-Censored Data

The SEVSELECT Procedure

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Model Selection Converged</th>
<th>AIC Selected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Burr</td>
<td>Yes</td>
<td>51.41112 No</td>
</tr>
<tr>
<td>Exp</td>
<td>Yes</td>
<td>44.64768 Yes</td>
</tr>
<tr>
<td>Gamma</td>
<td>Yes</td>
<td>47.63969 No</td>
</tr>
<tr>
<td>Igauss</td>
<td>Yes</td>
<td>48.05874 No</td>
</tr>
<tr>
<td>Logn</td>
<td>Yes</td>
<td>47.34027 No</td>
</tr>
<tr>
<td>Pareto</td>
<td>Yes</td>
<td>47.16908 No</td>
</tr>
<tr>
<td>Gpd</td>
<td>Yes</td>
<td>47.16908 No</td>
</tr>
<tr>
<td>Weibull</td>
<td>Yes</td>
<td>47.47700 No</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Distribution</th>
<th>-2 Log Likelihood</th>
<th>AIC</th>
<th>AICC</th>
<th>SBC</th>
<th>KS</th>
<th>AD</th>
<th>CvM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Burr</td>
<td>41.41112*</td>
<td>47.41112</td>
<td>51.41112</td>
<td>48.31888</td>
<td>0.19212*</td>
<td>0.03900*</td>
<td>0.00308*</td>
</tr>
<tr>
<td>Exp</td>
<td>42.14768</td>
<td>44.14768*</td>
<td>44.64768*</td>
<td>44.45026*</td>
<td>1.26425</td>
<td>3.75589</td>
<td>0.70532</td>
</tr>
<tr>
<td>Gamma</td>
<td>41.92541</td>
<td>45.92541*</td>
<td>47.63969</td>
<td>46.53058</td>
<td>0.84349</td>
<td>1.74191</td>
<td>0.28675</td>
</tr>
<tr>
<td>Igauss</td>
<td>42.34445</td>
<td>46.34445</td>
<td>48.05874</td>
<td>46.94962</td>
<td>1.76236</td>
<td>4.64961</td>
<td>0.96835</td>
</tr>
<tr>
<td>Logn</td>
<td>41.62598</td>
<td>45.62598</td>
<td>47.34027</td>
<td>46.23115</td>
<td>0.67652</td>
<td>0.71212</td>
<td>0.12592</td>
</tr>
<tr>
<td>Pareto</td>
<td>41.45480</td>
<td>45.45480</td>
<td>47.16908</td>
<td>46.05997</td>
<td>0.34270</td>
<td>0.27942</td>
<td>0.03434</td>
</tr>
<tr>
<td>Gpd</td>
<td>41.45480</td>
<td>45.45480</td>
<td>47.16908</td>
<td>46.05997</td>
<td>0.34270</td>
<td>0.27941</td>
<td>0.03434</td>
</tr>
<tr>
<td>Weibull</td>
<td>41.76272</td>
<td>45.76272</td>
<td>47.47700</td>
<td>46.36789</td>
<td>0.70017</td>
<td>1.24457</td>
<td>0.17843</td>
</tr>
</tbody>
</table>

Asterisk (*) denotes the best model in the column.

When the best distributions that are chosen by the likelihood-based and EDF-based statistics are different, you need to decide which fit statistic best represents your objective. In this example, if your objective is to minimize the distance between EDF and CDF values, then you should choose the Burr distribution. On the other hand, if your objective is to maximize the likelihood of the observed data while minimizing the model complexity, then you should choose the exponential distribution. Note that the exponential distribution has worse (lower) raw likelihood than the Burr distribution, but it has better AIC, AICC, and BIC statistics than the Burr distribution because the exponential distribution has only one parameter compared to the three parameters of the Burr distribution. Further, the small sample size of 10 helps accentuate the role of model complexity in the AIC, AICC, and BIC statistics. If the sample size would have been larger, the exponential distribution might not have won according to the likelihood-based statistics.
Chapter 6: The SEVSELECT Procedure

Example 6.5: Defining a Finite Mixture Model That Has a Scale Parameter

A finite mixture model is a stochastic model that postulates that the probability distribution of the data generation process is a mixture of a finite number of probability distributions. For example, when an insurance company analyzes loss data from multiple policies that are underwritten in different geographic regions, some regions might behave similarly, but the distribution that governs some regions might be different from the distribution that governs other regions. Further, it might not be known which regions behave similarly. Also, the larger amounts of losses might follow a different stochastic process from the stochastic process that governs the smaller amounts of losses. It helps to model all policies together in order to pool the data together and exploit any commonalities among the regions, and the use of a finite mixture model can help capture the differences in distributions across regions and ranges of loss amounts.

Formally, if $f_i$ and $F_i$ denote the PDF and CDF, respectively, of component distribution $i$ and $p_i$ represents the mixing probability that is associated with component $i$, then the PDF and CDF of the finite mixture of $K$ distribution components are

$$f(x; \Theta, p) = \sum_{i=1}^{K} p_i f_i(x; \Theta_i)$$

$$F(x; \Theta, p) = \sum_{i=1}^{K} p_i F_i(x; \Theta_i)$$

where $\Theta_i$ denotes the parameters of component distribution $i$ and $\Theta$ denotes the parameters of the mixture distribution, which is a union of all the $\Theta_i$ parameters. $p$ denotes the set of mixing probabilities. All mixing probabilities must add up to 1 ($\sum_{i=1}^{K} p_i = 1$).

You can define the finite mixture of a specific number of components and specific distributions for each of the components by defining the FCMP functions for the PDF and CDF. However, in general, it is not possible to fit a scale regression model by using any finite mixture distribution unless you take special care to ensure that the mixture distribution has a scale parameter. This example provides a formulation of a two-component finite mixture model that has a scale parameter.

To start with, each component distribution must have either a scale parameter or a log-transformed scale parameter. Let $\theta_1$ and $\theta_2$ denote the scale parameters of the first and second components, respectively. Let $p_1 = p$ be the mixing probability, which makes $p_2 = 1 - p$ by using the constraint on $p$. The PDF of the mixture of these two distributions can be written as

$$f(x; \theta_1, \theta_2, \Phi, p) = \frac{p}{\theta_1} f_1\left(\frac{x}{\theta_1}; \Phi_1\right) + \frac{1 - p}{\theta_2} f_2\left(\frac{x}{\theta_2}; \Phi_2\right)$$

where $\Phi_1$ and $\Phi_2$ denote the sets of nonscale parameters of the first and second components, respectively, and $\Phi$ denotes a union of $\Phi_1$ and $\Phi_2$. For the mixture to have the scale parameter $\theta$, the PDF must be of the form

$$f(x; \theta, \Phi', p) = \frac{1}{\theta} \left( pf_1\left(\frac{x}{\theta}; \Phi'_1\right) + (1 - p) f_2\left(\frac{x}{\theta}; \Phi'_2\right) \right)$$

where $\Phi'$, $\Phi'_1$, and $\Phi'_2$ denote the modified sets of nonscale parameters. One simple way to achieve this is to make $\theta_1 = \theta_2 = \theta$ and $\Phi' = \Phi$; that is, you simply equate the scale parameters of both components and keep the set of nonscale parameters unchanged. However, forcing the scale parameters to be equal in both components is restrictive, because the mixture cannot model potential differences in the scales of the
Example 6.5: Defining a Finite Mixture Model That Has a Scale Parameter

two components. A better approach is to tie the scale parameters of the two components by a ratio such that \( \theta_1 = \theta \) and \( \theta_2 = \rho \theta \). If the ratio parameter \( \rho \) is estimated along with the other parameters, then the mixture distribution becomes flexible enough to model the variations across the scale parameters of individual components.

To summarize, the PDF and CDF are of the following form for the two-component mixture that has a scale parameter:

\[
\begin{align*}
\text{pdf}(x; \theta, \rho, \Phi, p) &= \frac{1}{\theta} \left( pf_1\left(\frac{x}{\theta} ; \Phi_1\right) + (1 - p) f_2\left(\frac{x}{\theta} ; \rho, \Phi_2\right) \right) \\
\text{cdf}(x; \theta, \rho, \Phi, p) &= pF_1\left(\frac{x}{\theta} ; \Phi_1\right) + (1 - p) F_2\left(\frac{x}{\theta} ; \rho, \Phi_2\right)
\end{align*}
\]

This can be generalized to a mixture of \( K \) components by introducing the \( K - 1 \) ratio parameters \( \rho_i \) that relate the scale parameters of each of the \( K \) components to the scale parameter \( \theta \) of the mixture distribution as follows:

\[
\begin{align*}
\theta_1 &= \theta \\
\theta_i &= \rho_i \theta; \ i \in [2, K]
\end{align*}
\]

In order to illustrate this approach, define a mixture of two lognormal distributions by using the following PDF function:

\[
\begin{align*}
\text{pdf}(x; \mu, \sigma_1, p_2, \rho_2, \sigma_2) &= \frac{(1 - p_2)}{\sigma_1 x \sqrt{2\pi}} \exp\left(-\frac{(\log(x) - \mu)^2}{2\sigma_1^2}\right) + \\
& \quad \frac{p_2}{\sigma_2 x \sqrt{2\pi}} \exp\left(-\frac{(\log(x) - \mu - \log(p_2))^2}{2\sigma_2^2}\right)
\end{align*}
\]

You can verify that \( \mu \) serves as the log of the scale parameter \( \theta \) (\( \mu = \log(\theta) \)).

The following PROC FCMP steps encode this formulation in a distribution named SLOGNMIX2 for use with PROC SEVSELECT:

```sas
/*- Define mixture of 2 lognormal distributions with a log-scale parameter */
proc fcmp library=sashelp.svrtdist outlib=work.sevexmpl.models;
    function slognmix2_description() $128;
        return ("Mixture of two lognormals with a log-scale parameter Mu");
    endsub;

    function slognmix2_scaletransform() $8;
        return ("LOG");
    endsub;

    function slognmix2_pdf(x, Mu, Sigma1, p2, Rho2, Sigma2);
        Mu1 = Mu;
        Mu2 = Mu + log(Rho2);
        pdf1 = logn_pdf(x, Mu1, Sigma1);
        pdf2 = logn_pdf(x, Mu2, Sigma2);
        return ((1-p2)*pdf1 + p2*pdf2);
    endsub;
```
function slognmix2_cdf(x, Mu, Sigma1, p2, Rho2, Sigma2);
    Mu1 = Mu;
    Mu2 = Mu + log(Rho2);
    cdf1 = logn_cdf(x, Mu1, Sigma1);
    cdf2 = logn_cdf(x, Mu2, Sigma2);
    return ((1-p2)*cdf1 + p2*cdf2);
endsub;

subroutine slognmix2_parminit(dim, x[*], nx[*], F[*], Ftype, Mu, Sigma1, p2, Rho2, Sigma2);
    outargs Mu, Sigma1, p2, Rho2, Sigma2;
    array m[1] / nosymbols;
    p2 = 0.5;
    Rho2 = 0.5;
    median = svrtutil_percentile(0.5, dim, x, F, Ftype);
    Mu = log(2*median/1.5);
    call svrtutil_rawmoments(dim, x, nx, 1, m);
    lm1 = log(m[1]);

    /* Search Rho2 that makes log(sample mean) > Mu */
    do while (lm1 <= Mu and Rho2 < 1);
        Rho2 = Rho2 + 0.01;
        Mu = log(2*median/(1+Rho2));
    end;
    if (Rho2 >= 1) then
        /* If Mu cannot be decreased enough to make it less than log(sample mean), then revert to Rho2=0.5. That will set Sigma1 and possibly Sigma2 to missing. PROC SEVSELECT replaces missing initial values with 0.001. */
        Mu = log(2*median/1.5);
        Sigma1 = sqrt(2.0*(log(m[1])-Mu));
        Sigma2 = sqrt(2.0*(log(m[1])-Mu-log(Rho2)));
    ends subroutine sognmix2_parminit(dim, x[*], nx[*], F[*], Ftype, Mu, Sigma1, p2, Rho2, Sigma2);

subroutine sognmix2_upperbounds(Mu, Sigma1, p2, Rho2, Sigma2);
    outargs Mu, Sigma1, p2, Rho2, Sigma2;
    Mu = .; /* Mu has no upper bound */
    Sigma1 = .; /* Sigma1 has no upper bound */
    p2 = 1; /* p2 < 1 */
    Rho2 = 1; /* Rho2 < 1 */
    Sigma2 = .; /* Sigma2 has no upper bound */
endsub;

quit;
As shown in previous examples, an important aspect of defining a distribution for use with PROC SEVSELECT is the definition of the PARMINIT subroutine that initializes the parameters. For mixture distributions, in general, the parameter initialization is a nontrivial task. For a two-component mixture, some simplifying assumptions make the problem easier to handle. For the initialization of SLOGNMIX2, the initial values of \( p_2 \) and \( \rho_2 \) are fixed at 0.5, and the following two simplifying assumptions are made:

- The median of the mixture is the average of the medians of the two components:
  \[
  F^{-1}(0.5) = (\exp(\mu_1) + \exp(\mu_2))/2 = \exp(\mu)(1 + \rho_2)/2
  \]
  Solution of this equation yields the value of \( \mu \) in terms of \( \rho_2 \) and the sample median.

- Each component has the same mean, which implies the following:
  \[
  \exp(\mu + \sigma_1^2/2) = \exp(\mu + \log(\rho_2) + \sigma_2^2/2)
  \]
  If \( X_i \) represents the random variable of component distribution \( i \) and \( X \) represents the random variable of the mixture distribution, then the following equation holds for the raw moment of any order \( k \):
  \[
  E[X^k] = \sum_{i=1}^{K} p_i E[X_i^k]
  \]
  This, in conjunction with the assumption on component means, leads to the equations
  \[
  \log(m_1) = \mu + \frac{\sigma_1^2}{2}
  \]
  \[
  \log(m_1) = \mu + \log(\rho_2) + \frac{\sigma_2^2}{2}
  \]
  where \( m_1 \) denotes the first raw moment of the sample. Solving these equations leads to the following values of \( \sigma_1 \) and \( \sigma_2 \):
  \[
  \sigma_1^2 = 2(\log(m_1) - \mu)
  \]
  \[
  \sigma_2^2 = 2(\log(m_1) - \mu - \log(\rho_2))
  \]
  Note that \( \sigma_1 \) has a valid value only if \( \log(m_1) > \mu \). Among the many possible methods of ensuring this condition, the SLOGNMIX2_PARMINIT subroutine uses the method of doing a linear search over \( \rho_2 \).

Even when the preceding assumptions are not true for a particular problem, they produce reasonable initial values to help guide the nonlinear optimizer to an acceptable optimum if the mixture of two lognormal distributions is indeed a good fit for your input data. This is illustrated by the results of the following steps that fit the SLOGNMIX2 distribution to simulated data, which have different means for the two components (12.18 and 22.76, respectively), and the median of the sample (15.94) is not equal to the average of the medians of the two components (7.39 and 20.09, respectively):
Chapter 6: The SEVSELECT Procedure

```sas
/*-------- Simulate a lognormal mixture sample ----------*/
data testlognmix(keep=y);
call streaminit(12345);
Mu1 = 2;
Sigma1 = 1;
i = 0;
do j=1 to 2000;
y = exp(Mu1) * rand('LOGNORMAL')**Sigma1;
output;
end;
Mu2 = 3;
Sigma2 = 0.5;
do j=1 to 3000;
y = exp(Mu2) * rand('LOGNORMAL')**Sigma2;
output;
end;
run;

/*----- Load data into the CAS server -----*/
data mycas.testlognmix;
set testlognmix;
run;

/*-- Fit and compare scale regression models with 2-component --*/
/*-- lognormal mixture and the standard lognormal distribution --*/
options cmplib=(work.sevexmpl);
proc sevselect data=mycas.testlognmix print=all;
loss y;
dist slognmix2 logn;
run;
```

The comparison of the fit statistics of SLOGNMIX2 and LOGN, as shown in Output 6.5.1, confirms that the two-component mixture is certainly a better fit to these data than the single lognormal distribution.

Output 6.5.1  Comparison of Fitting One versus Two Lognormal Components to Mixture Data

<table>
<thead>
<tr>
<th>All Fit Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distribution</td>
</tr>
<tr>
<td>Slognmix2</td>
</tr>
<tr>
<td>Logn</td>
</tr>
</tbody>
</table>

Asterisk (*) denotes the best model in the column.

The detailed results for the SLOGNMIX2 distribution are shown in Output 6.5.2. According to the “Initial Parameter Values and Bounds” table, the initial value of $\rho_2$ is not 0.5, indicating that a linear search was conducted to ensure $\log(m_1) > \mu$. 

```sas
The SEVSELECT Procedure
```
Example 6.6: Scale Regression with Rich Regression Effects

Output 6.5.2  Detailed Estimation Results for the SLOGNMIX2 Distribution

The SEVSELECT Procedure

Slognmix2 Distribution

<table>
<thead>
<tr>
<th>Distribution Parameters</th>
<th>Initial Parameter Values and Bounds</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Parameter</td>
</tr>
<tr>
<td></td>
<td>Mu</td>
</tr>
<tr>
<td></td>
<td>Sigma1</td>
</tr>
<tr>
<td></td>
<td>P2</td>
</tr>
<tr>
<td></td>
<td>Rho2</td>
</tr>
<tr>
<td></td>
<td>Sigma2</td>
</tr>
</tbody>
</table>

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

Optimization Summary
- Optimization Technique: Trust Region
- Iterations: 9
- Function Calls: 24
- Log Likelihood: -19171.54807

Parameter Estimates

| Parameter | DF | Estimate | Standard Error | t Value | Approx Pr > |t|
|-----------|----|----------|----------------|---------|-------------|
| Mu        | 1  | 3.00922  | 0.01554        | 193.68  | <.0001      |
| Sigma1    | 1  | 0.49516  | 0.01451        | 34.13   | <.0001      |
| P2        | 1  | 0.40619  | 0.02600        | 15.62   | <.0001      |
| Rho2      | 1  | 0.37212  | 0.02038        | 18.26   | <.0001      |
| Sigma2    | 1  | 1.00019  | 0.02124        | 47.09   | <.0001      |

By using the relationship that \( \mu_2 = \mu + \log(\rho_2) \), you can see that the final parameter estimates are indeed close to the true parameter values that were used to simulate the input sample.

Example 6.6: Scale Regression with Rich Regression Effects

This example illustrates the use of regression effects that include CLASS variables and interaction effects.

Consider that you, as an actuary at an automobile insurance company, want to evaluate the effect of certain external factors on the distribution of the severity of the losses that your policyholders incur. Such analysis can help you determine the relative differences in premiums that you should charge to policyholders who have different characteristics. Assume that when you collect and record the information about each claim, you also collect and record some key characteristics of the policyholder and the vehicle that is involved in the
claim. This example focuses on the following five factors: type of car, safety rating of the car, gender of the
policyholder, education level of the policyholder, and annual household income of the policyholder (which
can be thought of as a proxy for the luxury level of the car). Let these regressors be recorded in the variables
CarType (1: sedan, 2: sport utility vehicle), CarSafety (scaled to be between 0 and 1, the safest being 1),
Gender (1: female, 2: male), Education (1: high school graduate, 2: college graduate, 3: advanced degree
holder), and Income (scaled by a factor of 1/100,000), respectively. Let the historical data about the severity
of each loss be recorded in the LossAmount variable of the mycas.Losses data table in the CAS libref. Let
the data table also contain two additional variables, Deductible and Limit, that record the deductible and
ground-up loss limit provisions, respectively, of the insurance policy that the policyholder has. The limit on
ground-up loss is usually derived from the payment limit that a typical insurance policy states. Deductible
serves as the left-truncation variable, and Limit serves as the right-censoring variable.

The following SAS statements simulate an example of the mycas.Losses data table:

```sas
proc format casfmtlib='myfmtlib';
  value genderFmt 1='Female'
                   2='Male';
run;

data losses(keep=gender carType education carSafety income
             lossAmount deductible limit);
  call streaminit(12345);
  array sx{8} _temporary_;
  array sbeta{9} _TEMPORARY_ (5 0.6 0.4 -0.75 -0.3 0.4 0.7 -0.5 -0.3);
  length carType $8 education $16;
  format gender genderFmt.;
  sigma = 0.5;
  do lossEventId=1 to 6000;
    /* Simulate policyholder and vehicle attributes */
    do i=1 to dim(sx);
      sx[i] = 0;
    end;
    if (rand('UNIFORM') < 0.5) then do;
      gender = 1; * female;
      sx[2] = 1;
    end;
    else do;
      gender = 2; * male;
    end;
    if (rand('UNIFORM') < 0.7) then do;
      carInt = 1;
      carType = 'Sedan';
    end;
    else do;
      carInt = 2;
      carType = 'SUV';
      sx[1] = 1;
    end;
    /* Simulate claim amounts */
    loss = 0;
    do j=1 to dim(sbeta);
      loss = loss + sbeta[j] * sx[j];
    end;
    loss = loss * sigma;
    lossAmount = max(loss, deductible);
    if (lossAmount <= limit) then do;
      /* Simulate claim amounts */
      /* Simulate claim amounts */
      /* Simulate claim amounts */
      /* Simulate claim amounts */
    end;
  end;
run;
```
The variables CarType, Education, and Gender each contain a known, finite set of discrete values. By specifying such variables as classification variables, you can separately identify the effect of each level of the variable on the severity distribution. For example, you might be interested in finding out how the magnitude of loss for a sport utility vehicle (SUV) differs from that for a sedan. This is an example of a main effect. You might also want to evaluate how the distribution of losses that are incurred by a policyholder with a college degree who drives a SUV differs from that of a policyholder with an advanced degree who drives...
Chapter 6: The SEVSELECT Procedure

A sedan. This is an example of an interaction effect. You can include various such types of effects in the scale regression model. For more information about the effect types, see the section “Specification and Parameterization of Model Effects” on page 33 in Chapter 1, “Shared Concepts.”

Analyzing such a rich set of regression effects can help you make more accurate predictions about the losses that a new applicant with certain characteristics might incur when he or she requests insurance for a specific vehicle, which can further help you with ratemaking decisions.

The following PROC SEVSELECT step fits the scale regression model with a lognormal distribution to data in the mycas.Losses data table, and stores the model and parameter estimate information in the mycas.Est data table on the CAS server:

```sas
/* Fit scale regression model with different types of regression effects */
proc sevselect data=mycas.losses outest=mycas.est print=all;
  loss lossAmount / lt=deductible rc=limit;
  class carType gender education;
  scalemodel carType gender carSafety income education*carType
    income*gender carSafety*income;
  dist logn;
run;
```

The SCALEMODEL statement in the preceding PROC SEVSELECT step includes two main effects (carType and gender), two singleton continuous effects (carSafety and income), one interaction effect (education*carType), one continuous-by-class effect (income*gender), and one polynomial continuous effect (carSafety*income).

When you specify a CLASS statement, it is recommended that you observe the “Class Level Information” table. For this example, the table is shown in Output 6.6.1. Note that if you specify BY-group processing, then the class level information might change from one BY group to the next, potentially resulting in a different parameterization for each BY group.

**Output 6.6.1** Class Level Information Table

<table>
<thead>
<tr>
<th>Class</th>
<th>Levels</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>carType</td>
<td>2</td>
<td>SUV Sedan</td>
</tr>
<tr>
<td>gender</td>
<td>2</td>
<td>Female Male</td>
</tr>
<tr>
<td>education</td>
<td>3</td>
<td>AdvancedDegree College High School</td>
</tr>
</tbody>
</table>

The regression modeling results for the lognormal distribution are shown in Output 6.6.2. The “Initial Parameter Values and Bounds” table is important especially because the preceding PROC SEVSELECT step uses the default GLM parameterization, which is a singular parameterization—that is, it results in some redundant parameters. As shown in the table, the redundant parameters correspond to the last level of each classification variable; this correspondence is a defining characteristic of a GLM parameterization. An alternative would be to use the reference parameterization by specifying the PARAM=REFERENCE option in the CLASS statement, which does not generate redundant parameters for effects that contain CLASS variables and enables you to specify a reference level for each CLASS variable.
Output 6.6.2 Initial Values for the Scale Regression Model with Class and Interaction Effects

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Initial Value</th>
<th>Lower Bound</th>
<th>Upper Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mu</td>
<td>4.85228</td>
<td>-709.78271</td>
<td>709.78271</td>
</tr>
<tr>
<td>Sigma</td>
<td>0.52348</td>
<td>1.05367E-8</td>
<td>Infy</td>
</tr>
<tr>
<td>carType SUV</td>
<td>0.54686</td>
<td>-709.78271</td>
<td>709.78271</td>
</tr>
<tr>
<td>carType Sedan</td>
<td>Redundant</td>
<td>-709.78271</td>
<td>709.78271</td>
</tr>
<tr>
<td>gender Female</td>
<td>0.34893</td>
<td>-709.78271</td>
<td>709.78271</td>
</tr>
<tr>
<td>gender Male</td>
<td>Redundant</td>
<td>-709.78271</td>
<td>709.78271</td>
</tr>
<tr>
<td>carSafety</td>
<td>-0.63504</td>
<td>-709.78271</td>
<td>709.78271</td>
</tr>
<tr>
<td>income</td>
<td>-0.24031</td>
<td>-709.78271</td>
<td>709.78271</td>
</tr>
<tr>
<td>carType SUV * education AdvancedDegree</td>
<td>0.32719</td>
<td>-709.78271</td>
<td>709.78271</td>
</tr>
<tr>
<td>carType SUV * education College</td>
<td>0.68899</td>
<td>-709.78271</td>
<td>709.78271</td>
</tr>
<tr>
<td>carType SUV * education High School</td>
<td>Redundant</td>
<td>-709.78271</td>
<td>709.78271</td>
</tr>
<tr>
<td>carType Sedan * education AdvancedDegree</td>
<td>-0.44650</td>
<td>-709.78271</td>
<td>709.78271</td>
</tr>
<tr>
<td>carType Sedan * education College</td>
<td>-0.26834</td>
<td>-709.78271</td>
<td>709.78271</td>
</tr>
<tr>
<td>carType Sedan * education High School</td>
<td>Redundant</td>
<td>-709.78271</td>
<td>709.78271</td>
</tr>
<tr>
<td>income * gender Female</td>
<td>0.00843</td>
<td>-709.78271</td>
<td>709.78271</td>
</tr>
<tr>
<td>income * gender Male</td>
<td>Redundant</td>
<td>-709.78271</td>
<td>709.78271</td>
</tr>
<tr>
<td>carSafety * income</td>
<td>-0.04744</td>
<td>-709.78271</td>
<td>709.78271</td>
</tr>
</tbody>
</table>

The convergence and optimization summary information in Output 6.6.3 indicates that the scale regression model for the lognormal distribution has converged with the default optimization technique in five iterations.

Output 6.6.3 Optimization Summary for the Scale Regression Model with Class and Interaction Effects

<table>
<thead>
<tr>
<th>Convergence Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convergence criterion (GCONV=1E-8) satisfied.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Optimization Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimization Technique</td>
</tr>
<tr>
<td>Iterations</td>
</tr>
<tr>
<td>Function Calls</td>
</tr>
<tr>
<td>Log Likelihood</td>
</tr>
</tbody>
</table>

The “Parameter Estimates” table in Output 6.6.4 shows the distribution parameter estimates and estimates for various regression effects. You can use the estimates for effects that contain CLASS variables to infer the relative influence of various CLASS variable levels. For example, on average, the magnitude of losses that are incurred by the female drivers is \( \exp(0.44145) \approx 1.56 \) times greater than that of male drivers, and an SUV driver with an advanced degree incurs a loss that is on average \( \exp(0.39393)/\exp(-0.35210) \approx 2.11 \) times greater than the loss that a college-educated sedan driver incurs. Neither the continuous-by-class effect income*gender nor the polynomial continuous effect carSafety*income is significant in this example.
If you want to update the model when new claims data arrive, then you can potentially speed up the estimation process by specifying the OUTEST= data table that is created by the preceding PROC SEVSELECT step as an INEST= data table in a new PROC SEVSELECT step. To illustrate, the following PROC SEVSELECT step refits the model on the same input data as the preceding PROC SEVSELECT, but it uses the mycas.Est data table that is created by that step as an INEST= data table:

```sas
/* Refit scale regression model on new data */
proc sevselect data=mycas.losses inest=mycas.est print=all;
  loss lossAmount / lt=deductible rc=limit;
  class carType gender education;
  scalemodel carType gender carSafety income education*carType income*gender carSafety*income;
  dist logn;
run;
```

Because the INEST= data table is used to initialize the distribution and regression parameters, the optimization occurs in very few iterations, as shown in Output 6.6.5.

**Output 6.6.5** Optimization Summary for Refitting the Scale Regression Model with the INEST= Option

### The SEVSELECT Procedure

#### Logn Distribution

<table>
<thead>
<tr>
<th>Convergence Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convergence criterion (ABSGCONV=0.00001) satisfied.</td>
</tr>
</tbody>
</table>
Example 6.7: Scale Regression Model Selection

This example extends “Example 6.6: Scale Regression with Rich Regression Effects” to illustrate the model selection features of the SEVSELECT procedure. There are two phases of model selection when you fit a scale regression model. First, you want to select an optimal set of regression effects for each distribution’s scale regression model. The SELECTION statement provides several methods to do that. Second, you want to identify the best severity distribution by comparing the final selected scale regression models for each distribution. PROC SEVSELECT displays tables to help you do that.

The following SEVSELECT step uses the same data table as in Example 6.6 to find the best set of scale regression effects for three severity distributions—lognormal (Logn), Burr, and Weibull:

```sas
proc sevselect data=mycas.losses outest=mycas.est print=all;
  loss lossAmount / lt=deductible rc=limit;
  class carType gender education;
  scalemodel carType gender carSafety income education*carType
    income*gender carSafety*income;
  selection;
  dist logn burr weibull;
  output out=mycas.score copyvars=(carType gender education carSafety income)
    functions=(mean) quantiles=(points=0.5 0.975 names=(median var));
run;
```

The SELECTION statement without any options uses the stepwise selection method by default. PROC SEVSELECT reports the settings of the selection process in the “Selection Information” table, as shown in Output 6.7.1.

```
Output 6.7.1  Selection Method Settings

The SEVSELECT Procedure

Logn Distribution

Selection Information

<table>
<thead>
<tr>
<th>Selection Method</th>
<th>Stepwise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Criterion</td>
<td>SBC</td>
</tr>
<tr>
<td>Stop Criterion</td>
<td>SBC</td>
</tr>
<tr>
<td>Effect Hierarchy</td>
<td>Enforced</td>
</tr>
<tr>
<td>Stop Horizon</td>
<td>3</td>
</tr>
</tbody>
</table>
```

For each severity distribution, PROC SEVSELECT reports the summary of the selection process, as shown in Output 6.7.2 for the lognormal distribution. The “Selection Summary” table shows the order in which regression effects enter or leave the model. In this case, no effect leaves the model after entering the model.
In the first step, PROC SEVSELECT fits seven models, each with an intercept and one of the seven eligible effects in the SCALEMODEL statement. The model that contains the intercept and carType*education effect has the lowest value of the Schwarz Bayesian criterion (SBC) statistic in the first step, and hence it enters the model. In the second step, PROC SEVSELECT compares the model that does not contain the existing effect, carType*education, with models that contain that effect and one of the remaining six effects. The model that contains the intercept, carType*education, and carSafety effects is the best in the second step according to the SBC statistic. The process of removing an effect from the model or adding an effect that is not in the model continues until the stop criterion does not improve or the stop horizon condition is met. As the results in Output 6.7.2 show, the optimal value of the SBC criterion is achieved in the fourth step. The final scale regression model for the lognormal distribution contains the following effects: intercept, gender, carSafety, carType*education, and income*gender.

**Output 6.7.2** Stepwise Selection Summary for the Lognormal Distribution

<table>
<thead>
<tr>
<th>Selection Summary</th>
<th>Number Effects In</th>
<th>SBC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 0 Intercept</td>
<td>1</td>
<td>35975.2456</td>
</tr>
<tr>
<td>Step 1 carType*education</td>
<td>2</td>
<td>34121.0008</td>
</tr>
<tr>
<td>Step 2 carSafety</td>
<td>3</td>
<td>33709.8419</td>
</tr>
<tr>
<td>Step 3 income*gender</td>
<td>4</td>
<td>33298.5871</td>
</tr>
<tr>
<td>Step 4 gender</td>
<td>5</td>
<td>33159.7904*</td>
</tr>
</tbody>
</table>

* Optimal Value Of Criterion

Stepwise selection stopped because adding or removing an effect does not improve the SBC criterion.

The model at step 4 is selected.

Selected Effects: intercept gender carSafety carType*education income*gender

The selection summary information for the Weibull distribution is shown in Output 6.7.3. It shows that the best model in the sixth step is obtained by removing the carSafety*income effect, which was added in the second step. The model in the sixth step also has the lowest value of the SBC criterion; hence, it is the best model and contains the following effects: intercept, gender, carSafety, income, and carType*education.
Example 6.7: Scale Regression Model Selection

Output 6.7.3  Stepwise Selection Summary for the Weibull Distribution

The SEVSELECT Procedure

Weibull Distribution
Selection Details

<table>
<thead>
<tr>
<th>Step Entered</th>
<th>Effect Entered</th>
<th>Effect Removed</th>
<th>Number Effects In</th>
<th>SBC</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Intercept</td>
<td></td>
<td>1</td>
<td>36381.5042</td>
</tr>
<tr>
<td>1</td>
<td>carType*education</td>
<td></td>
<td>2</td>
<td>34477.5474</td>
</tr>
<tr>
<td>2</td>
<td>carSafety*income</td>
<td></td>
<td>3</td>
<td>34092.5186</td>
</tr>
<tr>
<td>3</td>
<td>gender</td>
<td></td>
<td>4</td>
<td>33684.7598</td>
</tr>
<tr>
<td>4</td>
<td>carSafety</td>
<td></td>
<td>5</td>
<td>33573.1854</td>
</tr>
<tr>
<td>5</td>
<td>income</td>
<td></td>
<td>6</td>
<td>33517.6460</td>
</tr>
<tr>
<td>6</td>
<td>carSafety*income</td>
<td></td>
<td>5</td>
<td>33509.5021*</td>
</tr>
</tbody>
</table>

* Optimal Value Of Criterion

Stepwise selection stopped because adding or removing an effect does not improve the SBC criterion.

The model at step 6 is selected.

Selected Effects: Intercept gender carSafety income carType*education

PROC SEVSELECT reports the optimization summary, fit statistics, and parameter estimates for the final selected model of each severity distribution. The results for the lognormal distribution are shown in Output 6.7.4. The “Parameter Estimates” table shows that all regression parameters in the final model are statistically significant. Comparison of these parameter estimates with those in Output 6.6.4, which were obtained without using the SELECTION statement, shows that the statistically insignificant effect, carSafety*income, is not present in the final selected model. The comparison also shows that the stepwise selection process prefers income*gender effect over the income effect and, because of the exclusion of the income effect, the income*gender effect has become statistically significant.

Output 6.7.4  Selected Scale Regression Model for the Lognormal Distribution

Convergence Status

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

Optimization Summary

<table>
<thead>
<tr>
<th>Optimization Technique</th>
<th>Trust Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterations</td>
<td>5</td>
</tr>
<tr>
<td>Function Calls</td>
<td>14</td>
</tr>
<tr>
<td>Log Likelihood</td>
<td>-16533.05056</td>
</tr>
</tbody>
</table>
Chapter 6: The SEVSELECT Procedure

Output 6.7.4 continued

<table>
<thead>
<tr>
<th>Fit Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2 Log Likelihood</td>
</tr>
<tr>
<td>Akaike's Information Criterion</td>
</tr>
<tr>
<td>Corrected Akaike's Information Criterion</td>
</tr>
<tr>
<td>Schwarz’s Bayesian Information Criterion</td>
</tr>
<tr>
<td>Kolmogorov-Smirnov Statistic</td>
</tr>
<tr>
<td>Anderson-Darling Statistic</td>
</tr>
<tr>
<td>Cramer-von Mises Statistic</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter Estimates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter</td>
</tr>
<tr>
<td>Mu</td>
</tr>
<tr>
<td>Sigma</td>
</tr>
<tr>
<td>gender Female</td>
</tr>
<tr>
<td>gender Male</td>
</tr>
<tr>
<td>carSafety</td>
</tr>
<tr>
<td>carType SUV * education AdvancedDegree</td>
</tr>
<tr>
<td>carType SUV * education College</td>
</tr>
<tr>
<td>carType SUV * education High School</td>
</tr>
<tr>
<td>carType Sedan * education AdvancedDegree</td>
</tr>
<tr>
<td>carType Sedan * education College</td>
</tr>
<tr>
<td>carType Sedan * education High School</td>
</tr>
<tr>
<td>income * gender Female</td>
</tr>
<tr>
<td>income * gender Male</td>
</tr>
</tbody>
</table>

After the effect selection process chooses the best scale regression model for each severity distribution, the next phase of model selection is to compare the final models across all severity distributions and identify the best severity distribution. You can do that by using the “All Fit Statistics” table, which helps you compare the final selected models according to all the fit statistics. For this example, Output 6.7.5 compares the final models of the three severity distributions—lognormal, Burr, and Weibull.

Output 6.7.5 Comparison of the Selected Scale Regression Models for Different Severity Distributions

<table>
<thead>
<tr>
<th>Parameter Estimates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter</td>
</tr>
<tr>
<td>Mu</td>
</tr>
<tr>
<td>Sigma</td>
</tr>
<tr>
<td>gender Female</td>
</tr>
<tr>
<td>gender Male</td>
</tr>
<tr>
<td>carSafety</td>
</tr>
<tr>
<td>carType SUV * education AdvancedDegree</td>
</tr>
<tr>
<td>carType SUV * education College</td>
</tr>
<tr>
<td>carType SUV * education High School</td>
</tr>
<tr>
<td>carType Sedan * education AdvancedDegree</td>
</tr>
<tr>
<td>carType Sedan * education College</td>
</tr>
<tr>
<td>carType Sedan * education High School</td>
</tr>
<tr>
<td>income * gender Female</td>
</tr>
<tr>
<td>income * gender Male</td>
</tr>
</tbody>
</table>

After the effect selection process chooses the best scale regression model for each severity distribution, the next phase of model selection is to compare the final models across all severity distributions and identify the best severity distribution. You can do that by using the “All Fit Statistics” table, which helps you compare the final selected models according to all the fit statistics. For this example, Output 6.7.5 compares the final models of the three severity distributions—lognormal, Burr, and Weibull.

Output 6.7.5 Comparison of the Selected Scale Regression Models for Different Severity Distributions

| Output 6.7.5 Comparison of the Selected Scale Regression Models for Different Severity Distributions

<table>
<thead>
<tr>
<th>All Fit Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distribution</td>
</tr>
<tr>
<td>Logn</td>
</tr>
<tr>
<td>Burr</td>
</tr>
<tr>
<td>Weibull</td>
</tr>
</tbody>
</table>

Asterisk (\(^*\)) denotes the best model in the column.

The lognormal distribution’s final selected model is the best according to all the likelihood-based statistics, the Burr distribution’s final selected model is the best according to the Anderson-Darling (AD) statistic, and the Weibull distribution’s final selected model is the best according to the Kolmogorov-Smirnov (KS) and Cramér-von Mises (CvM) statistics. You can choose the fit statistic that is most suitable for your application and use it to identify the best severity distribution. Alternatively, if your goal is to estimate some loss distribution statistic, such as the value-at-risk (VaR), for a particular scenario, then instead of choosing
one severity distribution, you might want to compute and compare the desired statistic for the final scale regression model of each severity distribution that wins according to at least one fit statistic. The OUTPUT statement is useful for such evaluation, because it helps you score each observation of the input data table that you specify in the DATA= option. In the context of PROC SEVSELECT, scoring an observation is equivalent to computing particular statistics for the severity distribution of the losses. The previous PROC SEVSELECT step specifies the following OUTPUT statement to write the scores for each observation to the mycas.Score output table:

```plaintext
output out=mycas.score copyvars=(carType gender education carSafety income) functions=(mean) quantiles=(cdf=0.5 0.975 names=(median var));
```

When you specify the scale regression model, the scale parameter of each severity distribution depends on the values of the regression effects in the final selected model of that distribution. For each observation in the input table, PROC SEVSELECT computes the estimate of the scale parameter and uses it to evaluate the scoring and quantile functions that you specify in the FUNCTIONS= and QUANTILES= options, respectively. The OUTPUT statement of this example specifies that three statistics be computed for the final model of each severity distribution: the mean scoring function and two quantile functions, \( Q_{\text{dist}}(0.5, \hat{\theta}) \) and \( Q_{\text{dist}}(0.975, \hat{\theta}) \), where \( Q_{\text{dist}}(p, \theta) \) denotes the value of the quantile function for a distribution \( \text{dist} \) that is evaluated at the CDF value of \( p \) for the estimated distribution parameters \( \hat{\theta} \). The variable that contains the estimate of the mean function is named mean. The variables that contain the quantile function estimate for \( p=0.5 \) and \( p=0.975 \) are named median and var, respectively; var denotes the value-at-risk (VaR). In order to compute a function \( F \) in the FUNCTIONS= option, for each severity distribution \( \text{dist} \), the \( \text{dist}_F \) function must be available in the function library search path that you specify in the CMPLIB= system option. Also, in order to compute the quantile functions faster, it is recommended that the \( \text{dist}_\text{QUANTILE} \) function be available in the CMPLIB= search path. If PROC SEVSELECT does not find a quantile function’s definition, it computes the quantile function by numerically inverting the cumulative distribution function (CDF). For each predefined distribution, such as the lognormal, Burr, and Weibull distributions of this example, the \( \text{dist}_\text{MEAN} \) and \( \text{dist}_\text{QUANTILE} \) functions are already defined and available to you in the function library, Sashelp.Svrtdist, that PROC SEVSELECT includes by default.

Output 6.7.6, Output 6.7.7, and Output 6.7.8 compare the estimates of the mean, median, and value-at-risk (var), respectively, of the three severity distributions for the first 10 observations that are fetched from the mycas.Score table. Note: The estimates are all available in the same table but are shown here separately for better comparison among different severity distributions. These results illustrate the following:

- The OUT= data table contains the variables that you specify in the COPYVARS= option. This helps you associate the scoring statistics with the values of the regression effects that decide the severity distribution’s scale parameter.

- The estimates of a particular statistic vary by observation, because the scale parameter value depends on the values of the regression effects. You can interpret the estimate of the mean as the average loss that a particular policyholder will incur if he or she has the characteristics that are quantified in the values of the regression effects. For example, a male policyholder with a college degree who earns 0.70732 on a normalized scale and drives a sport utility vehicle (SUV) that has a normalized car safety rating of 0.8446 is expected to incur an average loss of 298 units if you use the lognormal distribution’s selected scale regression model for estimation, or an average loss of 283 units if you use the Weibull distribution’s selected scale regression model for estimation.

- The estimates of the median, which is a robust estimate of the central tendency, vary less across different severity distributions. So you might want to use the median loss to predict the average loss that a policyholder will incur.
The estimates of the value-at-risk (VaR), which depend on the heaviness of the tail of the distribution, vary significantly across severity distributions. For this example, the fitted Burr and Weibull distributions have the heaviest and lightest tails, respectively, among the three severity distributions. The VaR statistic is typically used to assess the worst-case loss that a policyholder might incur. You can interpret it as follows: for a female policyholder with an advanced degree who earns 0.60157 and drives a sedan with a car safety rating of 0.69361, the probability that she will incur a loss greater than 228 units is 2.5% if you use the Burr distribution’s scale regression model for estimation. You can add a CDF value of 0.995 to the CDF= suboption of the QUANTILES= option to get another value-at-risk estimate, which estimates a more extreme loss that a policyholder might incur with a probability of only 0.5%, but you might need to be prepared to cover that loss depending on the policy provisions.

Output 6.7.6  Estimates of the Mean for Final Models of All Severity Distributions

<table>
<thead>
<tr>
<th>Logn_MEAN</th>
<th>Burr_MEAN</th>
<th>Weibull_MEAN</th>
<th>carType</th>
<th>gender</th>
<th>education</th>
<th>carSafety</th>
<th>income</th>
</tr>
</thead>
<tbody>
<tr>
<td>264.595</td>
<td>279.531</td>
<td>249.048</td>
<td>SUV</td>
<td>Male</td>
<td>College</td>
<td>0.85747</td>
<td>1.00896</td>
</tr>
<tr>
<td>90.959</td>
<td>93.358</td>
<td>88.674</td>
<td>Sedan</td>
<td>Male</td>
<td>AdvancedDegree</td>
<td>0.01410</td>
<td>0.51942</td>
</tr>
<tr>
<td>135.151</td>
<td>137.595</td>
<td>133.730</td>
<td>Sedan</td>
<td>Female</td>
<td>College</td>
<td>0.36374</td>
<td>0.38431</td>
</tr>
<tr>
<td>94.361</td>
<td>97.079</td>
<td>90.119</td>
<td>Sedan</td>
<td>Male</td>
<td>College</td>
<td>0.14417</td>
<td>0.67508</td>
</tr>
<tr>
<td>298.219</td>
<td>312.766</td>
<td>282.880</td>
<td>SUV</td>
<td>Male</td>
<td>College</td>
<td>0.84460</td>
<td>0.70732</td>
</tr>
<tr>
<td>144.016</td>
<td>148.594</td>
<td>137.050</td>
<td>Sedan</td>
<td>Male</td>
<td>High School</td>
<td>0.27925</td>
<td>0.15000</td>
</tr>
<tr>
<td>269.407</td>
<td>278.323</td>
<td>259.974</td>
<td>SUV</td>
<td>Male</td>
<td>High School</td>
<td>0.15692</td>
<td>0.47704</td>
</tr>
<tr>
<td>50.215</td>
<td>51.930</td>
<td>49.286</td>
<td>Sedan</td>
<td>Male</td>
<td>AdvancedDegree</td>
<td>0.61586</td>
<td>0.77170</td>
</tr>
<tr>
<td>77.799</td>
<td>79.648</td>
<td>78.331</td>
<td>Sedan</td>
<td>Female</td>
<td>AdvancedDegree</td>
<td>0.69361</td>
<td>0.60157</td>
</tr>
<tr>
<td>79.634</td>
<td>82.498</td>
<td>76.822</td>
<td>Sedan</td>
<td>Male</td>
<td>AdvancedDegree</td>
<td>0.00579</td>
<td>0.90709</td>
</tr>
</tbody>
</table>

Output 6.7.7  Estimates of the Median for Final Models of All Severity Distributions

<table>
<thead>
<tr>
<th>Logn_MEDIAN</th>
<th>Burr_MEDIAN</th>
<th>Weibull_MEDIAN</th>
<th>carType</th>
<th>gender</th>
<th>education</th>
<th>carSafety</th>
<th>income</th>
</tr>
</thead>
<tbody>
<tr>
<td>225.018</td>
<td>228.080</td>
<td>227.398</td>
<td>SUV</td>
<td>Male</td>
<td>College</td>
<td>0.85747</td>
<td>1.00896</td>
</tr>
<tr>
<td>77.354</td>
<td>76.174</td>
<td>80.966</td>
<td>Sedan</td>
<td>Male</td>
<td>AdvancedDegree</td>
<td>0.01410</td>
<td>0.51942</td>
</tr>
<tr>
<td>114.936</td>
<td>112.269</td>
<td>122.104</td>
<td>Sedan</td>
<td>Female</td>
<td>College</td>
<td>0.36374</td>
<td>0.38431</td>
</tr>
<tr>
<td>80.247</td>
<td>79.211</td>
<td>82.285</td>
<td>Sedan</td>
<td>Male</td>
<td>College</td>
<td>0.14417</td>
<td>0.67508</td>
</tr>
<tr>
<td>253.613</td>
<td>255.198</td>
<td>258.289</td>
<td>SUV</td>
<td>Male</td>
<td>College</td>
<td>0.84460</td>
<td>0.70732</td>
</tr>
<tr>
<td>122.475</td>
<td>121.244</td>
<td>125.136</td>
<td>Sedan</td>
<td>Male</td>
<td>High School</td>
<td>0.27925</td>
<td>0.15000</td>
</tr>
<tr>
<td>229.111</td>
<td>227.094</td>
<td>237.373</td>
<td>SUV</td>
<td>Male</td>
<td>High School</td>
<td>0.15692</td>
<td>0.47704</td>
</tr>
<tr>
<td>42.704</td>
<td>42.372</td>
<td>45.001</td>
<td>Sedan</td>
<td>Male</td>
<td>AdvancedDegree</td>
<td>0.61586</td>
<td>0.77170</td>
</tr>
<tr>
<td>66.162</td>
<td>64.988</td>
<td>71.522</td>
<td>Sedan</td>
<td>Female</td>
<td>AdvancedDegree</td>
<td>0.69361</td>
<td>0.60157</td>
</tr>
<tr>
<td>67.723</td>
<td>67.313</td>
<td>70.143</td>
<td>Sedan</td>
<td>Male</td>
<td>AdvancedDegree</td>
<td>0.00579</td>
<td>0.90709</td>
</tr>
</tbody>
</table>
Example 6.7: Scale Regression Model Selection

Output 6.7.8 Estimates of the Value-at-Risk (VaR) for Final Models of All Severity Distributions

<table>
<thead>
<tr>
<th>Logn_VAR</th>
<th>Burr_VAR</th>
<th>Weibull_VAR</th>
<th>carType</th>
<th>gender</th>
<th>education</th>
<th>carSafety</th>
<th>income</th>
</tr>
</thead>
<tbody>
<tr>
<td>686.684</td>
<td>801.100</td>
<td>585.528</td>
<td>SUV</td>
<td>Male</td>
<td>College</td>
<td>0.85747</td>
<td>1.00896</td>
</tr>
<tr>
<td>236.059</td>
<td>267.552</td>
<td>208.479</td>
<td>Sedan</td>
<td>Male</td>
<td>AdvancedDegree</td>
<td>0.01410</td>
<td>0.51942</td>
</tr>
<tr>
<td>350.747</td>
<td>394.330</td>
<td>314.407</td>
<td>Sedan</td>
<td>Female</td>
<td>College</td>
<td>0.36374</td>
<td>0.38431</td>
</tr>
<tr>
<td>244.888</td>
<td>278.217</td>
<td>211.876</td>
<td>Sedan</td>
<td>Male</td>
<td>College</td>
<td>0.14417</td>
<td>0.67508</td>
</tr>
<tr>
<td>773.946</td>
<td>896.348</td>
<td>665.069</td>
<td>SUV</td>
<td>Male</td>
<td>College</td>
<td>0.84460</td>
<td>0.70732</td>
</tr>
<tr>
<td>373.755</td>
<td>425.851</td>
<td>322.214</td>
<td>Sedan</td>
<td>Male</td>
<td>High School</td>
<td>0.27925</td>
<td>0.15000</td>
</tr>
<tr>
<td>699.172</td>
<td>797.637</td>
<td>611.215</td>
<td>SUV</td>
<td>Male</td>
<td>High School</td>
<td>0.15692</td>
<td>0.47704</td>
</tr>
<tr>
<td>130.319</td>
<td>148.824</td>
<td>115.874</td>
<td>Sedan</td>
<td>Male</td>
<td>AdvancedDegree</td>
<td>0.61586</td>
<td>0.77170</td>
</tr>
<tr>
<td>201.905</td>
<td>228.260</td>
<td>184.162</td>
<td>Sedan</td>
<td>Female</td>
<td>AdvancedDegree</td>
<td>0.69361</td>
<td>0.60157</td>
</tr>
<tr>
<td>206.668</td>
<td>236.428</td>
<td>180.612</td>
<td>Sedan</td>
<td>Male</td>
<td>AdvancedDegree</td>
<td>0.00579</td>
<td>0.90709</td>
</tr>
</tbody>
</table>

PROC SEVSELECT always includes the intercept in all scale regression models, so the selection process never attempts to fit a model without an intercept. In addition to the intercept, you can force other regression effects to be included in each selected model by specifying the INCLUDE= option in the SCALEMODEL statement. The following SEVSELECT step illustrates this by forcing the carSafety effect into every selected model:

```plaintext
proc sevselect data=mycas.losses outest=mycas.est print=all;
  loss lossAmount / lt=deductible rc=limit;
  class carType gender education;
  scalemodel carType gender carType*gender carSafety income education
     income*gender carSafety*income / include=(carSafety);
  selection method=forward(stop=aicc) hierarchy=single stophorizon=2 details=all;
  dist logn weibull;
run;
```

The various options in the SELECTION statement do the following:

- The METHOD=FORWARD option specifies that the forward elimination method of selection be used.
- The HIERARCHY=SINGLE option instructs PROC SEVSELECT not to add an interaction effect to the model until all the main effects in the interaction are already in the model.
- The STOPHORIZON=2 option specifies that the STOP= criterion, which is the corrected Akaike’s information criterion (AICC), must worsen for two steps in order for a local extremum to be detected.
- The DETAILS=ALL option displays details of all the steps of the selection process.

The “Selection Information” table in Output 6.7.9 summarizes the selection settings.
Output 6.7.9  Selection Method Settings for Forward Selection

The SEVSELECT Procedure

Logn Distribution

<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 for the lognormal distribution, shown in Output 6.7.10, illustrates the following:

- The model at step 0 includes both the intercept and carSafety effects. The carSafety effect always stays in the model.

- Because of the HIERARCHY=SINGLE option, the carType*gender interaction effect does not enter the model until the model contains both main effects (carType and gender) that constitute it. Similarly, the carSafety*income interaction effect between two continuous effects also does not enter the model before both carSafety and income are in the model.

- The stop criterion keeps improving for several steps, so even the smaller stop horizon of 2 does not affect the selection process for the lognormal distribution. The selection stops when all effects are in the model.

- The final model is chosen by the optimal value of the STOP= criterion, because the CHOOSE= option is not specified. For the lognormal distribution, it is the model at step 6, which contains all regression effects except the income*gender effect. If you had specified the CHOOSE=SBC option, then the final model would be the model in step 4 that excludes carType*gender, carSafety*income, and income*gender effects, because it has the smallest value for the SBC criterion.
Output 6.7.10  Forward Selection Summary for the Lognormal Distribution

**The SEVSELECT Procedure**

**Lognormal Distribution**

**Selection Details**

<table>
<thead>
<tr>
<th>Step</th>
<th>Effect Entered</th>
<th>Number Effects In</th>
<th>AICC</th>
<th>SBC</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Intercept</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>carSafety</td>
<td>2</td>
<td>35766.6044</td>
<td>35786.1512</td>
</tr>
<tr>
<td>1</td>
<td>carType</td>
<td>3</td>
<td>34230.4771</td>
<td>34256.5378</td>
</tr>
<tr>
<td>2</td>
<td>gender</td>
<td>4</td>
<td>33895.0222</td>
<td>33927.5962</td>
</tr>
<tr>
<td>3</td>
<td>income</td>
<td>5</td>
<td>33656.7655</td>
<td>33695.8519</td>
</tr>
<tr>
<td>4</td>
<td>education</td>
<td>6</td>
<td>33594.6989</td>
<td>33646.8076*</td>
</tr>
<tr>
<td>5</td>
<td>carType*gender</td>
<td>7</td>
<td>33591.2653</td>
<td>33649.8839</td>
</tr>
<tr>
<td>6</td>
<td>carSafety*income</td>
<td>8</td>
<td>33590.2351*</td>
<td>33655.3629</td>
</tr>
<tr>
<td>7</td>
<td>income*gender</td>
<td>9</td>
<td>33591.8039</td>
<td>33663.4401</td>
</tr>
</tbody>
</table>

* Optimal Value Of Criterion

Selection stopped because all effects are in the model.

The model at step 6 is selected.

**Selected Effects:** Intercept carType gender carType*gender carSafety income education carSafety*income

The selection summary for the Weibull distribution is shown in Output 6.7.11. It indicates that the selection stopped at the local minimum of the AICC statistic at step 5 because of the shorter stop horizon.

Output 6.7.11  Forward Selection Summary for the Weibull Distribution

**The SEVSELECT Procedure**

**Weibull Distribution**

**Selection Details**

<table>
<thead>
<tr>
<th>Step</th>
<th>Effect Entered</th>
<th>Number Effects In</th>
<th>AICC</th>
<th>SBC</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Intercept</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>carSafety</td>
<td>2</td>
<td>36202.4284</td>
<td>36221.9751</td>
</tr>
<tr>
<td>1</td>
<td>carType</td>
<td>3</td>
<td>34560.1129</td>
<td>34586.1736</td>
</tr>
<tr>
<td>2</td>
<td>gender</td>
<td>4</td>
<td>34200.9756</td>
<td>34233.5496</td>
</tr>
<tr>
<td>3</td>
<td>income</td>
<td>5</td>
<td>33935.3440</td>
<td>33974.4303</td>
</tr>
<tr>
<td>4</td>
<td>education</td>
<td>6</td>
<td>33857.1156</td>
<td>33909.2243*</td>
</tr>
<tr>
<td>5</td>
<td>carType*gender</td>
<td>7</td>
<td>33856.5340*</td>
<td>33915.1526</td>
</tr>
<tr>
<td>6</td>
<td>carSafety*income</td>
<td>8</td>
<td>33857.1143</td>
<td>33922.2421</td>
</tr>
</tbody>
</table>

* Optimal Value Of Criterion

Selection stopped at a local minimum of the AICC criterion.
As the results in Output 6.7.10 and Output 6.7.11 show, although the SELECT= criterion is reported in the “Selection Summary” table, it does not decide the final selected model. Instead, the SELECT= criterion is used to choose the best effect to add or remove in each step. To illustrate that, Output 6.7.12 shows the details of step 1 for the Weibull distribution. PROC SEVSELECT produces the step details because of the DETAILS=ALL option in the SELECTION statement. Step 1 is the step after the initial step (step 0), which fits the model with the intercept and carSafety effect that the INCLUDE= option forces in. The “Entry Candidates” table of step 1 shows the candidates that PROC SEVSELECT evaluates for entering the model. For each candidate, it shows the value of the SELECT= criterion, which is the SBC statistic, of the model that includes that candidate. The interaction effects carType*gender and carSafety*income are not present in this list of candidates, because of the constraint that the HIERARCHY=SINGLE option imposes. The model that includes the carType effect has the smallest SBC statistic, so PROC SEVSELECT selects it in step 1. The “Optimization Summary” table shows that PROC SEVSELECT uses the dual quasi-Newton optimization method, which is the default optimization method for fitting a model in each step. You can specify the SELECTNLOTECH= option in the PROC SEVSELECT statement to specify a different optimization method to use during each step of the selection. The “Parameter Estimates” table shows that step 1’s model includes the best entry candidate, carType, and the carSafety effect, which entered the model in step 0. The estimate of the intercept effect is reflected in the estimate of the scale parameter Theta, as described in the section “Reporting Estimates of Regression Parameters” on page 259.
The fit summary of the final model for the Weibull distribution is shown in Output 6.7.13. It shows that PROC SEVSELECT uses the trust region optimization method to estimate the parameters of the final model, which is different from the default optimization method that it uses to fit the models within each step. The “Parameter Estimates” table shows that some regression parameters are not significant at the usual 95% confidence level. None of the selection criteria that PROC SEVSELECT supports depend directly on the p-values of regression effects. So a statistically insignificant effect might be included in the final model, as this example illustrates. If you do not want such effects in the final model, you can modify the selection method settings to search for a different model that contains all statistically significant effects. Alternatively, you can use your domain knowledge to decide which effects you want to retain from the selected model and then refit the model by appropriately modifying the SCALEMODEL statement.

Output 6.7.12 continued

| Parameter         | DF | Estimate | Standard Error | t Value | Approx Pr > |t| |
|-------------------|----|----------|----------------|---------|-------------|---|
|Theta              | 1  | 172.58425| 4.92614        | 35.03   | <.0001      |   |
|Tau                | 1  | 1.39719  | 0.02266        | 61.65   | <.0001      |   |
|carType SUV        | 1  | 1.28993  | 0.02919        | 44.18   | <.0001      |   |
|carType Sedan      | 0  | 0        | .              | .       | .           |   |
|carSafety          | 1  | -0.85372 | 0.04522        | -18.88  | <.0001      |   |
References


Subject Index

Amemiya-MaCurdy estimation, 143
CPANEL procedure, 143
ANCOVA effects
  Shared Concepts, 37
ANOVA effects
  Shared Concepts, 36
at sign (@) operator
  Shared Concepts, 35, 38
B-spline
  spline basis (Shared Concepts), 20
B-spline basis
  CCOPULA procedure, 20
  CNTSELECT procedure, 20
  CPANEL procedure, 20
  CQLIM procedure, 20
  SEVSELECT procedure, 20
backward elimination
  Shared Concepts, 46
bar (|) operator
  Shared Concepts, 34, 37, 38
between-groups regression, 133
  CPANEL procedure, 133
  instrumental variables, 145
bounds on parameter estimates, 179
BOUND S statement, 179
BY groups
  CNTSELECT procedure, 80
  SEVSELECT procedure, 224
CCOPULA procedure
  B-spline basis, 20
  computational method, 49, 50
  convergence criterion, 27–31
  function-based convergence criteria, 27, 28
  gradient-based convergence criteria, 27, 29
  multithreading, 49
  natural cubic spline basis, 22
  ODS table names, 22
  optimization techniques, 30, 50
  output table names, 65
  overview, 55
  parameter-based convergence criteria, 27, 30, 31
  splines and spline bases, 18
  syntax, 58
  truncated power function (TPF) basis, 19
censored regression models
  CQLIM procedure, 189
censoring and truncation
  SEVSELECT procedure, 251
class level
  CPANEL procedure, 149
CLASS statement
  Shared Concepts, 31
  syntax (Shared Concepts), 4, 31
classification variables
  Shared Concepts, 31
Clayton copula, 64
CNTSELECT procedure
  B-spline basis, 20
  bounds on parameter estimates, 79
  BY groups, 80
  computational method, 49, 50
  convergence criterion, 27–31
  function-based convergence criteria, 27, 28
  gradient-based convergence criteria, 27, 29
  multithreading, 49
  natural cubic spline basis, 22
  ODS table names, 22
  optimization techniques, 30, 50
  output table names, 106
  parameter-based convergence criteria, 27, 30, 31
  restrictions on parameter estimates, 85
  splines and spline bases, 18
  syntax, 75
  truncated power function (TPF) basis, 19
colon (:) operator
  Shared Concepts, 35
comparison tables
  CPANEL procedure, 121, 151
computational method
  CCOPULA procedure, 49, 50
  CNTSELECT procedure, 49, 50
  CPANEL procedure, 49, 50
  CQLIM procedure, 49, 50
  SEVSELECT procedure, 49, 50
conjugate-gradient optimization method, 79
continuous-by-class effects
  Shared Concepts, 39
continuous-nesting-class effects
  Shared Concepts, 39
convergence criterion
  CCOPULA procedure, 27–31
  CNTSELECT procedure, 27–31
  CPANEL procedure, 27–31
  CQLIM procedure, 27–31
  SEVSELECT procedure, 27–31
Subject Index

Shared Concepts, 27–31

copulas
Clayton, 64
Frank, 64
Gumbel, 64
normal, 62
Student’s t, 63
correlated variables
CPANEL procedure, 123
correlation matrix
CPANEL procedure, 151
covariance matrix
CPANEL procedure, 151
covariates
heteroscedasticity models, 181
CPANEL procedure, 112
Amemiya-MacCurdy estimation, 143
B-spline basis, 20
between-groups regression, 133
class level, 149
comparison tables, 121, 151
computational method, 49, 50
convergence criterion, 27–31
correlated variables, 123
correlation matrix, 151
covariance matrix, 151
displayed output, 149
dependent variables, 123
fit statistics, 150
fixed-effects test, 150
function-based convergence criteria, 27, 28
gradient-based convergence criteria, 27, 29
Hausman test, 150
Hausman tests, 147
Hausman-Taylor estimation, 142
ID variables, 124
input data tables, 119
instrumental variables, 124, 144
linear hypothesis testing, 148
model comparisons, 121
model information, 149
multithreading, 49
natural cubic spline basis, 22
ODS table names, 22
one-way fixed-effects by first differencing, 134
one-way fixed-effects model, 133
one-way random-effects model, 135
optimization techniques, 30, 50
output data tables, 128
parameter estimates, 150
parameter-based convergence criteria, 27, 30, 31
partitioning data, 120
poolability test, 150
poolability test, fixed effects, 146
pooled regression, 133
predicted values, 129
printing options, 120
R-square measure, 148
residuals, 129
splines and spline bases, 18
structural summary, 150
tests of linear hypotheses, 151
truncated power function (TPF) basis, 19
two-way fixed-effects model, 134
two-way random-effects model, 138
variance components, 150
CQLIM procedure, 172
B-spline basis, 20
BY groups, 179
censored regression models, 189
computational method, 49, 50
convergence criterion, 27–31
frontier, 190
function-based convergence criteria, 27, 28
gradient-based convergence criteria, 27, 29
heteroscedasticity, 191
limited dependent variable models, 189
multithreading, 49
natural cubic spline basis, 22
ODS table names, 22
optimization techniques, 30, 50
ordinal discrete choice modeling, 188
output, 193
output table names, 197
parameter-based convergence criteria, 27, 30, 31
splines and spline bases, 18
syntax, 176
tests on parameters, 192
truncated power function (TPF) basis, 19
truncated regression models, 189
crossed effects
Shared Concepts, 38
dash (-) operator
Shared Concepts, 35
defining a custom probability distribution
SEVSELECT procedure, 272
dependence measures, 61
displayed output
CPANEL procedure, 149
double dash (- -) operator
Shared Concepts, 36
double-dogleg optimization method, 79
dummy parameterization
Shared Concepts, 36
effect parameterization
Shared Concepts, 41
EFFECT statement
  collection effect (Shared Concepts), 10
  multimember effect (Shared Concepts), 10
  polynomial effect (Shared Concepts), 12
  spline effect (Shared Concepts), 15
  syntax (Shared Concepts), 8

empirical distribution function
  SEVSELECT procedure, 261

endogenous variables
  CPANEL procedure, 123
examples, CCOPIULA
  multimember effect, 11
examples, CNTSELECT
  multimember effect, 11
examples, CPANEL
  multimember effect, 11
examples, CQLIM
  multimember effect, 11
examples, SEVSELECT
  multimember effect, 11

fit statistics
  CPANEL procedure, 150
fitting custom probability distributions
  SEVSELECT procedure, 226

fixed-effects model
  one-way, 133
  one-way first differencing, 134
  poolability test, 146
  two-way, 134

fixed-effects test
  CPANEL procedure, 150
forward selection
  Shared Concepts, 44
forward swap selection
  Shared Concepts, 47
Frank copula, 64
Fuller-Battese
  variance components, 137, 141

function-based convergence criteria
  CCOPIULA procedure, 27, 28
  CNTSELECT procedure, 27, 28
  CPANEL procedure, 27, 28
  CQLIM procedure, 27, 28
  SEVSELECT procedure, 27, 28
  Shared Concepts, 27, 28

general effects
  Shared Concepts, 40
GLM parameterization
  Shared Concepts, 36
gradient-based convergence criteria
  CCOPIULA procedure, 27, 29
  CNTSELECT procedure, 27, 29
  CPANEL procedure, 27, 29
  CQLIM procedure, 27, 29
  SEVSELECT procedure, 27, 29
  Shared Concepts, 27, 29

Gumbel copula, 64

Hausman
  specification tests, 147
  Hausman specification test, 147
  random-effects model, 147
  Hausman test
    CPANEL procedure, 150
  Hausman tests
    CPANEL procedure, 147
  Hausman-Taylor estimation, 142
    CPANEL procedure, 142
heteroscedasticity models
  covariates, 181

ID variables
  CPANEL procedure, 124
instrumental variables
  between-groups regression, 145
  CPANEL procedure, 124, 144
  one-way fixed effects, 145
  one-way random effects, 146
  pooled regression, 144
  regression, 144
interaction effects
  Shared Concepts, 38
intercept
  Shared Concepts, 37
Kaplan-Meier’s EDF estimator
  SEVSELECT procedure, 263

Lagrange multiplier test
  nonlinear hypotheses, 192
levelization
  Shared Concepts, 31
limited dependent variable models
  CQLIM procedure, 189
linear hypothesis testing, 148
  CPANEL procedure, 148
loss distribution modeling
  SEVSELECT procedure, 202

main effects
  Shared Concepts, 37
measure
  dependence, 61
model comparisons
  CPANEL procedure, 121
model information
  CPANEL procedure, 149
modified Kaplan-Meier’s EDF estimator
SEVSELECT procedure, 263

multithreading
CCOPULA procedure, 49
CNTSELECT procedure, 49
CPANEL procedure, 49
CQLIM procedure, 49
SEVSELECT procedure, 49

natural cubic spline basis
CCOPULA procedure, 22
CNTSELECT procedure, 22
CPANEL procedure, 22
CQLIM procedure, 22
SEVSELECT procedure, 22

Newton-Raphson optimization method, 79
Newton-Raphson Ridge optimization method, 79
nonlinear hypotheses
Lagrange multiplier test, 192
nonsingular parameterization
Shared Concepts, 40
normal copula, 62

ODS table names
CCOPULA procedure, 22
CNTSELECT procedure, 22
CPANEL procedure, 22
CQLIM procedure, 22
SEVSELECT procedure, 22, 295

one-way
fixed-effects model, 133
random-effects model, 135
one-way first differencing
fixed-effects model, 134
one-way fixed effects
instrumental variables, 145
one-way fixed-effects by first differencing, 134
CPANEL procedure, 134
one-way fixed-effects model, 133
CPANEL procedure, 133
one-way random effects
instrumental variables, 146
one-way random-effects model, 135
CPANEL procedure, 135

parameter estimates
CPANEL procedure, 150
parameter-based convergence criteria
CCOPULA procedure, 27, 30, 31
CNTSELECT procedure, 27, 30, 31
CPANEL procedure, 27, 30, 31
CQLIM procedure, 27, 30, 31
SEVSELECT procedure, 27, 30, 31
Shared Concepts, 27, 30, 31

optimization techniques
CCOPULA procedure, 30, 50
CNTSELECT procedure, 30, 50
CPANEL procedure, 30, 50
CQLIM procedure, 30, 50
SEVSELECT procedure, 30, 50
Shared Concepts, 30

options summary
CLASS statement, 5
EFFECT statement, 9, 229
MODEL statement (CPANEL), 125

ordering
of class levels (Shared Concepts), 32

ordinal discrete choice modeling
CQLIM procedure, 188

ordinanl parameterization
Shared Concepts, 41

ORTHEFFECT parameterization
Shared Concepts, 42

ORTHORDINAL parameterization
Shared Concepts, 42

ORTHOTHERM parameterization
Shared Concepts, 42

ORTHPOLY parameterization
Shared Concepts, 42

ORTHREF parameterization
Shared Concepts, 42

output table names
CCOPULA procedure, 65
CNTSELECT procedure, 106
CQLIM procedure, 197

parameterization
dummy (Shared Concepts), 36
effect (Shared Concepts), 41
GLM (Shared Concepts), 36
nonsingular (Shared Concepts), 40
ordinal (Shared Concepts), 41
ORTHEFFECT (Shared Concepts), 42
ORTHORDINAL (Shared Concepts), 42
ORTHOTHERM (Shared Concepts), 42
ORTHPOLY (Shared Concepts), 42
ORTHREF (Shared Concepts), 43
polynomial (Shared Concepts), 42
reference (Shared Concepts), 41
Shared Concepts, 33
thermometer (Shared Concepts), 41
partitioning data
CPANEL procedure, 120
polynomial effects
Shared Concepts, 37
polynomial parameterization
Shared Concepts, 42
poolability test, 146
CPANEL procedure, 150
fixed-effects model, 146
poolability test, fixed effects
CPANEL procedure, 146
pooled regression, 133
CPANEL procedure, 133
instrumental variables, 144
predicted values
CPANEL procedure, 129
printing options
CPANEL procedure, 120
probability of observability
SEVSELECT procedure, 252
quasi-Newton optimization method, 79
R-square measure
CPANEL procedure, 148
random-effects model
Hausman specification test, 147
one-way, 135
two-way, 138
reference parameterization
Shared Concepts, 41
regression effects
Shared Concepts, 36, 37
residuals
CPANEL procedure, 129
scale regression model
SEVSELECT procedure, 256
selecting
variance components, 138
SELECTION statement
syntax (Shared Concepts), 23
SEVSELECT procedure
B-spline basis, 20
BY groups, 224
censoring and truncation, 251
computational method, 49, 50
convergence criterion, 27–31
defining a custom probability distribution, 272
dependent distribution function, 261
dependent function-based convergence criteria, 27, 28
dependent gradient-based convergence criteria, 27, 29
dependent input data tables, 219
dependent multithreading, 49
dependent natural cubic spline basis, 22
dependent ODS table names, 22, 295
dependent optimization techniques, 30, 50
dependent output data tables, 233
dependent parameter-based convergence criteria, 27, 30, 31
dependent predefined distributions, 241
dependent predefined utility functions, 285
dependent probability of observability, 252
dependent scale regression model, 256
dependent splines and spline bases, 18
dependent statistics of fit, 267
dependent truncated power function (TPF) basis, 19
dependent Tweedie distribution, 243
Shared Concepts
ABSCONV option, 27
ABSFCOV option, 27
ABSFTOL option, 27
ABSGCONV option, 27
ABSGTOL option, 27
ABSTOL option, 27
ABSXCONV option, 27
ABSXTOL option, 27
ANCOVA effects, 37
ANOVA effects, 36
at sign (@) operator, 35, 38
backward elimination, 46
bar (|) operator, 34, 37, 38
CLASS statement, 4, 31
classification variables, 31
collection effect (EFFECT statement), 10
colon (:) operator, 35
continuous-by-class effects, 39
continuous-nesting-class effects, 39
covariance criterion, 27–31
crossed effects, 38
dash (-) operator, 35
double dash (- -) operator, 36
dummy parameterization, 36
effect parameterization, 41
EFFECT statement, 8
FCONV option, 28
FCONV2 option, 28
forward selection, 44
forward swap selection, 47
FSIZE option, 28
FTOL option, 28
FTOL2 option, 28
function-based convergence criteria, 27, 28
Subject Index

GCONV option, 29
GCONV2 option, 29
general effects, 40
GLM parameterization, 36
gradient-based convergence criteria, 27, 29
GTOL option, 29
GTOL2 option, 29
interaction effects, 38
intercept, 37
levelization, 31
main effects, 37
MAXFUNC= option, 29
MAXITER= option, 29
MAXTIME= option, 30
MINITER= option, 30
missing values, CLASS variables, 32
multimember effect (EFFECT statement), 10
nested effects, 38
nested versus crossed effects, 38
nonsingular parameterization, 40
optimization techniques, 30
ORDER= option, 32
ordering of class levels, 32
ordinal parameterization, 41
ORTHEFFECT parameterization, 42
ORTHORDINAL parameterization, 42
ORTHOTHERM parameterization, 42
ORTHPOLY parameterization, 42
ORTHREF parameterization, 43
parameter-based convergence criteria, 27, 30, 31
parameterization, 33
polynomial effect (EFFECT statement), 12
polynomial effects, 37
polynomial parameterization, 42
reference parameterization, 41
regression effects, 36, 37
SELECTION statement, 23
singular parameterization, 37
sort order of class levels, 32
spline basis, B-spline, 20
spline basis, natural cubic spline, 22
spline basis, truncated power function, 19
spline effect (EFFECT statement), 15
stepwise selection, 46
TECHNIQUE= option, 30
thermometer parameterization, 41
XCONV option, 30
XSIZE option, 31
XTOL option, 30
singular parameterization
Shared Concepts, 37
Sklar’s theorem, 60
sort order
of class levels (Shared Concepts), 32
splines and spline bases
CCOPULA procedure, 18
CNTSELECT procedure, 18
CPANEL procedure, 18
CQLIM procedure, 18
SEVSELECT procedure, 18
statistics of fit
SEVSELECT procedure, 267
stepwise selection
Shared Concepts, 46
structural summary
CPANEL procedure, 150
Student’s t copula, 63
Swamy-Arora
variance components, 136, 139
tests of linear hypotheses
CPANEL procedure, 151
thermometer parameterization
Shared Concepts, 41
TPF basis, see truncated power function (TPF) basis
truncated power function (TPF) basis
CCOPULA procedure, 19
CNTSELECT procedure, 19
CPANEL procedure, 19
CQLIM procedure, 19
SEVSELECT procedure, 19
Shared Concepts, 19
truncated regression models
CQLIM procedure, 189
trust region optimization method, 79
Turnbull’s EDF estimator
SEVSELECT procedure, 264
Tweedie distribution
SEVSELECT procedure, 243
two-stage least squares (2SLS) regression, 142–144
two-way
fixed-effects model, 134
random-effects model, 138
two-way fixed-effects model, 134
CPANEL procedure, 134
two-way random-effects model, 138
CPANEL procedure, 138
variance components
CPANEL procedure, 150
Fuller-Battese, 137, 141
Nerlove, 138, 141
selecting, 138
Swamy-Arora, 136, 139
Wallace-Hussain, 136, 140
Wansbeek-Kapteyn, 137, 140
Wallace-Hussain
variance components, 136, 140
Wansbeek-Kapteyn
  variance components, 137, 140
Syntax Index

ABSCONV option
PROC CCOPULA statement, 27
PROC CNTSELECT statement, 27
PROC CPANEL statement, 27
PROC CQLIM statement, 27
PROC SEVSELECT statement, 27
ABSFCONV option
PROC CCOPULA statement, 27
PROC CNTSELECT statement, 27
PROC CPANEL statement, 27
PROC CQLIM statement, 27
PROC SEVSELECT statement, 27
ABSFTOL option
PROC CCOPULA statement, 27
PROC CNTSELECT statement, 27
PROC CPANEL statement, 27
PROC CQLIM statement, 27
PROC SEVSELECT statement, 27
ABSFTOL option
PROC CCOPULA statement, 27
PROC CNTSELECT statement, 27
PROC CPANEL statement, 27
PROC CQLIM statement, 27
PROC SEVSELECT statement, 27
ABSTOL option
PROC CCOPULA statement, 27
PROC CNTSELECT statement, 27
PROC CPANEL statement, 27
PROC CQLIM statement, 27
PROC SEVSELECT statement, 27
ABSTOL option
PROC CCOPULA statement, 27
PROC CNTSELECT statement, 27
PROC CPANEL statement, 27
PROC CQLIM statement, 27
PROC SEVSELECT statement, 27
ABSXCONV option
PROC CCOPULA statement, 27
PROC CNTSELECT statement, 27
PROC CPANEL statement, 27
PROC CQLIM statement, 27
PROC SEVSELECT statement, 27
ABSXCONV option
PROC CCOPULA statement, 27
PROC CNTSELECT statement, 27
PROC CPANEL statement, 27
PROC CQLIM statement, 27
PROC SEVSELECT statement, 27
ABXTOL option
PROC CCOPULA statement, 27
PROC CNTSELECT statement, 27
PROC CPANEL statement, 27
PROC CQLIM statement, 27
PROC SEVSELECT statement, 27
ALL option
TEST statement (CNTSELECT), 86
TEST statement (CPANEL), 130
TEST statement (CQLIM), 187
AMACURDY option
MODEL statement (CPANEL), 126
BASIS option
EFFECT statement, spline (CCOPULA), 15
EFFECT statement, spline (CNTSELECT), 15
EFFECT statement, spline (CPANEL), 15
EFFECT statement, spline (CQLIM), 15
EFFECT statement, spline (SEVSELECT), 15
BOUNDS statement
CNTSELECT procedure, 79
CQLIM procedure, 179
BTWNG option
MODEL statement (CPANEL), 126
BY statement
CNTSELECT procedure, 80
CPANEL procedure, 120
CQLIM procedure, 179
SEVSELECT procedure, 224
CASESENSITIVE option
DISPLAY statement (SEVSELECT), 225
CCOPULA procedure, 58
DEFINE statement, 59
PROC CCOPULA statement, 59
SIMULATE statement, 60
syntax, 58
VAR Statement, 60
CCOPULA procedure, CLASS statement
DESCENDING option, 5
MISSING option, 5
ORDER= option, 5
PARAM= option, 5
REF= option, 6
SPLIT option, 7
CCOPULA procedure, EFFECT statement
BASIS option (spline), 15
DATABOUNDARY option (spline), 15
DEGREE option (polynomial), 13
DEGREE option (spline), 16
DETAILS option (multimember), 11
DETAILS option (polynomial), 13
DETAILS option (spline), 16
KNOTMAX= option (spline), 16
KNOTMETHOD option (spline), 16
KNOTMIN= option (spline), 17
MDEGREE option (polynomial), 13
NATURALCUBIC option (spline), 17
NOEFFECT option (multimember), 11
NOSEPARATE option (polynomial), 13
SEPARATE option (spline), 17
SPLIT option (spline), 18
STANDARDIZE option (polynomial), 13

CCOPULA procedure, PROC CCOPULA statement
ABSCONV option, 27
ABSFCONV option, 27
ABSFTOL option, 27
ABSGCONV option, 27
ABSGTOL option, 27
ABSTOL option, 27
ABSXCONV option, 27
ABSXTOL option, 27
FCONV option, 28
FCONV2 option, 28
FSIZE option, 28
FTOL option, 28
FTOL2 option, 28
GCONV option, 29
GCONV2 option, 29
GTOL option, 29
GTOL2 option, 29
MAXFUNC= option, 29
MAXITER= option, 29
MAXTIME= option, 30
MINITER= option, 30
TECHNIQUE= option, 30
XCONV option, 30
XSIZE option, 31
XTOL option, 30

CCOPULA procedure, SELECTION statement
CHOOSE= option, 24
COMPETITIVE option, 24
DETAILS option, 24
HIERARCHY= option, 25
MAXEFFECTS= option, 24
MAXSTEPS= option, 24
METHOD= option, 23
MINEFFECTS= option, 24
ORDERSELECT option, 26
SELECT= option, 24
SELECTION= option, 26
STOP= option, 24
STOTHORIZON= option, 26

CENSORED option
ENDOGENOUS statement (CQLIM), 180, 184

CHOOSE= option
SELECTION statement (CCOPULA), 24
SELECTION statement (CNTSELECT), 24

CLASS statement
CCOPULA procedure, 4, 31
CNTSELECT procedure, 4, 31
CPANEL procedure, 4, 31
CQLIM procedure, 4, 31
SEVSELECT procedure, 4, 31, 224

CNTSELECT procedure, 75
syntax, 75

CNTSELECT procedure, CLASS statement, 80
DESCENDING option, 5
MISSING option, 5
ORDER= option, 5
PARAM= option, 5
REF= option, 6
SPLIT option, 7

CNTSELECT procedure, EFFECT statement
BASIS option (spline), 15
DATABASEBOUNDARY option (spline), 15
DEGREE option (polynomial), 13
DEGREE option (spline), 16
DETAILS option (multimember), 11
DETAILS option (polynomial), 13
DETAILS option (spline), 16
KNOTMAX= option (spline), 16
KNOTMETHOD option (spline), 16
KNOTMIN= option (spline), 17
MDEGREE option (polynomial), 13
NATURALCUBIC option (spline), 17
NOEFFECT option (multimember), 11
NOSEPARATE option (polynomial), 13
SEPARATE option (spline), 17
SPLIT option (spline), 18
STANDARDIZE option (polynomial), 13

CNTSELECT procedure, PROC CNTSELECT statement
ABSCONV option, 27
ABSFCONV option, 27
ABSFTOL option, 27
ABSGCONV option, 27
ABSGTOL option, 27
ABSTOL option, 27
ABSXCONV option, 27
ABSXTOL option, 27
FCONV option, 28
FCONV2 option, 28
FSIZE option, 28
FTOL option, 28
FTOL2 option, 28
GCONV option, 29
GCONV2 option, 29
GTOL option, 29

GTOL2 option, 29
MAXFUNC= option, 29
MAXITER= option, 29
MAXTIME= option, 30
MINITER= option, 30
TECHNIQUE= option, 30
XCONV option, 30
XSIZE option, 31
XTOL option, 30

CNTSELECT procedure, SELECTION statement
  CHOOSE= option, 24
  COMPETITIVE option, 24
  CRITERION= option, 24
  DETAILS= option, 25
  HIERARCHY= option, 25
  MAXEFFECTS= option, 24
  MAXSTEPS= option, 24
  METHOD= option, 23
  MINEFFECTS= option, 24
  ORDERSELECT option, 26
  SELECT= option, 24
  SELECTION= option, 26
  STOP= option, 24
  STOPHORIZON= option, 26

CNTSELECT procedure, TEST statement, 85
CNTSELECT procedure, WEIGHT statement, 87

COMPARE statement
  CPANEL procedure, 121

COMPETITIVE option
  SELECTION statement (CCOPULA), 24
  SELECTION statement (CNTSELECT), 24
  SELECTION statement (CPANEL), 24
  SELECTION statement (CQLIM), 24
  SELECTION statement (SEVSELECT), 24

CONDITIONAL
  OUTPUT statement (CQLIM), 185

COPYVAR= option
  OUTPUT statement (CNTSELECT), 83
  OUTPUT statement (CPANEL), 128
  OUTPUT statement (CQLIM), 185

COPYVARS= option
  OUTPUT statement (SEVSELECT), 233

CORRB option
  CQLIM procedure, 178
  MODEL statement, 83
  MODEL statement (CPANEL), 128
  PROC CNTSELECT statement, 78

CORRELATED statement
  CPANEL procedure, 123

COST option
  ENDogenous statement (CQLIM), 181, 184

COVB option
  CQLIM procedure, 178
  MODEL statement, 83
  MODEL statement (CPANEL), 128
  PROC CNTSELECT statement, 78

COVEST= option
  CQLIM procedure, 178
  PROC CNTSELECT statement, 78

COVOUT option
  PROC SEVSELECT statement, 218

CPANEL procedure, 118
  details, 130
  syntax, 118

CPANEL procedure, CLASS statement
  DESCENDING option, 5
  MISSING option, 5
  ORDER= option, 5
  PARAM= option, 5
  REF= option, 6
  SPLIT option, 7

CPANEL procedure, COMPARE statement, 121
  MSTAT option, 121
  OUTPARM= option, 122
  OUTSTAT= option, 122
  PSTAT option, 123

CPANEL procedure, CORRELATED statement, 123

CPANEL procedure, EFFECT statement
  BASIS option (spline), 15
  DATABOUNDARY option (spline), 15
  DEGREE option (polynomial), 13
  DEGREE option (spline), 16
  DETAILS option (multimember), 11
  DETAILS option (polynomial), 13
  DETAILS option (spline), 16
  KNOTMAX= option (spline), 16
  KNOTMETHOD option (spline), 16
  KNOTMIN= option (spline), 17
  MDEGREE option (polynomial), 13
  NATURALCUBIC option (spline), 17
  NOEFFECT option (multimember), 11
  NOSEPARATE option (polynomial), 13
  SEPARATE option (spline), 17
  SPLIT option (spline), 18
  STANDARDIZE option (polynomial), 13

CPANEL procedure, ENDOGENOUS statement, 123
CPANEL procedure, ID statement, 124
CPANEL procedure, INSTRUMENTS statement, 124
CPANEL procedure, MODEL statement, 125
  AMACURDY option, 126
  BTWNG option, 126
  CORRB option, 128
  COVB option, 128
  FDONE option, 126
  FIXONE option, 126
  FIXTWO option, 126
  HTAYLOR option, 126
  IVBTWNG option, 127
IVFIXONE option, 127
IVPOOLED option, 127
IVRANONE option, 127
NOINT option, 127
NOLABEL option, 128
POOLED option, 127
PRINTFIXED option, 128
RANONE option, 127
RANTWO option, 127
VCOMP= option, 127
CPANEL procedure, OUTPUT statement, 128
COPYVAR= option, 128
OUT= option, 128
P= option, 129
PREDICTED= option, 129
R= option, 129
RESIDUAL= option, 129
CPANEL procedure, PROC CPANEL statement, 119
ABSCONV option, 27
ABSFCONV option, 27
ABSFTOL option, 27
ABSGCONV option, 27
ABSGTOL option, 27
ABSTOL option, 27
ABSXCONV option, 27
ABSXTOL option, 27
DATA= option, 119
FCONV option, 28
FCONV2 option, 28
FSIZE option, 28
FTOL option, 28
FTOL2 option, 28
GCONV option, 29
GCONV2 option, 29
GTOL option, 29
GTOL2 option, 29
MAXEFFECTS= option, 24
MAXSTEPS= option, 24
METHOD= option, 23
MINEFFECTS= option, 24
ORDERSELECT option, 26
SELECT= option, 24
SELECTION= option, 26
STOP= option, 24
STOPHORIZON= option, 26
CPANEL procedure, TEST statement, 129
ALL option, 130
LM option, 130
LR option, 130
WALD option, 130
CQLIM procedure, 176
syntax, 176
CQLIM procedure, CLASS statement
DESCENDING option, 5
MISSING option, 5
ORDER= option, 5
PARAM= option, 5
REF= option, 6
SPLIT option, 7
CQLIM procedure, EFFECT statement
BASIS option (spline), 15
DATA= option (spline), 15
DEGREE option (polynomial), 13
DEGREE option (spline), 16
DETAILS option (multimember), 11
DETAILS option (polynomial), 13
DETAILS option (spline), 16
KNOTMAX= option (spline), 16
KNOTMETHOD option (spline), 16
KNOTMIN= option (spline), 17
MDEGREE option (polynomial), 13
MDEGREE option (polynomial), 13
NOEFFECT option (multimember), 11
NOEFFECT option (polynomial), 13
NOSEPARATE option (polynomial), 13
SEPARATE option (spline), 17
SPLIT option (spline), 18
STANDARDIZE option (polynomial), 13
CQLIM procedure, FREQ statement, 181
CQLIM procedure, PROC CQLIM statement
ABSCONV option, 27
ABSFCONV option, 27
ABSFTOL option, 27
ABSGCONV option, 27
ABSGTOL option, 27
ABSXCONV option, 27
ABSXTOL option, 27
FCONV option, 28
FCONV2 option, 28
FSIZE option, 28
CPANEL procedure, RESTRICT statement, 129
CPANEL procedure, SELECTION statement
CHOOS= option, 24
COMPETITIVE option, 24
CRIERION= option, 24
DETAILS= option, 25
HIERARCHY= option, 25
FTOL option, 28
FTOL2 option, 28
GCONV option, 29
GCONV2 option, 29
GTOL option, 29
GTOL2 option, 29
MAXFUNC= option, 29
MAXITER= option, 30
MINITER= option, 30
TECHNIQUE= option, 30
XCONV option, 30
XSIZE option, 31
XTOL option, 30
CQLIM procedure, SELECTION statement
  CHOOSE= option, 24
  COMPETITIVE option, 24
  CRITERION= option, 24
  DETAILS= option, 25
  HIERARCHY= option, 25
  MAXEFFECTS= option, 24
  MAXSTEPS= option, 24
  METHOD= option, 23
  MINEFFECTS= option, 24
  ORDERSELECT option, 26
  SELECT= option, 24
  SELECTION= option, 26
  STOP= option, 24
  STOPHORIZON= option, 26
CQLIM procedure, TEST statement, 186
CQLIM procedure, WEIGHT statement, 187
CRITERION= option
  PROC SEVSELECT statement, 218
  SELECTION statement (CCOPULA), 24
  SELECTION statement (CNTSELECT), 24
  SELECTION statement (CPANEL), 24
  SELECTION statement (CQLIM), 24
  SELECTION statement (SEVSELECT), 24
DATA= option
  PROC CNTSELECT statement, 77
  PROC CPANEL statement, 119
  PROC CQLIM statement, 178
  PROC SEVSELECT statement, 219
DATABOUNDARY option
  EFFECT statement, spline (CCOPULA), 15
  EFFECT statement, spline (CNTSELECT), 15
  EFFECT statement, spline (CPANEL), 15
  EFFECT statement, spline (CQLIM), 15
  EFFECT statement, spline (SEVSELECT), 15
DEFINE statement
  CCOPULA procedure, 59
DEGREE option
  EFFECT statement, polynomial (CCOPULA), 13
  EFFECT statement, polynomial (CNTSELECT), 13
  EFFECT statement, polynomial (CPANEL), 13
  EFFECT statement, polynomial (CQLIM), 13
  EFFECT statement, polynomial (SEVSELECT), 13
  EFFECT statement, spline (CCOPULA), 16
  EFFECT statement, spline (CNTSELECT), 16
  EFFECT statement, spline (CPANEL), 16
  EFFECT statement, spline (CQLIM), 16
  EFFECT statement, spline (SEVSELECT), 16
DESCENDING option
  CLASS statement (CCOPULA), 5
  CLASS statement (CNTSELECT), 5
  CLASS statement (CPANEL), 5
  CLASS statement (CQLIM), 5
  CLASS statement (SEVSELECT), 5
DETAILS option
  EFFECT statement, multimember (CCOPULA), 11
  EFFECT statement, multimember (CNTSELECT), 11
  EFFECT statement, multimember (CPANEL), 11
  EFFECT statement, multimember (CQLIM), 11
  EFFECT statement, multimember (SEVSELECT), 11
  EFFECT statement, polynomial (CCOPULA), 13
  EFFECT statement, polynomial (CNTSELECT), 13
  EFFECT statement, polynomial (CPANEL), 13
  EFFECT statement, polynomial (CQLIM), 13
  EFFECT statement, polynomial (SEVSELECT), 13
  EFFECT statement, spline (CCOPULA), 16
  EFFECT statement, spline (CNTSELECT), 16
  EFFECT statement, spline (CPANEL), 16
  EFFECT statement, spline (CQLIM), 16
  EFFECT statement, spline (SEVSELECT), 16
DETAILS= option
  SELECTION statement (CCOPULA), 25
  SELECTION statement (CNTSELECT), 25
  SELECTION statement (CPANEL), 25
  SELECTION statement (CQLIM), 25
  SELECTION statement (SEVSELECT), 25
DFMIXTURE= option
  SCALEMODEL statement (SEVSELECT), 237
DISCRETE option
  ENDOGENOUS statement (CQLIM), 180, 183
DISPERSION= option
  OUTPUT statement (CNTSELECT), 84
DISPLAY statement
  SEVSELECT procedure, 225
DISPLAYOUT statement
  SEVSELECT procedure, 226
DISPMODEL statement
  CNTSELECT procedure, 80
DIST statement
  SEVSELECT procedure, 226
DIST= option
  CNTSELECT statement (CNTSELECT), 82
  MODEL statement (CNTSELECT), 82
DISTRIBUTION= option
  ENDOGENOUS statement (CQLIM), 180, 183
EFFECT statement
  CCOPULA procedure, 8
  CNTSELECT procedure, 8
  collection effect, 10
  CPANEL procedure, 8
  CQLIM procedure, 8
  multimember effect, 10
  polynomial effect, 12
  SEVSELECT procedure, 8, 229
  spline effect, 15
EMPIRICALCDF= option
  PROC SEVSELECT statement, 221
ENDOGENOUS statement
  CPANEL procedure, 123
ERRORCOMP= option
  CNTSELECT statement (CNTSELECT), 82
  MODEL statement (CNTSELECT), 82
ERRSTD
  OUTPUT statement (CQLIM), 185
EXCLUDE option
  DISPLAY statement (SEVSELECT), 225
EXCLUDEALL option
  DISPLAY statement (SEVSELECT), 225
EXPECTED
  OUTPUT statement (CQLIM), 185
FCONV option
  PROC CCOPULA statement, 28
  PROC CNTSELECT statement, 28
  PROC CPANEL statement, 28
  PROC CQLIM statement, 28
  PROC SEVSELECT statement, 28
FCONV2 option
  PROC CCOPULA statement, 28
  PROC CNTSELECT statement, 28
  PROC CPANEL statement, 28
  PROC CQLIM statement, 28
  PROC SEVSELECT statement, 28
FDONE option
  MODEL statement (CPANEL), 126
FIXONE option
  MODEL statement (CPANEL), 126
FIXTWO option
  MODEL statement (CPANEL), 126
FREQ statement
  CNTSELECT procedure, 81
FRONTIER option
  ENDOGENOUS statement (CQLIM), 181, 184
FSIZE option
  PROC CCOPULA statement, 28
  PROC CNTSELECT statement, 28
  PROC CPANEL statement, 28
  PROC CQLIM statement, 28
  PROC SEVSELECT statement, 28
FTOL option
  PROC CCOPULA statement, 28
  PROC CNTSELECT statement, 28
  PROC CPANEL statement, 28
  PROC CQLIM statement, 28
  PROC SEVSELECT statement, 28
FTOL2 option
  PROC CCOPULA statement, 28
  PROC CNTSELECT statement, 28
  PROC CPANEL statement, 28
  PROC CQLIM statement, 28
  PROC SEVSELECT statement, 28
FUNCTIONS= option
  OUTPUT statement (SEVSELECT), 234
GCONV option
  PROC CCOPULA statement, 29
  PROC CNTSELECT statement, 29
  PROC CPANEL statement, 29
  PROC CQLIM statement, 29
  PROC SEVSELECT statement, 29
GCONV2 option
  PROC CCOPULA statement, 29
  PROC CNTSELECT statement, 29
  PROC CPANEL statement, 29
  PROC CQLIM statement, 29
  PROC SEVSELECT statement, 29
GDELTAY= option
  OUTPUT statement (CNTSELECT), 84
GROUPID= option
  PROC CNTSELECT statement, 77
GTOL option
  PROC CCOPULA statement, 29
  PROC CNTSELECT statement, 29
  PROC CPANEL statement, 29
  PROC CQLIM statement, 29
  PROC SEVSELECT statement, 29
GTOL2 option
  PROC CCOPULA statement, 29
  PROC CNTSELECT statement, 29
  PROC CPANEL statement, 29
  PROC CQLIM statement, 29
  PROC SEVSELECT statement, 29
HIERARCHY= option
SELECTION statement (CCOPULA), 25
SELECTION statement (CNTSELECT), 25
SELECTION statement (CPANEL), 25
SELECTION statement (CQLIM), 25
SELECTION statement (SEVSELECT), 25

HTAYLOR option
MODEL statement (CPANEL), 126

ID statement
CPANEL procedure, 124
INCLUDE option
SCALEMODEL statement (SEVSELECT), 238

INTEST= option
PROC SEVSELECT statement, 219

INFORMATIVE option
SCALEMODEL statement (SEVSELECT), 238

INFUNCDEF= option
DIST statement (SEVSELECT), 228

INIT statement
CNTSELECT procedure, 81
CQLIM procedure, 182

INIT= option
DIST statement (SEVSELECT), 227

INITSAMPLE option
PROC SEVSELECT statement, 223

INSTRUMENTS statement
CPANEL procedure, 124

ITPRINT option
CQLIM procedure, 178

IVBTWNG option
MODEL statement (CPANEL), 127

IVFIXONE option
MODEL statement (CPANEL), 127

IVPOOLED option
MODEL statement (CPANEL), 127

IVRANONE option
MODEL statement (CPANEL), 127

KNOTMAX= option
EFFECT statement, spline (CCOPULA), 16
EFFECT statement, spline (CNTSELECT), 16
EFFECT statement, spline (CPANEL), 16
EFFECT statement, spline (CQLIM), 16
EFFECT statement, spline (SEVSELECT), 16

KNOTMETHOD option
EFFECT statement, spline (CCOPULA), 16
EFFECT statement, spline (CNTSELECT), 16
EFFECT statement, spline (CPANEL), 16
EFFECT statement, spline (CQLIM), 16
EFFECT statement, spline (SEVSELECT), 16

KNOTMIN= option
EFFECT statement, spline (CCOPULA), 17
EFFECT statement, spline (CNTSELECT), 17
EFFECT statement, spline (CPANEL), 17

EFFECT statement, spline (CQLIM), 17
EFFECT statement, spline (SEVSELECT), 17

LAMBDA= option
OUTPUT statement (CNTSELECT), 84

LEFTCENSORED= option
LOSS statement (SEVSELECT), 231

LEFTTRUNCATED= option
LOSS statement (SEVSELECT), 231

LISTONLY option
DIST statement (SEVSELECT), 228

LM option
TEST statement (CNTSELECT), 86
TEST statement (CPANEL), 130
TEST statement (CQLIM), 187

LOSS statement
SEVSELECT procedure, 230

LOWERBOUND= option
ENDOGENOUS statement (CQLIM), 180, 184

LR option
TEST statement (CNTSELECT), 86
TEST statement (CPANEL), 130
TEST statement (CQLIM), 187

MARGINAL
OUTPUT statement (CQLIM), 185

MAXEFFECTS= option
SELECTION statement (CCOPULA), 24
SELECTION statement (CNTSELECT), 24
SELECTION statement (CPANEL), 24
SELECTION statement (CQLIM), 24
SELECTION statement (SEVSELECT), 24

MAXFUNC= option
PROC CCOPULA statement, 29
PROC CNTSELECT statement, 29
PROC CPANEL statement, 29
PROC CQLIM statement, 29
PROC SEVSELECT statement, 29

MAXITER= option
PROC CCOPULA statement, 29
PROC CNTSELECT statement, 29
PROC CPANEL statement, 29
PROC CQLIM statement, 29
PROC SEVSELECT statement, 29

MAXSTEPS= option
SELECTION statement (CCOPULA), 24
SELECTION statement (CNTSELECT), 24
SELECTION statement (CPANEL), 24
SELECTION statement (CQLIM), 24
SELECTION statement (SEVSELECT), 24

MAXTIME= option
PROC CCOPULA statement, 30
PROC CNTSELECT statement, 30
PROC CPANEL statement, 30
PROC CQLIM statement, 30
PROC SEVSELECT statement, 30

MDEGREE option
   EFFECT statement, polynomial (CCOPULA), 13
   EFFECT statement, polynomial (CNTSELECT), 13
   EFFECT statement, polynomial (CPANEL), 13
   EFFECT statement, polynomial (CQLIM), 13
   EFFECT statement, polynomial (SEVSELECT), 13

METHOD= option
   PROC CNTSELECT statement, 79
   SELECTION statement (CCOPULA), 23
   SELECTION statement (CNTSELECT), 23
   SELECTION statement (CPANEL), 23
   SELECTION statement (CQLIM), 23
   SELECTION statement (SEVSELECT), 23, 239

MILLS
   OUTPUT statement (CQLIM), 185

MINEFFECTS= option
   SELECTION statement (CCOPULA), 24
   SELECTION statement (CNTSELECT), 24
   SELECTION statement (CPANEL), 24
   SELECTION statement (CQLIM), 24
   SELECTION statement (SEVSELECT), 24

MINITER= option
   PROC CCOPULA statement, 30
   PROC CNTSELECT statement, 30
   PROC CPANEL statement, 30
   PROC CQLIM statement, 30
   PROC SEVSELECT statement, 30

MISSING option
   CLASS statement (CCOPULA), 5
   CLASS statement (CNTSELECT), 5
   CLASS statement (CPANEL), 5
   CLASS statement (CQLIM), 5
   CLASS statement (SEVSELECT), 5

MODEL= option
   OUTPUT statement (CNTSELECT), 84

MODEL statement
   CNTSELECT procedure, 81
   CPANEL procedure, 125
   CQLIM procedure, 183

MSTAT option
   COMPARE statement (CPANEL), 121

MU= option
   OUTPUT statement (CNTSELECT), 84

NATURALCUBIC option
   EFFECT statement, spline (CCOPULA), 17
   EFFECT statement, spline (CNTSELECT), 17
   EFFECT statement, spline (CPANEL), 17
   EFFECT statement, spline (CQLIM), 17
   EFFECT statement, spline (SEVSELECT), 17

NLOPTIONS statement
   SEVSELECT procedure, 233

NOCLPRINT option
   PROC SEVSELECT statement, 219

NOEFFECT option
   EFFECT statement, multimember (CCOPULA), 11
   EFFECT statement, multimember (CNTSELECT), 11
   EFFECT statement, multimember (CPANEL), 11
   EFFECT statement, multimember (CQLIM), 11
   EFFECT statement, multimember (SEVSELECT), 11

NOINT option
   MODEL statement (CNTSELECT), 82
   MODEL statement (CPANEL), 127
   MODEL statement (CQLIM), 183

NOLABEL option
   MODEL statement (CPANEL), 128

NONORMALIZE option
   WEIGHT statement (CNTSELECT), 87
   WEIGHT statement (CQLIM), 188

NOPART option
   PROC CPANEL statement, 120

NOPRINT option
   PROC CNTSELECT statement, 78, 83
   PROC CPANEL statement, 120
   PROC SEVSELECT statement, 219

NOREPLACE option
   DISPLAYOUT statement (SEVSELECT), 226

NOSEPARATE option
   EFFECT statement, polynomial (CCOPULA), 13
   EFFECT statement, polynomial (CNTSELECT), 13
   EFFECT statement, polynomial (CPANEL), 13
   EFFECT statement, polynomial (CQLIM), 13
   EFFECT statement, polynomial (SEVSELECT), 13

NU= option
   OUTPUT statement (CNTSELECT), 84

OFFSET= option
   MODEL statement (CNTSELECT), 82
   SCALEMODEL statement (SEVSELECT), 238

ORDER= option
   CLASS statement (CCOPULA), 5
   CLASS statement (CNTSELECT), 5
   CLASS statement (CPANEL), 5
   CLASS statement (CQLIM), 5
   CLASS statement (SEVSELECT), 5
   ENDGENOUS statement (CQLIM), 183

ORDERSELECT option
   SELECTION statement (CCOPULA), 26
   SELECTION statement (CNTSELECT), 26
SELECTION statement (CPANEL), 26
SELECTION statement (CQLIM), 26
SELECTION statement (SEVSELECT), 26

OUT= option
OUTPUT statement (CNTSELECT), 83
OUTPUT statement (CPANEL), 128
OUTPUT statement (CQLIM), 185
OUTPUT statement (SEVSELECT), 233

OUTTEST= option
PROC SEVSELECT statement, 219

OUTFUNCDEF= option
DIST statement (SEVSELECT), 228

OUTPARM= option
COMPARE statement (CPANEL), 122

OUTPUT statement
CNTSELECT procedure, 83
CPANEL procedure, 128
CQLIM procedure, 185
SEVSELECT procedure, 233

OUTSTAT= option
COMPARE statement (CPANEL), 122

P= option
OUTPUT statement (CPANEL), 129

PARAM= option
CLASS statement (CCOPULA), 5
CLASS statement (CNTSELECT), 5
CLASS statement (CPANEL), 5
CLASS statement (CQLIM), 5
CLASS statement (SEVSELECT), 5

PARAMETER= option
CNTSELECT statement (CNTSELECT), 82
MODEL statement (CNTSELECT), 82

POOLED option
MODEL statement (CPANEL), 127

PRED= option
OUTPUT statement (CNTSELECT), 84

PREDICTED
OUTPUT statement (CQLIM), 185
PREDICTED= option
OUTPUT statement (CPANEL), 129

PRINT= option
PROC SEVSELECT statement, 219

PRINTALL option
MODEL statement, 83
PROC CNTSELECT statement, 78
PROC CPANEL statement, 120

PRINTFIXED option
MODEL statement (CPANEL), 128

PRINTINTERNALNAMES option
PROC CNTSELECT statement, 78

PRINTTIMING option
PROC CCOPULA statement, 59
PROC CNTSELECT statement, 78
PROC CPANEL statement, 120

PROB
OUTPUT statement (CQLIM), 185
PROB= option
OUTPUT statement (CNTSELECT), 84

PROBALL
OUTPUT statement (CQLIM), 185

PROBCOUNT option
OUTPUT statement (CNTSELECT), 84

PROBOBSERVED= option
LOSS statement (SEVSELECT), 232

PROBZERO= option
OUTPUT statement (SEVSELECT), 232

PROC CCOPULA statement
CCOPULA procedure, 59

PROC CPANEL statement, see CPANEL procedure
PROC SEVSELECT statement, 218

PRODUCTION option
ENDOGENOUS statement (CQLIM), 181, 184

PSTAT option
COMPARE statement (CPANEL), 123

QUANTILES= option
OUTPUT statement (SEVSELECT), 235

R= option
OUTPUT statement (SEVSELECT), 235

RANONE option
MODEL statement (CPANEL), 127

RANTWO option
MODEL statement (CPANEL), 127

REF= option
CLASS statement (CCOPULA), 6
CLASS statement (CNTSELECT), 6
CLASS statement (CPANEL), 6
CLASS statement (CQLIM), 6
CLASS statement (SEVSELECT), 6

REPEATED option
DISPLAYOUT statement (SEVSELECT), 226

RESIDUAL
OUTPUT statement (CQLIM), 186
RESIDUAL= option
OUTPUT statement (CPANEL), 129

RESTRICT statement
CNTSELECT procedure, 85
CPANEL procedure, 129
CQLIM procedure, 186

RIGHTCENSORED= option
LOSS statement (SEVSELECT), 232

RIGHTTRUNCATED= option
LOSS statement (SEVSELECT), 232

SCALEMODEL statement
SEVSELECT procedure, 236

SELECT= option
SELECTION statement (CCOPULA), 24
SELECTION statement (CNTSELECT), 24
SELECTION statement (CPANEL), 24
SELECTION statement (CQLIM), 24
SELECTION statement (SEVSELECT), 24, 239

SELECTION statement
CCOPULA procedure, 23
CNTSELECT procedure, 23
CPANEL procedure, 23
CQLIM procedure, 23
SEVSELECT procedure, 23, 238

SELECTION= option
SELECTION statement (CCOPULA), 26
SELECTION statement (CNTSELECT), 26
SELECTION statement (CPANEL), 26
SELECTION statement (CQLIM), 26
SELECTION statement (SEVSELECT), 26

SELECTNLLOTTECH= option
PROC SEVSELECT statement, 224

SELECTOUT option
PROC SEVSELECT statement, 221

SEPARATE option
EFFECT statement, spline (CCOPULA), 17
EFFECT statement, spline (CNTSELECT), 17
EFFECT statement, spline (CPANEL), 17
EFFECT statement, spline (CQLIM), 17
EFFECT statement, spline (SEVSELECT), 17

SEVSELECT procedure, 214
DIST statement, 226
LOSS statement, 230
NLOPTIONS statement, 233
OUTPUT statement, 233
SCALEMODEL statement, 236
syntax, 214
WEIGHT statement, 240

SEVSELECT procedure, CLASS statement, 224
DESCENDING option, 5
MISSING option, 5
ORDER= option, 5
PARAM= option, 5
REF= option, 6
SPLIT option, 7

SEVSELECT procedure, DISPLAY statement
CASESENSITIVE option, 225
EXCLUDE option, 225
EXCLUDEALL option, 225
TRACE option, 225

SEVSELECT procedure, DISPLAYOUT statement
NOREPLACE option, 226
REPEATED option, 226

SEVSELECT procedure, DIST statement
INFUNCDEF= option, 228
INIT= option, 227
LISTONLY option, 228
OUTFUNCDEF= option, 228
VALIDATEONLY option, 228

SEVSELECT procedure, EFFECT statement
BASIS option (spline), 15
DATABOUNDARY option (spline), 15
DEGREE option (polynomial), 13
DEGREE option (spline), 16
DETAILS option (multimember), 11
DETAILS option (polynomial), 13
DETAILS option (spline), 16
KNOTMAX= option (spline), 16
KNOTMETHOD option (spline), 16
KNOTMIN= option (spline), 17
MDEGREE option (polynomial), 13
NATURALCUBIC option (spline), 17
NOEFFECT option (multimember), 11
NOSEPARATE option (polynomial), 13
SEPARATE option (spline), 17
SPLIT option (spline), 18
STANDARDIZE option (polynomial), 13

SEVSELECT procedure, LOSS statement
LEFTCENSORED= option, 231
LEFTTRUNCATED= option, 231
PROBOBSERVED= option, 232
RIGHTCENSORED= option, 232
RIGHTTRUNCATED= option, 232

SEVSELECT procedure, OUTPUT statement
COPYVARS= option, 233
FUNCTIONS= option, 234
OUT= option, 233
QUANTILES= option, 235

SEVSELECT procedure, PROC SEVSELECT statement, 218
ABS_CONV option, 27
ABS_FCONV option, 27
ABS_FTOL option, 27
ABS_GCONV option, 27
ABS_GTOL option, 27
ABS_TOL option, 27
ABS_XCONV option, 27
ABS_XTOL option, 27
COV_OUT option, 218
CRI_TERION= option, 218
DATA= option, 219
EDF=AUTO option, 221
EDF=KAPLANMEIER option, 221
EDF=MODIFIEDKM option, 221
EDF=NOTURNBULL option, 222
EDF=STANDARD option, 222
EDF=TURNBULL option, 222
EMPIRICALCDF= option, 221
FCONV option, 28
FCONV2 option, 28
FSIZE option, 28
FTOL option, 28
FTOL2 option, 28
GCONV option, 29
GCONV2 option, 29
GTOL option, 29
GTOL2 option, 29
INEST= option, 219
INITSAMPLE option, 223
MAXFUNC= option, 29
MAXITER= option, 29
MAXTIME= option, 30
MINITER= option, 30
NOCLPRINT option, 219
NOPRINT option, 219
OUTEST= option, 219
PRINT= option, 219
SELECTNLOTECH= option, 224
SELECTION option, 221
TECHNIQUE= option, 30
VARDEF= option, 221
XALIGN option, 31
XTOL option, 30
ZEROEST option, 221
SEVSELECT procedure, SCALEMODEL statement
DFMIXTURE= option, 237
INCLUDE option, 238
INFORMATIVE option, 238
OFFSET= option, 238
START option, 238
SEVSELECT procedure, SELECTION statement, 238
CHOOSE= option, 24, 239
COMPETITIVE option, 24
CRITERION= option, 24
DETAILS= option, 25
HIERARCHY= option, 25
MAXEFFECTS= option, 24
MAXSTEPS= option, 24
METHOD= option, 23, 239
MINEFFECTS= option, 24
ORDERSELECT option, 26
SELECT= option, 24, 239
SELECTION= option, 26
STOP= option, 24, 240
STOPHORIZON= option
SELECTION statement (CCOPULA), 24
SELECTION statement (CNTSELECT), 24
SELECTION statement (CPANEL), 24
SELECTION statement (CQLIM), 24
SELECTION statement (SEVSELECT), 24, 240
TE1
OUTPUT statement (CQLIM), 186
TE2
OUTPUT statement (CQLIM), 186
TECHNIQUE= option
PROC CCOPULA statement, 30
PROC CNTSELECT statement, 30
PROC CPANEL statement, 30
PROC CQLIM statement, 30
PROC SEVSELECT statement, 30
TEST statement
CPANEL procedure, 129
TRACE option
DISPLAY statement (SEVSELECT), 225
TRUNCATED option
ENDOGENOUS statement (CQLIM), 180, 184
UPPERBOUND= option
ENDOGENOUS statement (CQLIM), 180, 184
VALIDATEONLY option
DIST statement (SEVSELECT), 228
VAR Statement
CCOPULA procedure, 60
VARDEF= option
PROC SEVSELECT statement, 221
VARIANCE= option
OUTPUT statement (CNTSELECT), 84
VCOMP= option
MODEL statement (CPANEL), 127

WALD option
  TEST statement (CNTSELECT), 86
  TEST statement (CPANEL), 130
  TEST statement (CQLIM), 187

WEIGHT statement
  SEVSELECT procedure, 240

XBETA
  OUTPUT statement (CQLIM), 186

XBETA= option
  OUTPUT statement (CNTSELECT), 83

XCONV option
  PROC CCOPULA statement, 30
  PROC CNTSELECT statement, 30
  PROC CPANEL statement, 30
  PROC CQLIM statement, 30
  PROC SEVSELECT statement, 30

XSIZE option
  PROC CCOPULA statement, 31
  PROC CNTSELECT statement, 31
  PROC CPANEL statement, 31
  PROC CQLIM statement, 31
  PROC SEVSELECT statement, 31

XTOL option
  PROC CCOPULA statement, 30
  PROC CNTSELECT statement, 30
  PROC CPANEL statement, 30
  PROC CQLIM statement, 30
  PROC SEVSELECT statement, 30

ZEROEST option
  PROC SEVSELECT statement, 221

ZEROMODEL statement
  CNTSELECT procedure, 87

ZGAMMA= option
  OUTPUT statement (CNTSELECT), 84
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.