SAS/ETS® 15.1 User's Guide

High-Performance Procedures

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2018. *SAS/ETS® 15.1 User's Guide: High-Performance Procedures*. Cary, NC: SAS Institute Inc.

SAS/ETS® 15.1 User's Guide: High-Performance Procedures

SAS Institute Inc., SAS Campus Drive, Cary, NC 27513-2414

November 2018

15.1-P1:etshpug

Contents

Credits and Acknowledgments

Credits

Documentation

Editing	Anne Baxter, Ed Huddleston
Documentation Support	Tim Arnold

Software

The procedures in this book were was implemented by the following members of the development staff. Program development includes design, programming, debugging, support, and documentation. In the following list, the names of the developers who currently provide primary support are listed first; other developers and previous developers are also listed.

HPCDM	Mahesh Joshi, Richard Potter, Jan Chvosta
HPCOPULA	Richard Potter, Hao Chen, Jan Chvosta
HPCOUNTREG	Richard Potter, Jan Chvosta
HPPANEL	Bobby Gutierrez, Richard Potter, Linxia Ren
HPQLIM	Christian Macaro, Richard Potter, Jan Chvosta
HPSEVERITY	Mahesh Joshi
High-performance computing foundation	Steve E. Krueger
High-performance analytics foundation	Mary Carter, Robert Cohen, Vino Gona, Georges H. Guirguis, Trevor Kearney, Richard Knight, Gang Meng, Scott Pope, Oliver Schabenberger, Charles Shorb, Tom P. Weber, Yongqiao Xiao
Numerical routines	Georges H. Guirguis, Scott Pope

The following people have contributed with their leadership and support: Chris Bailey, Tonya Balan, David Pope, Oliver Schabenberger, Renee Sciortino, jonathan Wexler.

Testing

Tim Carter, Ming Chun-Chang, Bruce Elsheimer, Girija Gavankar, Sanggohn Han, Sylvie Kabisa, Jim McKenzie, Dright Ho, Cheryl LeSaint, Jim Metcalf, Bengt Pederson, Oleksiy Tokovenko

Internationalization Testing

Mi-Na Chu, Jacky Dong, Masayuki Iizuka, David Li, Lan Luan, Haiyong Rong, Felix Sha, Bin Sun, Frank Wang, Lina Xu

Technical Support

Rob Agnelli, Wen Bardsley, Phil Gibbs, David Schlotzhauer, Jill Tao, Donna Woodward

Acknowledgments

Many people make significant and continuing contributions to the development of SAS software products.

The final responsibility for the SAS System lies with SAS alone. We hope that you will always let us know your opinions about the SAS System and its documentation. It is through your participation that SAS software is continuously improved.

Chapter 1
Introduction

Contents

Overview of SAS/ETS High-Performance Procedures

SAS/ETS high-performance procedures provide econometric modeling tools that have been specially developed to take advantage of parallel processing in both multithreaded single-machine mode and distributed multiple-machine mode. Econometric modeling methods include regression for count data, models for the severity of losses or other events, and regression models for qualitative and limited dependent variables.

In addition to the high-performance econometric procedures described in this book, SAS/ETS includes high-performance utility procedures, which are described in *Base SAS Procedures Guide: High-Performance Procedures*. You can run all these procedures in single-machine mode without licensing SAS High-Performance Econometrics. However, to run these procedures in distributed mode, you must license SAS High-Performance Econometrics.

About This Book

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

Chapter Organization

This book is organized as follows:

Chapter 1, this chapter, provides an overview of SAS/ETS high-performance procedures.

Chapter 2, "Shared Concepts and Topics," describes the modes in which SAS/ETS high-performance procedures can execute.

Subsequent chapters describe the individual procedures. These chapters appear in alphabetical order by procedure name. Each chapter is organized as follows:

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

Typographical Conventions

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

roman	is the standard type style used for most text.
UPPERCASE ROMAN	is used for SAS statements, options, and other SAS language elements when they appear in the text. However, you can enter these elements in your own SAS programs in lowercase, uppercase, or a mixture of the two.
UPPERCASE BOLD	is used in the "Syntax" sections' initial lists of SAS statements and options.
oblique	is used in the syntax definitions and in text to represent arguments for which you supply a value.
VariableName	is used for the names of variables and data sets when they appear in the text.
bold	is used to for matrices and vectors.
italic	is used for terms that are defined in the text, for emphasis, and for references to publications.
`monospace`	is used for example code. In most cases, this book uses lowercase type for SAS code.

Options Used in Examples

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

```
ods html style=HTMLBlue;
. . .
ods html close;

ods pdf style=PearlJ;
. . .
ods pdf close;
```

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

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

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

SAS Technical Support Services

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

Chapter 2
Shared Concepts and Topics

Contents

Overview

This chapter describes the modes of execution in which SAS high-performance analytical procedures can execute. If you have SAS/ETS installed, you can run any procedure in this book on a single machine. However, to run procedures in this book in distributed mode, you must also have SAS High-Performance Econometrics software installed. For more information about these modes, see the next section.

This chapter provides details of how you can control the modes of execution and includes the syntax for the PERFORMANCE statement, which is common to all high-performance analytical procedures.

Processing Modes

Single-Machine Mode

Single-machine mode is a computing model in which multiple processors or multiple cores are controlled by a single operating system and can access shared resources, such as disks and memory. In this book, single-machine mode refers to an application running multiple concurrent threads on a multicore machine in order to take advantage of parallel execution on multiple processing units. More simply, single-machine mode for high-performance analytical procedures means multithreading on the client machine.

All high-performance analytical procedures are capable of running in single-machine mode, and this is the default mode when a procedure runs on the client machine. The procedure uses the number of CPUs (cores) on the machine to determine the number of concurrent threads. High-performance analytical procedures use different methods to map core count to the number of concurrent threads, depending on the analytic task. Using one thread per core is not uncommon for the procedures that implement data-parallel algorithms.

Distributed Mode

Distributed mode is a computing model in which several nodes in a distributed computing environment participate in the calculations. In this book, the distributed mode of a high-performance analytical procedure refers to the procedure performing the analytics on an appliance that consists of a cluster of nodes. This appliance can be one of the following:

- a database management system (DBMS) appliance on which the SAS High-Performance Analytics infrastructure is also installed
- a cluster of nodes that have the SAS High-Performance Analytics infrastructure installed but no DBMS software installed

Controlling the Execution Mode with Environment Variables and Performance Statement Options

You control the execution mode by using environment variables or by specifying options in the PERFORMANCE statement in high-performance analytical procedures, or by a combination of these methods.

The important environment variables follow:

- *grid host* identifies the domain name system (DNS) or IP address of the appliance node to which the SAS High-Performance Econometrics software connects to run in distributed mode.
- *installation location* identifies the directory where the SAS High-Performance Econometrics software is installed on the appliance.

You can set an environment variable directly from the SAS program by using the OPTION SET= command. For example, the following statements define the grid host and the location where the SAS High-Performance software is installed on the appliance:

```
option set=GRIDHOST      ="hpa.sas.com";
option set=GRIDINSTALLLOC="/opt/TKGrid";
```

Alternatively, you can set the parameters in the PERFORMANCE statement in high-performance analytical procedures. For example:

```
performance host      ="hpa.sas.com"
            install   ="/opt/TKGrid";
```

A specification in the PERFORMANCE statement overrides a specification of an environment variable without resetting its value. An environment variable that you set in the SAS session by using an OPTION SET= command remains in effect until it is modified or until the SAS session terminates.

The key variable that determines whether a high-performance analytical procedure executes in single-machine or distributed mode is the *grid host*. The installation location is needed to ensure that a connection to the grid host can be made, given that a host is specified. This book assumes that the installation location has been set by your system administrator.

The following sets of SAS statements are functionally equivalent:

```
proc hpreduce;
   reduce unsupervised x:;
   performance host="hpa.sas.com";
run;

option set=GRIDHOST="hpa.sas.com";
proc hpreduce;
   reduce unsupervised x:;
run;
```

Determining Single-Machine Mode or Distributed Mode

High-performance analytical procedures use the following rules to determine whether they run in single-machine mode or distributed mode:

- If a grid host is not specified, the analysis is carried out in single-machine mode on the client machine that runs the SAS session.
- If a grid host is specified, the behavior depends on whether the execution is alongside the database or alongside HDFS. If the data are local to the client (that is, not stored in the distributed database or HDFS on the appliance), you need to use the NODES= option in the PERFORMANCE statement to specify the number of nodes on the appliance or cluster that you want to engage in the analysis. If the procedure executes alongside the database or alongside HDFS, you do not need to specify the NODES= option.

The following example shows single-machine and client-data distributed configurations for a data set of 100,000 observations that are simulated from a logistic regression model. The following DATA step generates the data:

```
data simData;
   array _a{8} _temporary_ (0,0,0,1,0,1,1,1);
   array _b{8} _temporary_ (0,0,1,0,1,0,1,1);
   array _c{8} _temporary_ (0,1,0,0,1,1,0,1);
   do obsno=1 to 100000;
      x  = rantbl(1,0.28,0.18,0.14,0.14,0.03,0.09,0.08,0.06);
      a  = _a{x};
      b  = _b{x};
      c  = _c{x};
      x1 = int(ranuni(1)*400);
      x2 = 52 + ranuni(1)*38;
      x3 = ranuni(1)*12;
      lp = 6. -0.015*(1-a) + 0.7*(1-b) + 0.6*(1-c) + 0.02*x1 -0.05*x2 - 0.1*x3;
      y  = ranbin(1,1,(1/(1+exp(lp))));
      output;
   end;
   drop x lp;
run;
```

The following statements run PROC HPLOGISTIC to fit a logistic regression model:

```
proc hplogistic data=simData;
   class a b c;
   model y = a b c x1 x2 x3;
run;
```

Figure 2.1 shows the results from the analysis.

Figure 2.1 Results from Logistic Regression in Single-Machine Mode

The HPLOGISTIC Procedure

Performance Information	
Execution Mode	Single-Machine
Number of Threads	4

Data Access Information			
Data	**Engine**	**Role**	**Path**
WORK.SIMDATA	V9	Input	On Client

Model Information	
Data Source	WORK.SIMDATA
Response Variable	y
Class Parameterization	GLM
Distribution	Binary
Link Function	Logit
Optimization Technique	Newton-Raphson with Ridging

Parameter Estimates

Parameter	Estimate	Standard Error	DF	t Value	Pr > |t|
Intercept	5.7011	0.2539	Infty	22.45	<.0001
a 0	-0.01020	0.06627	Infty	-0.15	0.8777
a 1	0	.	.	.	.
b 0	0.7124	0.06558	Infty	10.86	<.0001
b 1	0	.	.	.	.
c 0	0.8036	0.06456	Infty	12.45	<.0001
c 1	0	.	.	.	.
x1	0.01975	0.000614	Infty	32.15	<.0001
x2	-0.04728	0.003098	Infty	-15.26	<.0001
x3	-0.1017	0.009470	Infty	-10.74	<.0001

The entries in the "Performance Information" table show that the HPLOGISTIC procedure runs in single-machine mode and uses four threads, which are chosen according to the number of CPUs on the client machine. You can force a certain number of threads on any machine that is involved in the computations by specifying the NTHREADS option in the PERFORMANCE statement. Another indication of execution on the client is the following message, which is issued in the SAS log by all high-performance analytical procedures:

```
NOTE: The HPLOGISTIC procedure is executing in single-machine mode.
```

The following statements use 10 nodes (in distributed mode) to analyze the data on the appliance; results appear in Figure 2.2:

```
proc hplogistic data=simData;
   class a b c;
   model y = a b c x1 x2 x3;
   performance host="hpa.sas.com" nodes=10;
run;
```

Figure 2.2 Results from Logistic Regression in Distributed Mode

The HPLOGISTIC Procedure

Performance Information	
Host Node	hpa.sas.com
Execution Mode	Distributed
Number of Compute Nodes	10
Number of Threads per Node	24

Data Access Information			
Data	**Engine**	**Role**	**Path**
WORK.SIMDATA	V9	Input	From Client

Model Information	
Data Source	WORK.SIMDATA
Response Variable	y
Class Parameterization	GLM
Distribution	Binary
Link Function	Logit
Optimization Technique	Newton-Raphson with Ridging

Parameter Estimates

Parameter	Estimate	Standard Error	DF	t Value	Pr > |t|
Intercept	5.7011	0.2539	Infty	22.45	<.0001
a 0	-0.01020	0.06627	Infty	-0.15	0.8777
a 1	0	.	.	.	.
b 0	0.7124	0.06558	Infty	10.86	<.0001
b 1	0	.	.	.	.
c 0	0.8036	0.06456	Infty	12.45	<.0001
c 1	0	.	.	.	.
x1	0.01975	0.000614	Infty	32.15	<.0001
x2	-0.04728	0.003098	Infty	-15.26	<.0001
x3	-0.1017	0.009470	Infty	-10.74	<.0001

The specification of a host causes the "Performance Information" table to display the name of the host node of the appliance. The "Performance Information" table also indicates that the calculations were performed in a distributed environment on the appliance. Twenty-four threads on each of 10 nodes were used to perform the calculations—for a total of 240 threads.

Another indication of distributed execution on the appliance is the following message, which is issued in the SAS log by all high-performance analytical procedures:

```
NOTE: The HPLOGISTIC procedure is executing in the distributed
      computing environment with 10 worker nodes.
```

You can override the presence of a grid host and force the computations into single-machine mode by specifying the NODES=0 option in the PERFORMANCE statement:

```
proc hplogistic data=simData;
   class a b c;
   model y = a b c x1 x2 x3;
   performance host="hpa.sas.com" nodes=0;
run;
```

Figure 2.3 shows the "Performance Information" table. The numeric results are not reproduced here, but they agree with the previous analyses, which are shown in Figure 2.1 and Figure 2.2.

Figure 2.3 Single-Machine Mode Despite Host Specification

The HPLOGISTIC Procedure

Performance Information	
Execution Mode	Single-Machine
Number of Threads	4

Data Access Information			
Data	**Engine**	**Role**	**Path**
WORK.SIMDATA	V9	Input	On Client

The "Performance Information" table indicates that the HPLOGISTIC procedure executes in single-machine mode on the client. This information is also reported in the following message, which is issued in the SAS log:

```
NOTE: The HPLOGISTIC procedure is executing in single-machine mode.
```

In the analysis shown previously in Figure 2.2, the data set Work.simData is local to the client, and the HPLOGISTIC procedure distributed the data to 10 nodes on the appliance. The High-Performance Analytics infrastructure does not keep these data on the appliance. When the procedure terminates, the in-memory representation of the input data on the appliance is freed.

When the input data set is large, the time that is spent sending client-side data to the appliance might dominate the execution time. In practice, transfer speeds are usually lower than the theoretical limits of the network connection or disk I/O rates. At a transfer rate of 40 megabytes per second, sending a 10-gigabyte data set to the appliance requires more than four minutes. If analytic execution time is in the range of seconds, the "performance" of the process is dominated by data movement.

The alongside-the-database execution model, unique to high-performance analytical procedures, enables you to read and write data in distributed form from the database that is installed on the appliance.

Data Access Modes

Single-Machine Data Access Mode

When high-performance analytical procedures run in single-machine mode, they access data in the same way as traditional SAS procedures. They use Base SAS to access input and output SAS data sets on the

client machine, and they use the relevant SAS/ACCESS interface to bring data from other sources, such as third-party databases, Hadoop, and SAS LASR servers, to the client.

Distributed Data Access Mode

When high-performance analytical procedures run in distributed mode, input data must be brought to the computation that is performed on the nodes of the grid, and output data must be sent from the computational nodes. This can be accomplished in several ways:

- Client-data (local-data) mode: The input and output data for the analytic task are stored on the client machine where the high-performance procedure is invoked. When the procedure runs, the SAS High-Performance Analytics infrastructure sends input data from the client to the distributed computing environment and sends output data from the distributed computing environment to the client.
- Parallel symmetric mode: Input and output data are stored on the same nodes that are used for the distributed computation, and the data move in parallel from the data store to the computational nodes without crossing node boundaries. Parallel symmetric mode is available with the following distributed data sources:
  - Data in Greenplum databases that are collocated with the computational nodes. This access mode is also called alongside-the-database mode. For more information, see the section "Alongside-the-Database Execution" on page 14.
  - Data in SASHDAT format in the Hadoop Distributed File System (HDFS) that is collocated with the computational nodes. This access mode is also called alongside-HDFS mode. For more information, see the section "Alongside-HDFS Execution by Using the SASHDAT Engine" on page 23.
  - Data in a SAS LASR Analytic Server that is collocated with the computational nodes. This access mode is also called alongside-LASR mode. For more information, see the section "Running High-Performance Analytical Procedures Alongside a SAS LASR Analytic Server in Distributed Mode" on page 17.
- Parallel asymmetric mode: The primary reason for providing this mode is to enable you to manage and house data on appliances (the data appliances) and to run high-performance analytical procedures on a different appliance (the computing appliance). The high-performance analytical procedures run in a SAS process on the computing appliance. For each data source that is accessed in parallel asymmetric mode, a SAS Embedded Process must run on the associated data appliance. Data are requested by a SAS data feeder that runs on the computing appliance and communicates with the SAS Embedded Process on the data appliance. The SAS Embedded Process transfers the data in parallel to the SAS data feeder that runs on each of the nodes of the computing appliance. This mode is called asymmetric mode because the number of nodes on the data appliance does not need to match the number of nodes on the computing appliance. Parallel asymmetric mode is supported for data in Teradata, Greenplum, and Oracle databases and for data in HDFS and SAP HANA. In these cases, the parallel asymmetric access is somewhat loosely described as being asymmetric alongside access, even though the data storage and computation can occur on different appliances. For more information, see the section "Running High-Performance Analytical Procedures in Asymmetric Mode" on page 20.

- Through-the-client mode: When data can be accessed through a SAS/ACCESS interface but the data reside in a file system or in a distributed data source on which a SAS Embedded Process is not running, those data cannot be accessed in parallel in either symmetric or asymmetric mode. The SAS/ACCESS interface is used to transfer input data from the data source to the client machine on which the high-performance procedure is invoked, and the data are then sent to the distributed computing environment by the SAS High-Performance Analytics infrastructure. The data path is reversed for output data. This mode of data access is referred to as through-the-client access.

Determining the Data Access Mode

High-performance analytical procedures determine the data access mode individually for each data set that is used in the analysis. When high-performance analytical procedures run in distributed mode, parallel symmetric or parallel asymmetric mode is used whenever possible. There are two reasons why parallel access might not be possible. The first reason is that for a particular data set, the required SAS Embedded Process is not installed on the appliance that houses the data. In such cases, access to those data reverts to through-the-client access, and a note like the following is reported in the SAS log:

```
NOTE: The data MYLIB.MYDATA are being routed through the client because a
      SAS Embedded Process is not running on the associated data server.
```

The second reason why parallel data access might not be possible for a particular data set is that the required driver software might not be installed on the compute nodes. In this case, the required data feeder that moves the data from the compute nodes to the data source cannot be successfully loaded, and a note like the following is reported in the SAS log:

```
NOTE: The data MYLIB.MYDATA are being routed through the client because
      the ORACLE data feeder could not be loaded on the specified grid host.
```

For distributed data in SASHDAT format in HDFS or data in a SAS LASR Analytic Server, parallel symmetric access is used when the data nodes and compute nodes are collocated on the same appliance. For data in a LASR Analytic Server that cannot be accessed in parallel symmetric mode, through-the-client mode is used. Through-the-client access is not supported for data in SASHDAT format in HDFS.

For data in Greenplum databases, parallel symmetric access is used if the compute nodes and the data nodes are collocated on the same appliance and you do not specify the NODES=*n* option in a PERFORMANCE statement. In this case, the number of nodes that are used is determined by the number of nodes across which the data are distributed. If you specify NODES=*n*, then parallel asymmetric access is used.

High-performance analytical procedures produce a “Data Access Information” table that shows you how each data set that is used in the analysis is accessed. The following statements provide an example in which PROC HPDS2 is used to copy a distributed data set named Neuralgia (which is stored in SASHDAT format in HDFS) to a SAS data set on the client machine:

```
libname hdatlib sashdat
                host='hpa.sas.com';
                hdfs_path="/user/hps";

proc hpds2 data=hdatlib.neuralgia out=neuralgia;
```

```
    performance host='hpa.sas.com';
    data DS2GTF.out;
       method run();
      set DS2GTF.in;
    end;
  enddata;
run;
```

Figure 2.4 shows the output that PROC HPDS2 produces. The "Performance Information" table shows that PROC HPDS2 ran in distributed mode on a 13-node grid. The "Data Access Information" table shows that the input data were accessed in parallel symmetric mode and the output data set was sent to the client, where the V9 (base) engine stored it as a SAS data set in the Work directory.

Figure 2.4 Performance Information and Data Access Information Tables

The HPDS2 Procedure

Performance Information	
Host Node	hpa.sas.com
Execution Mode	Distributed
Number of Compute Nodes	13
Number of Threads per Node	24

Data Access Information			
Data	**Engine**	**Role**	**Path**
HDATLIB.NEURALGIA	SASHDAT	Input	Parallel, Symmetric
WORK.NEURALGIA	V9	Output	To Client

Alongside-the-Database Execution

High-performance analytical procedures interface with the distributed database management system (DBMS) on the appliance in a unique way. If the input data are stored in the DBMS and the grid host is the appliance that houses the data, high-performance analytical procedures create a distributed computing environment in which an analytic process is collocated with the nodes of the DBMS. Data then pass from the DBMS to the analytic process on each node. Instead of moving across the network and possibly back to the client machine, the data pass locally between the processes on each node of the appliance.

Because the analytic processes on the appliance are separate from the database processes, the technique is referred to as alongside-the-database execution in contrast to in-database execution, where the analytic code executes in the database process.

In general, when you have a large amount of input data, you can achieve the best performance from high-performance analytical procedures if execution is alongside the database.

Before you can run alongside the database, you must distribute the data to the appliance. The following statements use the HPDS2 procedure to distribute the data set Work.simData into the mydb database on the hpa.sas.com appliance. In this example, the appliance houses a Greenplum database.

```
option set=GRIDHOST="green.sas.com";
libname applianc greenplm
        server  ="green.sas.com"
        user    =XXXXXX
        password=YYYYY
        database=mydb;


option set=GRIDHOST="compute_appliance.sas.com";

proc datasets lib=applianc nolist; delete simData;
proc hpds2 data=simData
           out =applianc.simData(distributed_by='distributed randomly');
  performance commit=10000 nodes=8;
  data DS2GTF.out;
     method run();
        set DS2GTF.in;
     end;
  enddata;
run;
```

If the output table applianc.simData exists, the DATASETS procedure removes the table from the Greenplum database because a DBMS does not usually support replacement operations on tables.

Note that the libref for the output table points to the appliance. The data set option informs the HPDS2 procedure to distribute the records randomly among the data segments of the appliance. The statements that follow the PERFORMANCE statement are the DS2 program that copies the input data to the output data without further transformations.

Because you loaded the data into a database on the appliance, you can use the following HPLOGISTIC statements to perform the analysis on the appliance in the alongside-the-database mode. These statements are almost identical to the first PROC HPLOGISTIC example in a previous section, which executed in single-machine mode.

```
proc hplogistic data=applianc.simData;
   class a b c;
   model y = a b c x1 x2 x3;
run;
```

The subtle differences are as follows:

- The grid host environment variable that you specified in an OPTION SET= command is still in effect.
- The DATA= option in the high-performance analytical procedure uses a libref that identifies the data source as being housed on the appliance. This libref was specified in a prior LIBNAME statement.

Figure 2.5 shows the results from this analysis. The "Performance Information" table shows that the execution was in distributed mode, and the "Data Access Information" table shows that the data were accessed asymmetrically in parallel from the Greenplum database. The numeric results agree with the previous analyses, which are shown in Figure 2.1 and Figure 2.2.

Figure 2.5 Alongside-the-Database Execution on Greenplum

The HPLOGISTIC Procedure

Performance Information	
Host Node	compute_appliance.sas.com
Execution Mode	Distributed
Number of Compute Nodes	139
Number of Threads per Node	40

Data Access Information			
Data	**Engine**	**Role**	**Path**
APPLIANC.SIMDATA	GREENPLM	Input	Parallel, Asymmetric

Model Information	
Data Source	APPLIANC.SIMDATA
Response Variable	y
Class Parameterization	GLM
Distribution	Binary
Link Function	Logit
Optimization Technique	Newton-Raphson with Ridging

Parameter Estimates

Parameter	Estimate	Standard Error	DF	t Value	Pr > |t|
Intercept	5.7011	0.2539	Infty	22.45	<.0001
a 0	-0.01020	0.06627	Infty	-0.15	0.8777
a 1	0	.	.	.	.
b 0	0.7124	0.06558	Infty	10.86	<.0001
b 1	0	.	.	.	.
c 0	0.8036	0.06456	Infty	12.45	<.0001
c 1	0	.	.	.	.
x1	0.01975	0.000614	Infty	32.15	<.0001
x2	-0.04728	0.003098	Infty	-15.26	<.0001
x3	-0.1017	0.009470	Infty	-10.74	<.0001

Alongside-LASR Distributed Execution

You can execute high-performance analytical procedures in distributed mode alongside a SAS LASR Analytic Server. When high-performance analytical procedures run in this mode, the data are preloaded in distributed form in memory that is managed by a LASR Analytic Server. The data on the nodes of the appliance are accessed in parallel in the process that runs the LASR Analytic Server, and they are transferred to the process where the high-performance analytical procedure runs. In general, each high-performance analytical procedure copies the data to memory that persists only while that procedure executes. Hence, when a high-performance analytical procedure runs alongside a LASR Analytic Server, both the high-performance analytical procedure and the LASR Analytic Server have a copy of the subset of the data that is used by the high-performance analytical procedure. The advantage of running high-performance analytical procedures alongside a LASR Analytic Server (as opposed to running alongside a DBMS table or alongside HDFS) is

that the initial transfer of data from the LASR Analytic Server to the high-performance analytical procedure is a memory-to-memory operation that is faster than the disk-to-memory operation when the procedure runs alongside a DBMS or HDFS. When the cost of preloading a table into a LASR Analytic Server is amortized by multiple uses of these data in separate runs of high-performance analytical procedures, using the LASR Analytic Server can result in improved performance.

Running High-Performance Analytical Procedures Alongside a SAS LASR Analytic Server in Distributed Mode

This section provides an example of steps that you can use to start and load data into a SAS LASR Analytic Server instance and then run high-performance analytical procedures alongside this LASR Analytic Server instance.

Starting a SAS LASR Analytic Server Instance

The following statements create a SAS LASR Analytic Server instance and load it with the simData data set that is used in the preceding examples. The data that are loaded into the LASR Analytic Server persist in memory across procedure boundaries until these data are explicitly deleted or until the server instance is terminated.

```
proc lasr port=54545
          data=simData
          path="/tmp/";
   performance host="hpa.sas.com" nodes=ALL;
run;
```

The PORT= option specifies a network port number to use. The PATH= option specifies the directory in which the server and table signature files are to be stored. The specified directory must exist on each machine in the cluster. The DATA= option specifies the name of a data set that is loaded into this LASR Analytic Server instance. (You do not need to specify the DATA= option at this time because you can add tables to the LASR Analytic Server instance at any stage of its life.) For more information about starting and using a LASR Analytic Server, see the *SAS LASR Analytic Server: Administration Guide*.

The NODES=ALL option in the PERFORMANCE statement specifies that the LASR Analytic Server run on all the nodes on the appliance. You can start a LASR Analytic Server on a subset of the nodes on an appliance, but this might affect whether high-performance analytical procedures can run alongside the LASR Analytic Server. For more information, see the section "Alongside-LASR Distributed Execution on a Subset of the Appliance Nodes" on page 19.

Figure 2.6 shows the "Performance Information" and "Data Access Information" tables, which show that the LASR procedure ran in distributed mode on 13 nodes and that the data were sent from the client to the appliance.

Figure 2.6 Performance and Data Access Information

The LASR Procedure

Performance Information	
Host Node	hpa.sas.com
Execution Mode	Distributed
Number of Compute Nodes	13

Data Access Information			
Data	**Engine**	**Role**	**Path**
WORK.SIMDATA	V9	Input	From Client

Associating a SAS Libref with the SAS LASR Analytic Server Instance

The following statements use a LIBNAME statement that associates a SAS libref (named MyLasr) with tables on the server instance as follows:

```
libname MyLasr sasiola port=54545 host="hpa.sas.com";
```

The SASIOLA option requests that the MyLasr libref use the SASIOLA engine, and the PORT= value associates this libref with the appropriate server instance. For more information about creating a libref that uses the SASIOLA engine, see the *SAS LASR Analytic Server: Administration Guide*.

Running a High-Performance Analytical Procedure Alongside the SAS LASR Analytic Server Instance

You can use the MyLasr libref to specify the input data for high-performance analytical procedures. You can also create output data sets in the SAS LASR Analytic Server instance by using this libref to request that the output data set be held in memory by the server instance as follows:

```
proc hplogistic data=MyLasr.simData;
   class a b c;
   model y = a b c x1 x2 x3;
   output out=MyLasr.simulateScores pred=PredictedProbabliity;
run;
```

Because you previously specified the GRIDHOST= environment variable and the input data are held in distributed form in the associated server instance, this PROC HPLOGISTIC step runs in distributed mode alongside the LASR Analytic Server, as indicated in the "Performance Information" table shown in Figure 2.7.

Figure 2.7 Performance and Data Access Information

The HPLOGISTIC Procedure

Performance Information	
Host Node	hpa.sas.com
Execution Mode	Distributed
Number of Compute Nodes	13
Number of Threads per Node	24

Data Access Information			
Data	**Engine**	**Role**	**Path**
MYLASR.SIMDATA	SASIOLA	Input	Parallel, Symmetric
MYLASR.SIMULATESCORES	SASIOLA	Output	Parallel, Symmetric

The "Data Access Information" table shows that both the input and output data were read and written, respectively, in parallel symmetric mode.

The preceding OUTPUT statement creates an output table that is added to the LASR Analytic Server instance. Output data sets do not have to be created in the same server instance that holds the input data. You can use a different LASR Analytic Server instance to hold the output data set. However, in order for the output data to be created in parallel symmetric mode, all the nodes that are used by the server instance that holds the input data must also be used by the server instance that holds the output data.

Terminating a SAS LASR Analytic Server Instance

You can continue to run high-performance analytical procedures and add and delete tables from the SAS LASR Analytic Server instance until you terminate the server instance as follows:

```
proc lasr term port=54545;
run;
```

Alongside-LASR Distributed Execution on a Subset of the Appliance Nodes

When you run PROC LASR to start a SAS LASR Analytic Server, you can specify the NODES= option in a PERFORMANCE statement to control how many nodes the LASR Analytic Server executes on. Similarly, a high-performance analytical procedure can execute on a subset of the nodes either because you specify the NODES= option in a PERFORMANCE statement or because you run alongside a DBMS or HDFS with an input data set that is distributed on a subset of the nodes on an appliance. In such situations, if a high-performance analytical procedure uses nodes on which the LASR Analytic Server is not running, then running alongside LASR is not supported. You can avoid this issue by specifying the NODES=ALL in the PERFORMANCE statement when you use PROC LASR to start the LASR Analytic Server.

Running High-Performance Analytical Procedures in Asymmetric Mode

This section provides examples of how you can run high-performance analytical procedures in asymmetric mode.

Asymmetric mode is commonly used when the data appliance and the computing appliance are distinct appliances. In order to be able to use an appliance as a data provider for high-performance analytical procedures that run in asymmetric mode on another appliance, it is not necessary that SAS High-Performance Econometrics be installed on the data appliance. However, it is essential that a SAS Embedded Process be installed on the data appliance and that SAS High-Performance Econometrics be installed on the computing appliance.

The following examples use a 24-node data appliance named "data_appliance.sas.com," which houses a Teradata DBMS and has a SAS Embedded Process installed.

The following statements load the simData data set of the preceding sections onto the data appliance:

```
libname dataLib teradata
        server  ="tera2650"
        user    =XXXXXX
        password=YYYYY
        database=mydb;

data dataLib.simData;
   set simData;
run;
```

NOTE: You can provision the appliance with data even if SAS High-Performance Econometrics software is not installed on the appliance.

The following subsections show how you can run the HPLOGISTIC procedure asymmetrically on distinct data and computing appliances.

Running in Asymmetric Mode on Distinct Appliances

Usually, there is no advantage to executing high-performance analytical procedures in asymmetric mode on one appliance, because data might have to be unnecessarily moved between nodes. The following example demonstrates the more typical use of asymmetric mode. In this example, the specified grid host "compute_appliance.sas.com" is a 142-node computing appliance that is different from the 24-node data appliance "data_appliance.sas.com," which houses the Teradata DBMS where the data reside.

The advantage of using different computing and data appliances is that the data appliance is not affected by the execution of high-performance analytical procedures except during the initial parallel data transfer. A potential disadvantage of this asymmetric mode of execution is that the performance can be limited by the bandwidth with which data can be moved between the appliances. However, because this data movement takes place in parallel from the nodes of the data appliance to the nodes of the computing appliance, this

potential performance bottleneck can be overcome with appropriately provisioned hardware. The following statements show how this is done:

```
proc hplogistic data=dataLib.simData;
   class a b c;
   model y = a b c x1 x2 x3;
   performance host = "compute_appliance.sas.com" nodes=30;
run;
```

Figure 2.8 shows the "Performance Information" and "Data Access Information" tables.

Figure 2.8 Asymmetric Mode with Distinct Data and Computing Appliances

The HPLOGISTIC Procedure

Performance Information	
Host Node	compute_appliance.sas.com
Execution Mode	Distributed
Number of Compute Nodes	30
Number of Threads per Node	40

Data Access Information			
Data	**Engine**	**Role**	**Path**
DATALIB.simData	TERADATA	Input	Parallel, Asymmetric

PROC HPLOGISTIC ran on 30 nodes of the computing appliance, even though the data were partitioned across the 24 nodes of the data appliance. The numeric results are not reproduced here, but they agree with the previous analyses shown in Figure 2.1 and Figure 2.2.

Every time you run a high-performance analytical procedure in asymmetric mode that uses different computing and data appliances, data are transferred between these appliances. If you plan to make repeated use of the same data, then it might be advantageous to temporarily persist the data that you need on the computing appliance. One way to persist the data is to store them as a table in a SAS LASR Analytic Server that runs on the computing appliance. By running PROC LASR in asymmetric mode, you can load the data in parallel from the data appliance nodes to the nodes on which the LASR Analytic Server runs on the computing appliance. You can then use a LIBNAME statement that associates a SAS libref with tables on the LASR Analytic Server. The following statements show how you do this:

```
proc lasr port=54345
          data=dataLib.simData
          path="/tmp/";
   performance host ="compute_appliance.sas.com" nodes=30;
run;

libname MyLasr sasiola tag="dataLib" port=54345 host="compute_appliance.sas.com" ;
```

Figure 2.9 show the "Performance Information" and "Data Access Information" tables.

Figure 2.9 PROC LASR Running in Asymmetric Mode

The LASR Procedure

Performance Information	
Host Node	compute_appliance.sas.com
Execution Mode	Distributed
Number of Compute Nodes	30

Data Access Information			
Data	**Engine**	**Role**	**Path**
DATALIB.simData	TERADATA	Input	Parallel, Asymmetric

By default, all the nodes on the computing appliance would be used. However, because NODES=30 was specified in the PERFORMANCE statement, PROC LASR ran on only 30 nodes of the computing appliance. The data were loaded asymmetrically in parallel from the 24 data appliance nodes to the 30 compute nodes on which PROC LASR ran.

After the data are loaded into a LASR Analytic Server that runs on the computing appliance, you can run high-performance analytical procedures alongside this LASR Analytic Server as shown by the following statements:

```
proc hplogistic data=MyLasr.simData;
   class a b c;
   model y = a b c x1 x2 x3;
   output out=MyLasr.myOutputData pred=myPred;
   performance host = "compute_appliance.sas.com";
run;
```

The following note, which appears in the SAS log, confirms that the output data set is created successfully:

```
NOTE: The table DATALIB.MYOUTPUTDATA has been added to the LASR Analytic Server
      with port 54345. The Libname is MYLASR.
```

You can use the dataLib libref that you used to load the data onto the data appliance to create an output data set on the data appliance.

```
proc hplogistic data=MyLasr.simData;
   class a b c;
   model y = a b c x1 x2 x3;
   output out=dataLib.myOutputData pred=myPred;
   performance host  = "compute_appliance.sas.com";
run;
```

The following note, which appears in the SAS log, confirms that the output data set is created successfully on the data appliance:

```
NOTE: The data set DATALIB.myOutputData has 100000 observations and 1 variables.
```

When you run a high-performance analytical procedure on a computing appliance and either read data from or write data to a different data appliance on which a SAS Embedded Process is running, the Read and Write operations take place in parallel without any movement of data to and from the SAS client.

When you no longer need the data in the SAS LASR Analytic Server, you should terminate the server instance as follows:

```
proc lasr term port=54345;
    performance host="compute_appliance.sas.com";
run;
```

If you configured Hadoop on the computing appliance, then you can create output data tables that are stored in the HDFS on the computing appliance. You can do this by using the SASHDAT engine as described in the section "Alongside-HDFS Execution" on page 23.

Alongside-HDFS Execution

Running high-performance analytical procedures alongside HDFS shares many features with running alongside the database. You can execute high-performance analytical procedures alongside HDFS by using either the SASHDAT engine or the Hadoop engine.

You use the SASHDAT engine to read and write data that are stored in HDFS in a proprietary SASHDAT format. In SASHDAT format, metadata that describe the data in the Hadoop files are included with the data. This enables you to access files in SASHDAT format without supplying any additional metadata. Additionally, you can also use the SASHDAT engine to read data in CSV (comma-separated value) format, but you need supply metadata that describe the contents of the CSV data. The SASHDAT engine provides highly optimized access to data in HDFS that are stored in SASHDAT format.

The Hadoop engine reads data that are stored in various formats from HDFS and writes data to HDFS in CSV format. This engine can use metadata that are stored in Hive, which is a data warehouse that supplies metadata about data that are stored in Hadoop files. In addition, this engine can use metadata that you create by using the HDMD procedure.

The following subsections provide details about using the SASHDAT and Hadoop engines to execute high-performance analytical procedures alongside HDFS.

Alongside-HDFS Execution by Using the SASHDAT Engine

If the grid host is a cluster that houses data that have been distributed by using the SASHDAT engine, then high-performance analytical procedures can analyze those data in the alongside-HDFS mode. The procedures use the distributed computing environment in which an analytic process is collocated with the nodes of the cluster. Data then pass from HDFS to the analytic process on each node of the cluster.

Before you can run a procedure alongside HDFS, you must distribute the data to the cluster. The following statements use the SASHDAT engine to distribute to HDFS the simData data set that was used in the previous two sections:

```
option set=GRIDHOST="hpa.sas.com";

libname hdatLib sashdat
        path="/hps";

 data hdatLib.simData (replace = yes) ;
     set simData;
 run;
```

In this example, the GRIDHOST is a cluster where the SAS Data in HDFS Engine is installed. If a data set that is named simData already exists in the hps directory in HDFS, it is overwritten because the REPLACE=YES data set option is specified. For more information about using this LIBNAME statement, see the section "LIBNAME Statement for the SAS Data in HDFS Engine" in the *SAS LASR Analytic Server: Administration Guide*.

The following HPLOGISTIC procedure statements perform the analysis in alongside-HDFS mode. These statements are almost identical to the PROC HPLOGISTIC example in the previous two sections, which executed in single-machine mode and alongside-the-database distributed mode, respectively.

Figure 2.10 shows the "Performance Information" and "Data Access Information" tables. You see that the procedure ran in distributed mode and that the input data were read in parallel symmetric mode. The numeric results shown in Figure 2.11 agree with the previous analyses shown in Figure 2.1, Figure 2.2, and Figure 2.5.

Figure 2.10 Alongside-HDFS Execution Performance Information

The HPLOGISTIC Procedure

Performance Information	
Host Node	hpa.sas.com
Execution Mode	Distributed
Number of Compute Nodes	13
Number of Threads per Node	24

Data Access Information			
Data	**Engine**	**Role**	**Path**
HDATLIB.SIMDATA	SASHDAT	Input	Parallel, Symmetric

Figure 2.11 Alongside-HDFS Execution Model Information

Model Information	
Data Source	HDATLIB.SIMDATA
Response Variable	y
Class Parameterization	GLM
Distribution	Binary
Link Function	Logit
Optimization Technique	Newton-Raphson with Ridging

Figure 2.11 *continued*

Parameter Estimates

Parameter	Estimate	Standard Error	DF	t Value	Pr > |t|
Intercept	5.7011	0.2539	Infty	22.45	<.0001
a 0	-0.01020	0.06627	Infty	-0.15	0.8777
a 1	0	.	.	.	.
b 0	0.7124	0.06558	Infty	10.86	<.0001
b 1	0	.	.	.	.
c 0	0.8036	0.06456	Infty	12.45	<.0001
c 1	0	.	.	.	.
x1	0.01975	0.000614	Infty	32.15	<.0001
x2	-0.04728	0.003098	Infty	-15.26	<.0001
x3	-0.1017	0.009470	Infty	-10.74	<.0001

Alongside-HDFS Execution by Using the Hadoop Engine

The following LIBNAME statement sets up a libref that you can use to access data that are stored in HDFS and have metadata in Hive:

```
libname hdoopLib hadoop
        server       = "hpa.sas.com"
        user         = XXXXX
        password     = YYYYY
        database     = myDB
        config       = "demo.xml" ;
```

For more information about LIBNAME options available for the Hadoop engine, see the LIBNAME topic in the Hadoop section of *SAS/ACCESS for Relational Databases: Reference*. The configuration file that you specify in the CONFIG= option contains information that is needed to access the Hive server. It also contains information that enables this configuration file to be used to access data in HDFS without using the Hive server. This information can also be used to specify replication factors and block sizes that are used when the engine writes data to HDFS.

The following DATA step uses the Hadoop engine to distribute to HDFS the simData data set that was used in the previous sections. The engine creates metadata for the data set in Hive.

```
data hdoopLib.simData;
    set simData;
run;
```

After you have loaded data or if you are accessing preexisting data in HDFS that have metadata in Hive, you can access this data alongside HDFS by using high-performance analytical procedures. The following HPLOGISTIC procedure statements perform the analysis in alongside-HDFS mode. These statements are similar to the PROC HPLOGISTIC example in the previous sections.

```
proc hplogistic data=hdoopLib.simData;
   class a b c;
   model y = a b c x1 x2 x3;
   performance host = "compute_appliance.sas.com" nodes=8;
run;
```

Figure 2.12 shows the "Performance Information" and "Data Access Information" tables. You see that the procedure ran in distributed mode and that the input data were read in parallel asymmetric mode. The numeric results shown in Figure 2.13 agree with the previous analyses.

Figure 2.12 Alongside-HDFS Execution by Using the Hadoop Engine

The HPLOGISTIC Procedure

Performance Information	
Host Node	compute_appliance.sas.com
Execution Mode	Distributed
Number of Compute Nodes	8
Number of Threads per Node	40

Data Access Information			
Data	**Engine**	**Role**	**Path**
GRIDLIB.SIMDATA	HADOOP	Input	Parallel, Asymmetric

Figure 2.13 Alongside-HDFS Execution by Using the Hadoop Engine

Model Information	
Data Source	GRIDLIB.SIMDATA
Response Variable	y
Class Parameterization	GLM
Distribution	Binary
Link Function	Logit
Optimization Technique	Newton-Raphson with Ridging

Parameter Estimates

Parameter	Estimate	Standard Error	DF	t Value	Pr > |t|
Intercept	5.7011	0.2539	Infty	22.45	<.0001
a 0	-0.01020	0.06627	Infty	-0.15	0.8777
a 1	0	.	.	.	.
b 0	0.7124	0.06558	Infty	10.86	<.0001
b 1	0	.	.	.	.
c 0	0.8036	0.06456	Infty	12.45	<.0001
c 1	0	.	.	.	.
x1	0.01975	0.000614	Infty	32.15	<.0001
x2	-0.04728	0.003098	Infty	-15.26	<.0001
x3	-0.1017	0.009470	Infty	-10.74	<.0001

The Hadoop engine also enables you to access tables in HDFS that are stored in various formats and that are not registered in Hive. You can use the HDMD procedure to generate metadata for tables that are stored in the following file formats:

- delimited text
- fixed-record length binary
- sequence files
- XML text

To read any other kind of file in Hadoop, you can write a custom file reader plug-in in Java for use with PROC HDMD. For more information about LIBNAME options available for the Hadoop engine, see the LIBNAME topic in the Hadoop section of *SAS/ACCESS for Relational Databases: Reference*.

The following example shows how you can use PROC HDMD to register metadata for CSV data independently from Hive and then analyze these data by using high-performance analytical procedures. The CSV data in the table csvExample.csv is stored in HDFS in the directory /user/demo/data. Each record in this table consists of the following fields, in the order shown and separated by commas.

1. a string of at most six characters
2. a numeric field with values of 0 or 1
3. a numeric field with real numbers

Suppose you want to fit a logistic regression model to these data, where the second field represents a target variable named Success, the third field represents a regressor named Dose, and the first field represents a classification variable named Group.

The first step is to use PROC HDMD to create metadata that are needed to interpret the table, as in the following statements:

```
libname hdoopLib hadoop
                 server       = "hpa.sas.com"
                 user         = XXXXX
                 password     = YYYYY
                 HDFS_PERMDIR = "/user/demo/data"
                 HDFS_METADIR = "/user/demo/meta"
                 config       = "demo.xml"
                 DBCREATE_TABLE_EXTERNAL=YES;

proc hdmd name=hdoopLib.csvExample data_file='csvExample.csv'
          format=delimited encoding=utf8 sep = ',';

     column Group    char(6);
     column Success  double;
     column Dose     double;
run;
```

The metadata that are created by PROC HDMD for this table are stored in the directory /user/demo/meta that you specified in the HDFS_METADIR = option in the preceding LIBNAME statement. After you create the metadata, you can execute high-performance analytical procedures with these data by using the hdoopLib libref. For example, the following statements fit a logistic regression model to the CSV data that are stored in the csvExample.csv table:

```
proc hplogistic data=hdoopLib.csvExample;
    class Group;
    model Success = Dose;
    performance host     = "compute_appliance.sas.com"
                gridmode = asym
                nodes    = 8;
run;
```

Figure 2.14 shows the results of this analysis. You see that the procedure ran in distributed mode and that the input data were read in parallel asymmetric mode. The metadata that you created by using the HDMD procedure have been used successfully in executing this analysis.

Figure 2.14 Alongside-HDFS Execution with CSV Data

The HPLOGISTIC Procedure

Performance Information	
Host Node	compute_appliance.sas.com
Execution Mode	Distributed
Number of Compute Nodes	8
Number of Threads per Node	40

Data Access Information			
Data	**Engine**	**Role**	**Path**
GRIDLIB.CSVEXAMPLE	HADOOP	Input	Parallel, Asymmetric

Model Information	
Data Source	GRIDLIB.CSVEXAMPLE
Response Variable	Success
Class Parameterization	GLM
Distribution	Binary
Link Function	Logit
Optimization Technique	Newton-Raphson with Ridging

Class Level Information		
Class	**Levels**	**Values**
Group	3	group1 group2 group3

Number of Observations Read	1000
Number of Observations Used	1000

Parameter Estimates					
Parameter	**Estimate**	**Standard Error**	**DF**	**t Value**	**Pr > |t|**
Intercept	0.1243	0.1295	Infty	0.96	0.3371
Dose	-0.2674	0.2216	Infty	-1.21	0.2277

Output Data Sets

In the alongside-the-database mode, the data are read in distributed form, minimizing data movement for best performance. Similarly, when you write output data sets and a high-performance analytical procedure executes in distributed mode, the data can be written in parallel into the database.

For example, in the following statements, the HPLOGISTIC procedure executes in distributed mode by using eight nodes on the appliance to perform the logistic regression on work.simData:

```
proc hplogistic data=simData;
   class a b c;
   model y = a b c x1 x2 x3;
   id a;
   output out=applianc.simData_out pred=p;
   performance host="hpa.sas.com" nodes=8;
run;
```

The output data set applianc.simData_out is written in parallel into the database. Although the data are fed on eight nodes, the database might distribute the data on more nodes.

When a high-performance analytical procedure executes in single-machine mode, all output objects are created on the client. If the libref of the output data sets points to the appliance, the data are transferred to the database on the appliance. This can lead to considerable performance degradation compared to execution in distributed mode.

Many procedures in SAS software add the variables from the input data set when an observationwise output data set is created. The assumption of high-performance analytical procedures is that the input data sets can be large and contain many variables. For performance reasons, the output data set contains the following:

- variables that are explicitly created by the statement
- variables that are listed in the ID statement, as described in Chapter 3, “Shared Statistical Concepts” (*SAS/STAT User’s Guide: High-Performance Procedures*)
- distribution keys or hash keys that are transferred from the input data set

Including this information enables you to add to the output data set information necessary for subsequent SQL joins without copying the entire input data set to the output data set.

Working with Formats

You can use SAS formats and user-defined formats with high-performance analytical procedures as you can with other procedures in the SAS System. However, because the analytic work is carried out in a distributed environment and might depend on the formatted values of variables, some special handling can improve the efficiency of work with formats.

High-performance analytical procedures examine the variables that are used in an analysis for association with user-defined formats. Any user-defined formats that are found by a procedure are transmitted automatically to the appliance. If you are running multiple high-performance analytical procedures in a SAS session and the analysis variables depend on user-defined formats, you can preprocess the formats. This step involves generating an XML stream (a file) of the formats and passing the stream to the high-performance analytical procedures.

Suppose that the following formats are defined in your SAS program:

```
proc format;
     value YesNo        1='Yes'        0='No';
     value checkThis    1='ThisisOne'  2='ThisisTwo';
     value $cityChar    1='Portage'    2='Kinston';
run;
```

The next group of SAS statements create the XML stream for the formats in the file *Myfmt.xml*, associate that file with the file reference myxml, and pass the file reference with the FMTLIBXML= option in the PROC HPLOGISTIC statement:

```
filename myxml 'Myfmt.xml';
libname  myxml XML92 xmltype=sasfmt tagset=tagsets.XMLsuv;
proc format cntlout=myxml.allfmts;
run;


proc hplogistic data=six fmtlibxml=myxml;
   class wheeze cit age;
   format wheeze best4. cit $cityChar.;
   model wheeze = cit age;
run;
```

Generation and destruction of the stream can be wrapped in convenience macros:

```
%macro Make_XMLStream(name=tempxml);
    filename &name 'fmt.xml';
    libname  &name XML92 xmltype=sasfmt tagset=tagsets.XMLsuv;
    proc format cntlout=&name..allfmts;
    run;
%mend;

%macro Delete_XMLStream(fref);
    %let rc=%sysfunc(fdelete(&fref));
%mend;
```

If you do not pass an XML stream to a high-performance analytical procedure that supports the FMTLIBXML= option, the procedure generates an XML stream as needed when it is invoked.

PERFORMANCE Statement

PERFORMANCE < *performance-options* > **;**

The PERFORMANCE statement defines performance parameters for multithreaded and distributed computing, passes variables that describe the distributed computing environment, and requests detailed results about the performance characteristics of a high-performance analytical procedure.

You can also use the PERFORMANCE statement to control whether a high-performance analytical procedure executes in single-machine or distributed mode.

You can specify the following *performance-options* in the PERFORMANCE statement:

BPC=*n*

specifies the number of bytes per character that is used in processing character strings in multibyte encodings. The default is the bytes per character of the encoding. The number of characters in a string is calculated as the byte length of the string divided by the bytes per character of the encoding. This will be incorrect when the strings in the multibyte encoding contain one or more single byte characters. In such cases, setting BPC=1 enables appropriate byte lengths to be used in processing such strings.

COMMIT=*n*

requests that the high-performance analytical procedure write periodic updates to the SAS log when observations are sent from the client to the appliance for distributed processing.

High-performance analytical procedures do not have to use input data that are stored on the appliance. You can perform distributed computations regardless of the origin or format of the input data, provided that the data are in a format that can be read by the SAS System (for example, because a SAS/ACCESS engine is available).

In the following example, the HPREG procedure performs LASSO variable selection where the input data set is stored on the client:

```
proc hpreg data=work.one;
   model y = x1-x500;
   selection method=lasso;
   performance nodes=10 host='mydca' commit=10000;
run;
```

In order to perform the work as requested using 10 nodes on the appliance, the data set Work.One needs to be distributed to the appliance.

High-performance analytical procedures send the data in blocks to the appliance. Whenever the number of observations sent exceeds an integer multiple of the COMMIT= size, a SAS log message is produced. The message indicates the actual number of observations distributed, and not an integer multiple of the COMMIT= size.

DETAILS

requests a table that shows a timing breakdown of the procedure steps.

GRIDHOST=*"name"*

HOST=*"name"*

specifies the name of the appliance host in single or double quotation marks. If this option is specified, it overrides the value of the GRIDHOST environment variable.

GRIDMODE=SYM | ASYM

MODE=SYM | ASYM

is a deprecated option that specifies whether to run the high-performance analytical procedure in symmetric (SYM) mode or asymmetric (ASYM) mode. This option overrides the GRIDMODE environment variable.

GRIDTIMEOUT=*s*

TIMEOUT=*s*

specifies the time-out in seconds for a high-performance analytical procedure to wait for a connection to the appliance and establish a connection back to the client. The default is 120 seconds. If jobs are submitted to the appliance through workload management tools that might suspend access to the appliance for a longer period, you might want to increase the time-out value.

INSTALL=*"name"*

INSTALLLOC=*"name"*

specifies the directory in which the shared libraries for the high-performance analytical procedure are installed on the appliance. Specifying the INSTALL= option overrides the GRIDINSTALLLOC environment variable.

LASRSERVER=*"path"*

LASR=*"path"*

specifies the fully qualified path to the description file of a SAS LASR Analytic Server instance. If the input data set is held in memory by this LASR Analytic Server instance, then the procedure runs alongside LASR. This option is not needed to run alongside LASR if the DATA= specification of the input data uses a libref that is associated with a LASR Analytic Server instance. For more information, see the section "Alongside-LASR Distributed Execution" on page 16 and the *SAS LASR Analytic Server: Administration Guide*.

NODES=ALL | *n*

NNODES=ALL | *n*

specifies the number of nodes in the distributed computing environment, provided that the data are not processed alongside the database.

Specifying NODES=0 indicates that you want to process the data in single-machine mode on the client machine. If the input data are not alongside the database, this is the default. The high-performance analytical procedures then perform the analysis on the client. For example, the following sets of statements are equivalent:

```
proc hplogistic data=one;
   model y = x;
run;
```

```
proc hplogistic data=one;
   model y = x;
   performance nodes=0;
run;
```

If the data are not read alongside the database, the NODES= option specifies the number of nodes on the appliance that are involved in the analysis. For example, the following statements perform the analysis in distributed mode by using 10 units of work on the appliance that is identified in the HOST= option:

```
proc hplogistic data=one;
   model y = x;
   performance nodes=10 host="hpa.sas.com";
run;
```

If the number of nodes can be modified by the application, you can specify a NODES=*n* option, where *n* exceeds the number of physical nodes on the appliance. The SAS High-Performance Econometrics software then *oversubscribes* the nodes and associates nodes with multiple units of work. For example, on a system that has 16 appliance nodes, the following statements oversubscribe the system by a factor of 3:

```
proc hplogistic data=one;
   model y = x;
   performance nodes=48 host="hpa.sas.com";
run;
```

Usually, it is not advisable to oversubscribe the system because the analytic code is optimized for a certain level of multithreading on the nodes that depends on the CPU count. You can specify NODES=ALL if you want to use all available nodes on the appliance without oversubscribing the system.

If the data are read alongside the distributed database on the appliance, specifying a nonzero value for the NODES= option has no effect. The number of units of work in the distributed computing environment is then determined by the distribution of the data and cannot be altered. For example, if you are running alongside an appliance with 24 nodes, the NODES= option in the following statements is ignored:

```
libname GPLib greenplm server=gpdca user=XXX password=YYY
              database=ZZZ;
proc hplogistic data=gplib.one;
   model y = x;
   performance nodes=10 host="hpa.sas.com";
run;
```

NTHREADS=*n*

THREADS=*n*

specifies the number of threads for analytic computations and overrides the SAS system option THREADS | NOTHREADS. If you do not specify the NTHREADS= option, the number of threads is determined based on the number of CPUs on the host on which the analytic computations execute. The algorithm by which a CPU count is converted to a thread count is specific to the high-performance analytical procedure. Most procedures create one thread per CPU for the analytic computations.

By default, high-performance analytical procedures run in multiple concurrent threads unless multithreading has been turned off by the NOTHREADS system option or you force single-threaded execution by specifying NTHREADS=1. The largest number that can be specified for *n* is 256. Individual high-performance analytical procedures can impose more stringent limits if called for by algorithmic considerations.

NOTE: The SAS system options THREADS | NOTHREADS apply to the client machine on which the SAS high-performance analytical procedures execute. They do not apply to the compute nodes in a distributed environment.

Chapter 3
The HPCDM Procedure

Contents

Overview: HPCDM Procedure

In many loss modeling applications, the loss events are analyzed by modeling the severity (magnitude) of loss and the frequency (count) of loss separately. The primary goal of preparing these models is to estimate the aggregate loss—that is, the total loss that occurs over a period of time for which the frequency model is applicable. For example, an insurance company might want to assess the expected and worst-case losses for a particular business line, such as automobile insurance, over an entire year given the models for the number of losses in a year and the severity of each loss. A bank might want to assess the value-at-risk (VaR), a measure of the worst-case loss, for a portfolio of assets given the frequency and severity models for each asset type.

Loss severity and loss frequency are random variables, so the aggregate loss is also a random variable. Instead of preparing a point estimate of the expected aggregate loss, it is more desirable to estimate its probability distribution, because this enables you to infer various aspects of the aggregate loss such as measures of location, scale (variability), and shape in addition to percentiles. For example, the value-at-risk that banks or insurance companies use to compute regulatory capital requirements is usually the estimate of the 97.5th or 99th percentile from the aggregate loss distribution.

Let N represent the frequency random variable for the number of loss events that occur in the time period of interest. Let X represent the severity random variable for the magnitude of one loss event. Then, the aggregate loss S is defined as

$$S = \sum_{j=1}^{N} X_j$$

The goal is to estimate the probability distribution of S. Let $F_X(x)$ denote the cumulative distribution function (CDF) of X, $F_X^{*n}(x)$ denote the n-fold convolution of the CDF of X, and $\Pr(N = n)$ denote the probability of seeing n losses as per the frequency distribution. The CDF of S is theoretically computable as

$$F_S(s) = \sum_{n=0}^{\infty} \Pr(N = n) \cdot F_X^{*n}(x)$$

This probability distribution model of S, characterized by the CDF $F_S(s)$, is referred to as a *compound distribution model* (CDM). The HPCDM procedure computes an estimate of the CDM, given the distribution models of X and N.

PROC HPCDM accepts the severity model of X as estimated by the SEVERITY procedure. It accepts the frequency model of N as estimated by the COUNTREG procedure. Both the SEVERITY and COUNTREG procedures are part of SAS/ETS software. Both procedures allow models of X and N to be conditional on external factors (regressors). In particular, you can model the severity distribution such that its scale parameter depends on severity regressors, and you can model the frequency distribution such that its mean depends on frequency regressors. The frequency model can also be a zero-inflated model. PROC HPCDM uses the estimates of model parameters and the values of severity and frequency regressors to estimate the compound distribution model.

Direct computation of F_S is usually a difficult task because of the need to compute the n-fold convolution. Klugman, Panjer, and Willmot (1998, Ch. 4) suggest some relatively efficient recursion and inversion methods for certain combinations of severity and frequency distributions. However, those methods assume that distributions of N and X are fixed and all Xs are identically distributed. When the distributions of X

and N are conditional on regressors, each set of regressor values results in a different distribution. So you must repeat the recursion and inversion methods for each combination of regressor values, and this repetition makes these methods prohibitively expensive. PROC HPCDM instead estimates the compound distribution by using a Monte Carlo simulation method, which can use all available computational resources to generate a sufficiently large, representative sample of the compound distribution while accommodating the dependence of distributions of X and N on external factors. Conceptually, the simulation method works as follows:

1. Use the specified frequency model to draw a value N, which represents the number of loss events.
2. Use the specified severity model to draw N values, each of which represents the magnitude of loss for each of the N loss events.
3. Add the N severity values from step 2 to compute aggregate loss S as

$$S = \sum_{j=1}^{N} X_j$$

   This forms one sample point of the CDM.

Steps 1 through 3 are repeated M number of times, where M is specified by you, to obtain the representative sample of the CDM. PROC HPCDM analyzes this sample to compute empirical estimates of various summary statistics of the compound distribution such as the mean, variance, skewness, and kurtosis in addition to percentiles such as the median, the 95th percentile, the 99th percentile, and so on. You can also use PROC HPCDM to write the entire simulated sample to an output data set and to produce the plot of the empirical distribution function (EDF), which serves as a nonparametric estimate of F_S.

The simulation process gets more complicated when the frequency and severity models contain regression effects. The CDM is then conditional on the given values of regressors. The simulation process essentially becomes a scenario analysis, because you need to specify the expected values of the regressors that together represent the scenario for which you want to estimate the CDM. PROC HPCDM enables you to specify an input data set that contains the scenario. If you are modeling a group of entities together (such as a portfolio of multiple assets or a group of insurance policies), each with a different set of characteristics, then the scenario consists of more than one observation, and each observation corresponds to a different entity. PROC HPCDM enables you to specify such a group scenario in the input data set and performs a realistic simulation of loss events that each entity can generate.

PROC HPCDM also enables you to specify externally simulated counts. This is useful if you have an empirical frequency model or if you estimate the frequency model by using a method other than PROC COUNTREG and simulate counts by using such a model. You can specify M replications of externally simulated counts. For each of the replications, in step 1 of the simulation, instead of using the frequency model, PROC HPCDM uses the count N that you specify. If the severity model contains regression effects, then you can specify the scenario to simulate for each of the M replications.

If the parameters of your severity and frequency models have uncertainty associated with them, and they usually do, then you can use PROC HPCDM to conduct parameter perturbation analysis to assess the effect of parameter uncertainty on the estimates of CDM. If you specify that P perturbed samples be generated, then the parameter set is perturbed P times, and each time PROC HPCDM makes a random draw from either the univariate normal distribution of each parameter or the multivariate normal distribution over all parameters. For each of the P perturbed parameter sets, a full compound distribution sample is simulated and summarized.

This process yields P number of estimates for each summary statistic and percentile, which are then used to provide you with estimates of the location and variability of each summary statistic and percentile.

You can also use PROC HPCDM to compute the distribution of an aggregate *adjusted* loss. For example, in insurance applications, you might want to compute the distribution of the *amount paid* in a given time period after applying adjustments such as deductible and policy limit to each individual loss. PROC HPCDM enables you to specify SAS programming statements to adjust each severity value. If X_j^a represents the adjusted severity value, then PROC HPCDM computes S^a, an aggregate adjusted loss, as

$$S^a = \sum_{j=1}^{N} X_j^a$$

All the analyses that PROC HPCDM conducts for the aggregate unadjusted loss, including scenario analysis and parameter perturbation analysis, are also conducted for the aggregate adjusted loss, thereby giving you a comprehensive picture of the adjusted compound distribution model.

Getting Started: HPCDM Procedure

This section outlines the use of the HPCDM procedure to fit compound distribution models. The examples are intended as a gentle introduction to some of the features of the procedure.

Estimating a Simple Compound Distribution Model

This example illustrates the simplest use of PROC HPCDM. Assume that you are an insurance company that has used the historical data about the number of losses per year and the severity of each loss to determine that the Poisson distribution is the best distribution for the loss frequency and that the gamma distribution is the best distribution for the severity of each loss. Now, you want to estimate the distribution of an aggregate loss to determine the worst-case loss that can be incurred by your policyholders in a year. In other words, you want to estimate the compound distribution of $S = \sum_{i=1}^{N} X_i$, where the loss frequency, N, follows the fitted Poisson distribution and the severity of each loss event, X_i, follows the fitted gamma distribution.

If your historical count and severity data are stored in the data sets Work.ClaimCount and Work.ClaimSev, respectively, then you need to ensure that you use the following PROC COUNTREG and PROC SEVERITY steps to fit and store the parameter estimates of the frequency and severity models:

```
/* Fit an intercept-only Poisson count model and
   write estimates to an item store */
proc countreg data=claimcount;
   model numLosses= / dist=poisson;
   store countStorePoisson;
run;

/* Fit severity models and write estimates to a data set */
proc severity data=claimsev criterion=aicc outest=sevest covout plots=none;
   loss lossValue;
   dist _predefined_;
run;
```

The STORE statement in the PROC COUNTREG step saves the count model information, including the parameter estimates, in the Work.CountStorePoisson item store. An item store contains the model information in a binary format that cannot be modified after it is created. You can examine the contents of an item store that is created by a PROC COUNTREG step by specifying a combination of the RESTORE= option and the SHOW statement in another PROC COUNTREG step. For more information, see Chapter 11, "The COUNTREG Procedure" (*SAS/ETS User's Guide*).

The OUTEST= option in the PROC SEVERITY statement stores the estimates of all the fitted severity models in the Work.SevEst data set. Let the best severity model that the PROC SEVERITY step chooses be the gamma distribution model.

You can now submit the following PROC HPCDM step to simulate an aggregate loss sample of size 10,000 by specifying the count model's item store in the COUNTSTORE= option and the severity model's data set of estimates in the SEVERITYEST= option:

```
/* Simulate and estimate Poisson-gamma compound distribution model */
proc hpcdm countstore=countStorePoisson severityest=sevest
           seed=13579 nreplicates=10000 plots=(edf(alpha=0.05) density)
           print=(summarystatistics percentiles);
   severitymodel gamma;
   output out=aggregateLossSample samplevar=aggloss;
   outsum out=aggregateLossSummary mean stddev skewness kurtosis
          p01 p05 p95 p995=var pctlpts=90 97.5;
run;
```

The SEVERITYMODEL statement requests that an aggregate sample be generated by compounding only the gamma distribution and the frequency distribution. Specifying the SEED= value helps you get an identical sample each time you execute this step, provided that you use the same execution environment. In the single-machine mode of execution, the execution environment is the combination of the operating environment and the number of threads that are used for execution. In the distributed computing mode, the execution environment is the combination of the operating environment, the number of nodes, and the number of threads that are used for execution on each node.

Upon completion, PROC HPCDM creates the two output data sets that you specify in the OUT= options of the OUTPUT and OUTSUM statements. The Work.AggregateLossSample data set contains 10,000 observations such that the value of the AggLoss variable in each observation represents one possible aggregate loss value that you can expect to see in one year. Together, the set of the 10,000 values of the AggLoss variable represents one sample of compound distribution. PROC HPCDM uses this sample to compute the empirical estimates of various summary statistics and percentiles of the compound distribution. The Work.AggregateLossSummary data set contains the estimates of mean, standard deviation, skewness, and kurtosis that you specify in the OUTSUM statement. It also contains the estimates of the 1st, 5th, 90th, 95th, 97.5th, and 99.5th percentiles that you specify in the OUTSUM statement. The value-at-risk (VaR) is an aggregate loss value such that there is a very low probability that an observed aggregate loss value exceeds the VaR. One of the commonly used probability levels to define VaR is 0.005, which makes the 99.5th percentile an empirical estimate of the VaR. Hence, the OUTSUM statement of this example stores the 99.5th percentile in a variable named VaR. VaR is one of the widely used measures of worst-case risk.

Some of the default output and some of the output that you have requested by specifying the PRINT= option are shown in Figure 3.1.

Figure 3.1 Information, Summary Statistics, and Percentiles of the Poisson-Gamma Compound Distribution

The HPCDM Procedure
Severity Model: Gamma
Count Model: Poisson

Compound Distribution Information	
Severity Model	Gamma Distribution
Count Model	Poisson Model in Item Store WORK.COUNTSTOREPOISSON

Sample Summary Statistics			
Mean	4076.8	**Median**	3440.2
Standard Deviation	3442.6	**Interquartile Range**	4523.9
Variance	11851305.5	**Minimum**	0
Skewness	1.14554	**Maximum**	27971.5
Kurtosis	1.75272	**Sample Size**	10000

Sample Percentiles	
Percentile	**Value**
1	0
5	0
25	1430.7
50	3440.2
75	5954.6
90	8743.8
95	10740.0
97.5	12453.3
99	14738.1
99.5	16406.8

Percentile Method = 5

The "Sample Summary Statistics" table indicates that for the given parameter estimates of the Poisson frequency and gamma severity models, you can expect to see a mean aggregate loss of 4,076.8 and a median aggregate loss of 3,440.2 in a year. The "Sample Percentiles" table indicates that there is a 0.5% chance that the aggregate loss exceeds 16,406.8, which is the VaR estimate, and a 2.5% chance that the aggregate loss exceeds 12,453.3. These summary statistic and percentile estimates provide a quantitative picture of the compound distribution. You can also visually analyze the compound distribution by examining the plots that PROC HPCDM prepares. The first plot in Figure 3.2 shows the empirical distribution function (EDF), which is a nonparametric estimate of the cumulative distribution function (CDF). The second plot shows the histogram and the kernel density estimate, which are nonparametric estimates of the probability density function (PDF).

Figure 3.2 Nonparametric CDF and PDF Plots of the Poisson-Gamma Compound Distribution

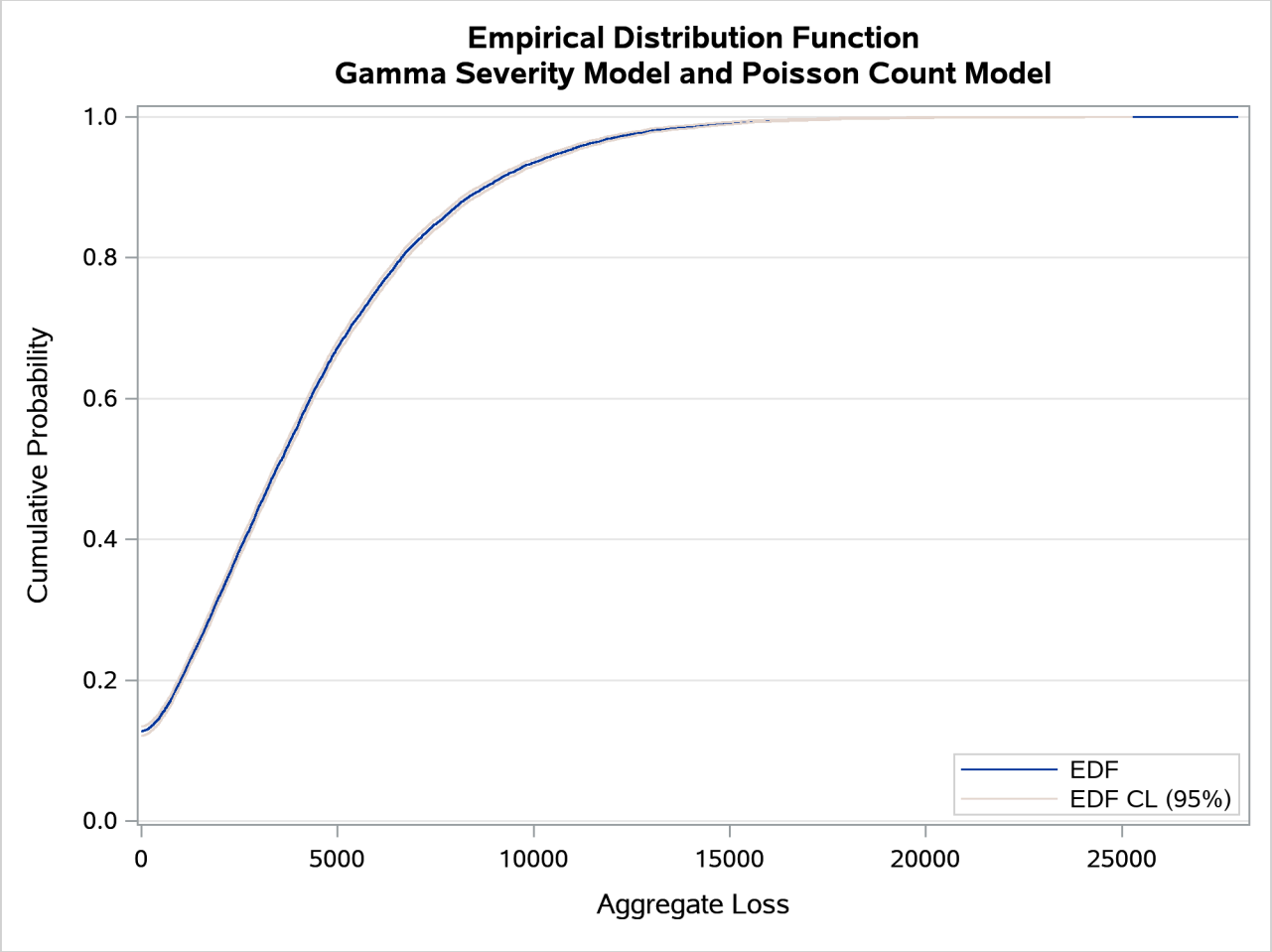
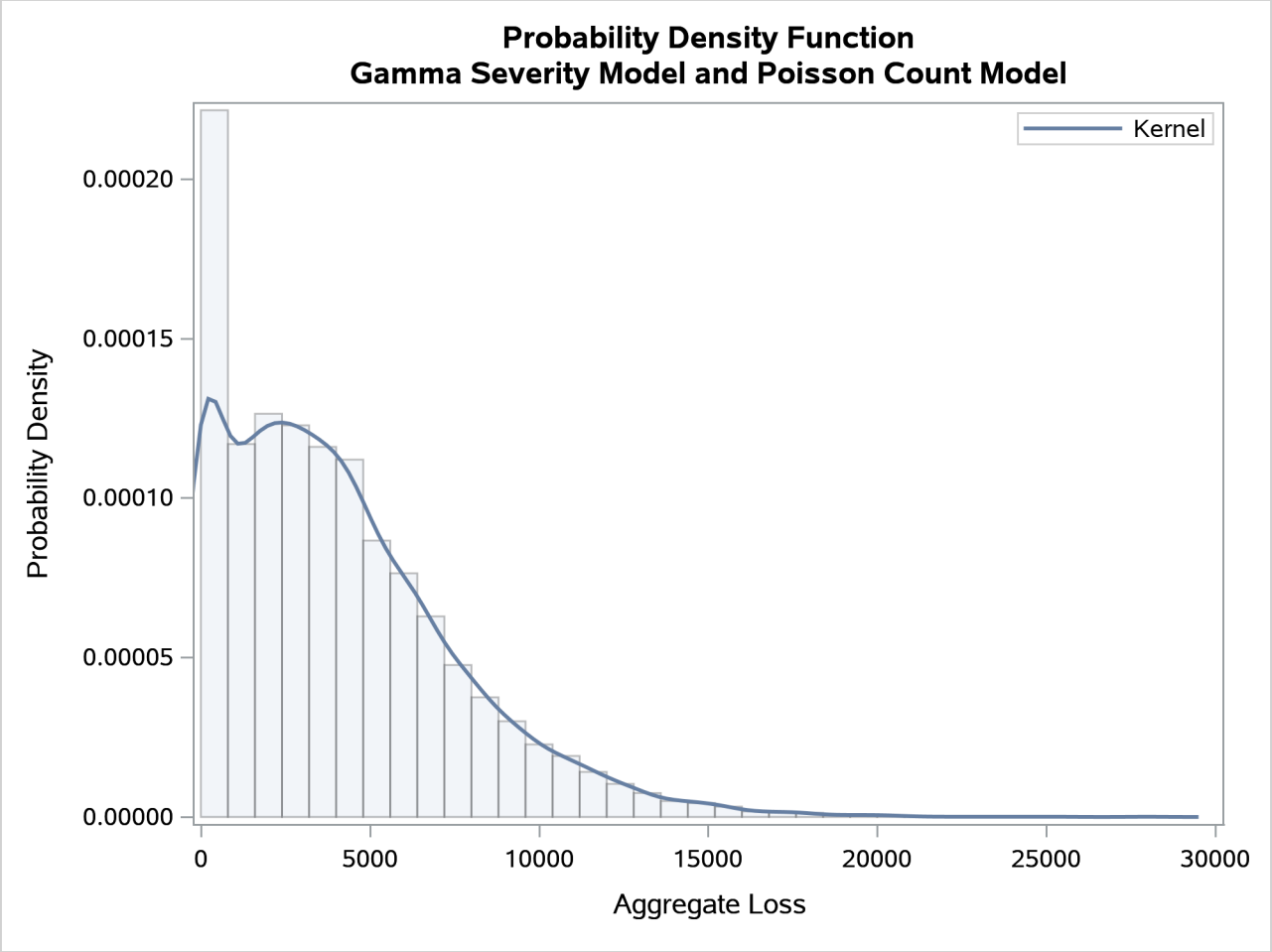


Figure 3.2 *continued*



The plots confirm the right skew that is indicated by the estimate of skewness in Figure 3.1 and a relatively fat tail, which is indicated by comparing the maximum and the 99.5th percentiles in Figure 3.1.

Analyzing the Effect of Parameter Uncertainty on the Compound Distribution

Continuing with the previous example, note that you have fitted the frequency and severity models by using the historical data. Even if you choose the best-fitting models, the true underlying models are not known exactly. This fact is reflected in the uncertainty that is associated with the parameters of your models. Any compound distribution estimate that is computed by using these uncertain parameter estimates is inherently uncertain. You can request that PROC HPCDM conduct parameter perturbation analysis, which assesses the effect of the parameter uncertainty on the estimates of the compound distribution by simulating multiple samples, each with parameters that are randomly perturbed from their mean estimates.

The following PROC HPCDM step adds the NPERTURBEDSAMPLES= option to the PROC HPCDM statement to request that perturbation analysis be conducted and the PRINT=PERTURBSUMMARY option to request that a summary of the perturbation analysis be displayed:

```
/* Conduct parameter perturbation analysis of
   the Poisson-gamma compound distribution model */
proc hpcdm countstore=countStorePoisson severityest=sevest
           seed=13579 nreplicates=10000 nperturbedsamples=30
           print(only)=(perturbsummary) plots=none;
   severitymodel gamma;
   output out=aggregateLossSample samplevar=aggloss;
   outsum out=aggregateLossSummary mean stddev skewness kurtosis
          p01 p05 p95 p995=var pctlpts=90 97.5;
run;
```

The Work.AggregateLossSummary data set contains the specified summary statistics and percentiles for all 30 perturbed samples. You can identify a perturbed sample by the value of the _DRAWID_ variable. The first few observations of the Work.AggregateLossSummary data set are shown in Figure 3.3. For the first observation, the value of the _DRAWID_ variable is 0, which represents an unperturbed sample—that is, the aggregate sample that is simulated without perturbing the parameters from their means.

Figure 3.3 Summary Statistics and Percentiles of the Perturbed Samples

SEVERITYMODEL	_COUNTMODEL_	_DRAWID_	_SAMPLEVAR_	N	MEAN	STDDEV
Gamma	Poisson	0	aggloss	10000	4076.78	3442.57
Gamma	Poisson	1	aggloss	10000	4155.34	3430.45
Gamma	Poisson	2	aggloss	10000	4024.20	3407.80
Gamma	Poisson	3	aggloss	10000	4241.48	3565.67
Gamma	Poisson	4	aggloss	10000	4161.65	3544.71
Gamma	Poisson	5	aggloss	10000	3892.26	3273.01
Gamma	Poisson	6	aggloss	10000	4474.95	3704.71
Gamma	Poisson	7	aggloss	10000	4216.14	3476.55
Gamma	Poisson	8	aggloss	10000	4049.96	3413.21
Gamma	Poisson	9	aggloss	10000	3950.08	3350.04
Gamma	Poisson	10	aggloss	10000	4286.65	3668.01

SKEWNESS	KURTOSIS	P01	P05	P90	P95	P97_5	var
1.14554	1.75272	0	0	8743.85	10740.03	12453.26	16406.81
1.12929	1.85707	0	0	8783.93	10569.44	12448.84	16390.42
1.16006	1.84717	0	0	8599.78	10441.09	12242.83	16219.61
1.11373	1.48627	0	0	9133.00	11107.39	12974.48	16946.76
1.17400	1.79535	0	0	8943.12	10800.95	12780.92	17142.43
1.08180	1.45528	0	0	8334.01	10180.93	11742.12	15147.64
1.07704	1.41288	0	0	9606.49	11489.24	13304.55	17662.93
1.11500	1.58827	0	0	8890.20	10732.59	12600.30	16581.44
1.14044	1.61876	0	0	8671.02	10546.62	12323.83	16333.81
1.09693	1.35455	0	0	8561.27	10322.30	11986.43	15829.09
1.16766	1.75264	0	0	9328.43	11299.10	13240.13	17417.01

The PRINT=PERTURBSUMMARY option in the preceding PROC HPCDM step produces the "Sample Perturbation Analysis" and "Sample Percentile Perturbation Analysis" tables that are shown in Figure 3.4. The tables show that you can expect a mean aggregate loss of about 4,098.5 and the standard error of the mean is 172.1. If you want to use the VaR estimate to determine the amount of reserves that you need to maintain to cover the worst-case loss, then you should consider not only the mean estimate of the 99.5th

percentile, which is about 16,448.2, but also the standard error of 708.9 to account for the effect of uncertainty in your frequency and severity parameter estimates.

Figure 3.4 Summary of Perturbation Analysis of the Poisson-Gamma Compound Distribution

The HPCDM Procedure
Severity Model: Gamma
Count Model: Poisson

Sample Perturbation Analysis

Statistic	Estimate	Standard Error
Mean	4098.5	172.08823
Standard Deviation	3470.4	136.68712
Variance	12062522	947666.8
Skewness	1.13817	0.04237
Kurtosis	1.65486	0.21853

Number of Perturbed Samples = 30

Size of Each Sample = 10000

Sample Percentile Perturbation Analysis

Percentile	Estimate	Standard Error
1	0	0
5	0	0
25	1425.4	90.99084
50	3421.7	155.81011
75	6003.1	244.90738
90	8818.2	362.42625
95	10732.8	422.41895
97.5	12540.3	504.12071
99	14839.4	680.49452
99.5	16448.2	708.87293

Number of Perturbed Samples = 30

Size of Each Sample = 10000

Scenario Analysis

The distributions of loss frequency and loss severity often depend on exogenous variables (regressors). For example, the number of losses and the severity of each loss that an automobile insurance policyholder incurs might depend on the characteristics of the policyholder and the characteristics of the vehicle. When you fit frequency and severity models, you need to account for the effects of such regressors on the probability distributions of the counts and severity. The COUNTREG procedure enables you to model regression effects on the mean of the count distribution, and the SEVERITY procedure enables you to model regression effects on the scale parameter of the severity distribution. When you use these models to estimate the compound distribution model of the aggregate loss, you need to specify a set of values for all the regressors, which represents the state of the world for which the simulation is conducted. This is referred to as the what-if or scenario analysis.

Consider that you, as an automobile insurance company, have postulated that the distribution of the loss event frequency depends on five regressors (external factors): age of the policyholder, gender, type of car, annual miles driven, and policyholder's education level. Further, the distribution of the severity of each loss depends on three regressors: type of car, safety rating of the car, and annual household income of the policyholder (which can be thought of as a proxy for the luxury level of the car). Note that the frequency model regressors and severity model regressors can be different, as illustrated in this example.

Let these regressors be recorded in the variables Age (scaled by a factor of 1/50), Gender (1: female, 2: male), CarType (1: sedan, 2: sport utility vehicle), AnnualMiles (scaled by a factor of 1/5,000), Education (1: high school graduate, 2: college graduate, 3: advanced degree holder), CarSafety (scaled to be between 0 and 1, the safest being 1), and Income (scaled by a factor of 1/100,000), respectively. Let the historical data about the number of losses that various policyholders incur in a year be recorded in the NumLoss variable of the Work.LossCounts data set, and let the severity of each loss be recorded in the LossAmount variable of the Work.Losses data set.

The following PROC COUNTREG step fits the count regression model and stores the fitted model information in the Work.CountregModel item store:

```
/* Fit negative binomial frequency model for the number of losses */
proc countreg data=losscounts;
   model numloss = age gender carType annualMiles education / dist=negbin;
   store work.countregmodel;
run;
```

You can examine the parameter estimates of the count model that are stored in the Work.CountregModel item store by submitting the following statements:

```
/* Examine the parameter estimates for the model in the item store */
proc countreg restore=work.countregmodel;
   show parameters;
run;
```

The "Parameter Estimates" table that is displayed by the SHOW statement is shown in Figure 3.5.

Figure 3.5 Parameter Estimates of the Count Regression Model

ITEM STORE CONTENTS: WORK.COUNTREGMODEL

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	0.910479	0.090515	10.06	<.0001
age	1	-0.626803	0.058547	-10.71	<.0001
gender	1	1.025034	0.032099	31.93	<.0001
carType	1	0.615165	0.031153	19.75	<.0001
annualMiles	1	-1.010276	0.017512	-57.69	<.0001
education	1	-0.280246	0.021677	-12.93	<.0001
_Alpha	1	0.318403	0.020090	15.85	<.0001

The following PROC SEVERITY step fits the severity scale regression models for all the common distributions that are predefined in PROC SEVERITY:

```
/* Fit severity models for the magnitude of losses */
proc severity data=losses plots=none outest=work.sevregest print=all;
   loss lossamount;
   scalemodel carType carSafety income;
   dist _predef_;
   nloptions maxiter=100;
run;
```

The comparison of fit statistics of various scale regression models is shown in Figure 3.6. The scale regression model that is based on the lognormal distribution is deemed the best-fitting model according to the likelihood-based statistics, whereas the scale regression model that is based on the generalized Pareto distribution (GPD) is deemed the best-fitting model according to the EDF-based statistics.

Figure 3.6 Severity Model Comparison

The SEVERITY Procedure

All Fit Statistics

Distribution	-2 Log Likelihood		AIC		AICC		BIC		KS		AD		CvM	
Burr	127231		127243		127243		127286		7.75407		224.47578		27.41346	
Exp	128431		128439		128439		128467		6.13537		181.75649		12.33919	
Gamma	128324		128334		128334		128370		7.54562		275.83377		24.59515	
Igauss	127434		127444		127444		127480		6.15855		211.51200		17.70942	
Logn	127062	*	127072	*	127072	*	127107	*	6.77687		212.70400		21.47945	
Pareto	128166		128176		128176		128211		5.37453		110.53673		7.07119	
Gpd	128166		128176		128176		128211		5.37453	*	110.53660	*	7.07116	*
Weibull	128429		128439		128439		128475		6.21268		190.73733		13.45425	

Note: The asterisk (*) marks the best model according to each column's criterion.

Now, you are ready to analyze the distribution of the aggregate loss that can be expected from a specific policyholder—for example, a 59-year-old male policyholder with an advanced degree who earns 159,870 and drives a sedan that has a very high safety rating about 11,474 miles annually. First, you need to encode and scale this information into the appropriate regressor variables of a data set. Let that data set be named Work.SinglePolicy, with an observation as shown in Figure 3.7.

Figure 3.7 Scenario Analysis Data for One Policyholder

age	gender	carType	annualMiles	education	carSafety	income
1.18	2	1	2.2948	3	0.99532	1.5987

Now, you can submit the following PROC HPCDM step to analyze the compound distribution of the aggregate loss that is incurred by the policyholder in the Work.SinglePolicy data set in a given year by using the frequency model from the Work.CountregModel item store and the two best severity models, lognormal and GPD, from the Work.SevRegEst data set:

```
/* Simulate the aggregate loss distribution for the scenario
   with single policyholder */
proc hpcdm data=singlePolicy nreplicates=10000 seed=13579 print=all
           countstore=work.countregmodel severityest=work.sevregest;
   severitymodel logn gpd;
```

```
    outsum out=onepolicysum mean stddev skew kurtosis median
          pctlpts=97.5 to 99.5 by 1;
run;
```

The displayed results from the preceding PROC HPCDM step are shown in Figure 3.8.

When you use a severity scale regression model, it is recommended that you verify the severity scale regressors that are used by PROC HPCDM by examining the Scale Model Regressors row of the "Compound Distribution Information" table. PROC HPCDM detects the severity regressors automatically by examining the variables in the SEVERITYEST= and DATA= data sets. If those data sets contain variables that you did not include in the SCALEMODEL statement in PROC SEVERITY, then such variables can be treated as severity regressors. One common mistake that can lead to this situation is to fit a severity model by using the BY statement and forget to specify the identical BY statement in the PROC HPCDM step; this can cause PROC HPCDM to treat BY variables as scale model regressors. In this example, Figure 3.8 confirms that the correct set of scale model regressors is detected.

Figure 3.8 Scenario Analysis Results for One Policyholder with Lognormal Severity Model

The HPCDM Procedure
Severity Model: Logn
Count Model: NegBin(p=2)

Compound Distribution Information	
Severity Model	Lognormal Distribution
Scale Model Regressors	carType carSafety income
Count Model	NegBin(p=2) Model in Item Store WORK.COUNTREGMODEL

Sample Summary Statistics			
Mean	214.05031	**Median**	0
Standard Deviation	436.27333	**Interquartile Range**	264.68948
Variance	190334.4	**Minimum**	0
Skewness	5.15057	**Maximum**	9005.2
Kurtosis	50.23372	**Sample Size**	10000

Sample Percentiles	
Percentile	**Value**
1	0
5	0
25	0
50	0
75	264.68948
95	950.03355
97.5	1340.0
98.5	1682.8
99	1979.5
99.5	2664.5
Percentile Method = 5	

The "Sample Summary Statistics" and "Sample Percentiles" tables in Figure 3.8 show estimates of the aggregate loss distribution for the lognormal severity model. The average expected loss is about 214, and the worst-case loss, if approximated by the 97.5th percentile, is about 1,340. The percentiles table shows that

the distribution is highly skewed to the right; this is also confirmed by the skewness estimate. The median estimate of 0 can be interpreted in two ways. One way is to conclude that the policyholder will not incur any loss in 50% of the years during which he or she is insured. The other way is to conclude that 50% of policyholders who have the characteristics of this policyholder will not incur any loss in a given year. However, there is a 2.5% chance that the policyholder will incur a loss that exceeds 1,340 in any given year and a 0.5% chance that the policyholder will incur a loss that exceeds 2,665 in any given year.

If the aggregate loss sample is simulated by using the GPD severity model, then the results are as shown in Figure 3.9. The average and worst-case losses are 213 and 1,337, respectively. These estimates are very close to the values that are predicted by the lognormal severity model.

Figure 3.9 Scenario Analysis Results for One Policyholder with GPD Severity Model

The HPCDM Procedure
Severity Model: Gpd
Count Model: NegBin(p=2)

Compound Distribution Information	
Severity Model	Generalized Pareto Distribution
Scale Model Regressors	carType carSafety income
Count Model	NegBin(p=2) Model in Item Store WORK.COUNTREGMODEL

Sample Summary Statistics			
Mean	212.54792	**Median**	0
Standard Deviation	401.95332	**Interquartile Range**	275.99091
Variance	161566.5	**Minimum**	0
Skewness	3.46433	**Maximum**	5360.2
Kurtosis	18.55938	**Sample Size**	10000

Sample Percentiles	
Percentile	**Value**
1	0
5	0
25	0
50	0
75	275.99091
95	977.06997
97.5	1337.4
98.5	1622.2
99	1867.4
99.5	2303.2
Percentile Method = 5	

The scenario that you just analyzed contains only one policyholder. You can extend the scenario to include multiple policyholders. Let the Work.GroupOfPolicies data set record information about five different policyholders, as shown in Figure 3.10.

Figure 3.10 Scenario Analysis Data for Multiple Policyholders

policyholderId	age	gender	carType	annualMiles	education	carSafety	income
1	1.18	2	1	2.2948	3	0.99532	1.59870
2	0.66	2	1	2.6718	2	0.86412	0.84459
3	0.64	2	2	1.9528	1	0.86478	0.50177
4	0.46	1	2	2.6402	2	0.27062	1.18870
5	0.62	1	1	1.7294	1	0.32830	0.37694

The following PROC HPCDM step conducts a scenario analysis for the aggregate loss that is incurred by all five policyholders in the Work.GroupOfPolicies data set together in one year:

```
/* Simulate the aggregate loss distribution for the scenario
   with multiple policyholders */
proc hpcdm data=groupOfPolicies nreplicates=10000 seed=13579 print=all
           countstore=work.countregmodel severityest=work.sevregest
           plots=(conditionaldensity(rightq=0.95)) nperturbedSamples=50;
   severitymodel logn gpd;
   outsum out=multipolicysum mean stddev skew kurtosis median
         pctlpts=97.5 to 99.5 by 1;
run;
```

The preceding PROC HPCDM step conducts perturbation analysis by simulating 50 perturbed samples. The perturbation summary results for the lognormal severity model are shown in Figure 3.11, and the results for the GPD severity model are shown in Figure 3.12. If the severity of each loss follows the fitted lognormal distribution, then you can expect that the group of policyholders together incurs an average loss of 5,300 $\pm$ 328 and a worst-case loss of 15,734 $\pm$ 960 when you define the worst-case loss as the 97.5th percentile.

Figure 3.11 Perturbation Analysis of Losses from Multiple Policyholders with Lognormal Severity Model

The HPCDM Procedure
Severity Model: Logn
Count Model: NegBin(p=2)

Compound Distribution Information	
Severity Model	Lognormal Distribution
Scale Model Regressors	carType carSafety income
Count Model	NegBin(p=2) Model in Item Store WORK.COUNTREGMODEL

Sample Perturbation Analysis		
Statistic	**Estimate**	**Standard Error**
Mean	5299.8	327.70569
Standard Deviation	4151.9	269.78790
Variance	17311274	2254196.7
Skewness	2.14414	1.24620
Kurtosis	16.65290	58.38318
Number of Perturbed Samples = 50		
Size of Each Sample = 10000		

Figure 3.11 *continued*

Sample Percentile Perturbation Analysis		
Percentile	Estimate	Standard Error
1	194.20187	28.77686
5	742.04381	59.84686
25	2379.0	154.80380
50	4324.3	272.87497
75	7113.4	438.24370
95	13101.5	805.58237
97.5	15734.1	960.35241
98.5	17746.7	1098.9
99	19384.7	1189.9
99.5	22409.7	1433.0
Number of Perturbed Samples = 50		
Size of Each Sample = 10000		

If the severity of each loss follows the fitted GPD distribution, then you can expect an average loss of 5,236 $\pm$ 365 and a worst-case loss of 14,992 $\pm$ 1,014.

If you decide to use the 99.5th percentile to define the worst-case loss, then the worst-case loss is 22,410 $\pm$ 1,433 for the lognormal severity model and 20,246 $\pm$ 1,400 for the GPD severity model. The numbers for lognormal and GPD are well within two standard errors of each other, which indicates that the aggregate loss distribution is less sensitive to the choice of these two severity distributions in this particular example; you can use the results from either of them.

Figure 3.12 Perturbation Analysis of Losses from Multiple Policyholders with GPD Severity Model

The HPCDM Procedure
Severity Model: Gpd
Count Model: NegBin(p=2)

Compound Distribution Information	
Severity Model	Generalized Pareto Distribution
Scale Model Regressors	carType carSafety income
Count Model	NegBin(p=2) Model in Item Store WORK.COUNTREGMODEL

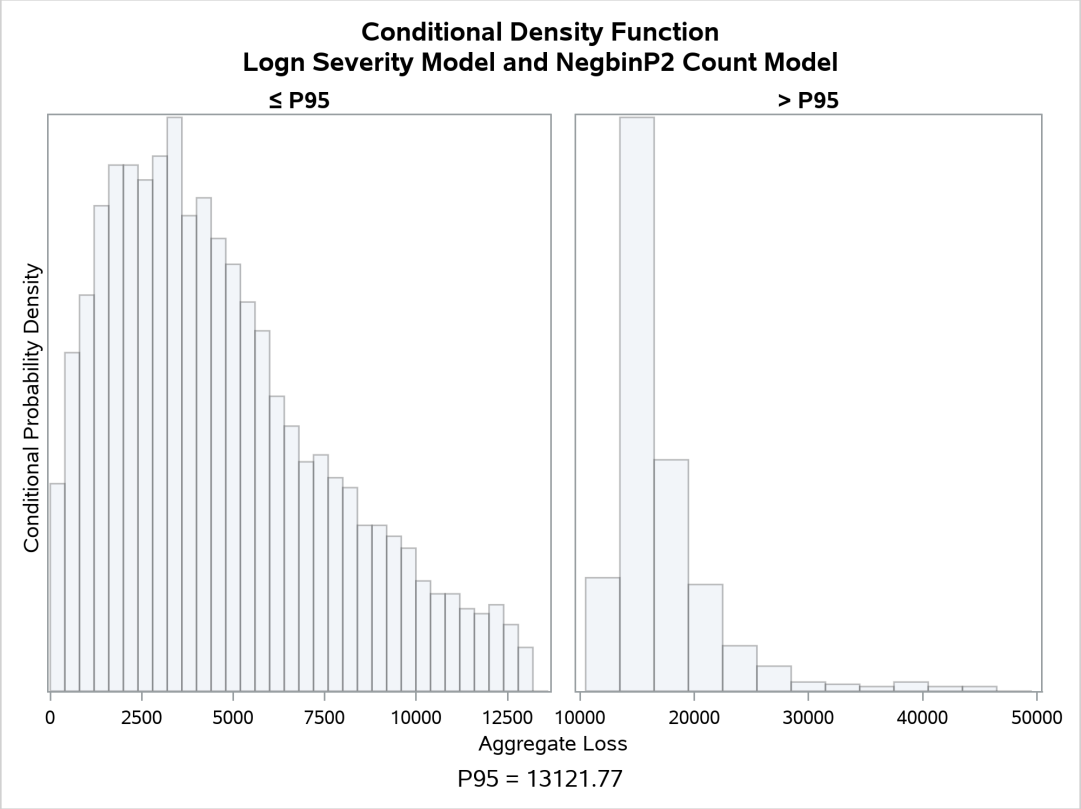
Sample Perturbation Analysis		
Statistic	Estimate	Standard Error
Mean	5235.5	364.77905
Standard Deviation	3894.0	270.62630
Variance	15236520	2107602.2
Skewness	1.48825	0.24040
Kurtosis	4.33915	6.27802
Number of Perturbed Samples = 50		
Size of Each Sample = 10000		

Figure 3.12 *continued*

Sample Percentile Perturbation Analysis		
Percentile	**Estimate**	**Standard Error**
1	155.29557	25.93762
5	699.37268	62.80951
25	2381.4	173.33561
50	4367.2	308.51028
75	7136.8	498.42048
95	12717.7	883.48043
97.5	14991.8	1014.0
98.5	16657.1	1148.8
99	17993.5	1235.1
99.5	20246.2	1399.7
Number of Perturbed Samples = 50		
Size of Each Sample = 10000		

The PLOTS=CONDITIONALDENSITY option that is used in the preceding PROC HPCDM step prepares the conditional density plots for the body and right-tail regions of the density function of the aggregate loss. The plots for the aggregate loss sample that is generated by using the lognormal severity model are shown in Figure 3.13. The plot on the left side is the plot of $\Pr(Y|Y \leq 13{,}122)$, where the limit 13,122 is the 95th percentile as specified by the RIGHTQ=0.95 option. The plot on the right side is the plot of $\Pr(Y|Y > 13{,}122)$, which helps you visualize the right-tail region of the density function. You can also request the plot of the left tail by specifying the LEFTQ= suboption of the CONDITIONALDENSITY option if you want to explore the details of the left tail region. Note that the conditional density plots are always produced by using the unperturbed sample.

Figure 3.13 Conditional Density Plots for the Aggregate Loss of Multiple Policyholders



Syntax: HPCDM Procedure

The following statements are available in the HPCDM procedure:

PROC HPCDM *options* **;**
BY *variable-list* **;**
DISTBY *replication-id-variable* **;**
SEVERITYMODEL *severity-model-list* **;**
EXTERNALCOUNTS COUNT=*frequency-variable* < **ID=***replication-id-variable* > **;**
OUTPUT OUT=*SAS-data-set* < *variable-name-options* > < / *out-option* > **;**
OUTSUM OUT=*SAS-data-set statistic-keyword*< =*variable-name* > < . . . *statistic-keyword*< =*variable-name* > > < *outsum-options* > **;**
PERFORMANCE *options* **;**
Programming statements ;

Functional Summary

Table 3.1 summarizes the statements and options available in the HPCDM procedure.

Table 3.1 PROC HPCDM Functional Summary

Description	Statement	Option
Statements		
Specifies the names of severity distribution models	SEVERITYMODEL	
Specifies externally simulated count data	EXTERNALCOUNTS	
Specifies where and how the full simulated samples are written	OUTPUT	
Specifies where and how the summary statistics of simulated samples are written	OUTSUM	
Specifies performance options	PERFORMANCE	
Specifies programming statements that define an objective function	Programming statements	
Data Set Options		
Specifies the input data set	PROC HPCDM	DATA=
Specifies the output data set for the full simulated samples	OUTPUT	OUT=
Specifies the output data set for the summary statistics	OUTSUM	OUT=
Model Input Options		
Specifies the variable that contains externally simulated counts	EXTERNALCOUNTS	COUNT=
Specifies the item store that contains the frequency (count) model	PROC HPCDM	COUNTSTORE=
Specifies the replicate identifier variable for external counts	EXTERNALCOUNTS	ID=
Specifies the input data set for parameter estimates of the severity models	PROC HPCDM	SEVERITYEST=
Specifies the item store for parameter estimates of the severity models	PROC HPCDM	SEVERITYSTORE=
Simulation Options		
Specifies the adjusted severity symbol in the programming statements	PROC HPCDM	ADJUSTEDSEVERITY=
Specifies the upper limit on the count for each sample point	PROC HPCDM	MAXCOUNTDRAW=
Specifies the number of parameter-perturbed samples to be simulated	PROC HPCDM	NPERTURBEDSAMPLES=
Specifies a number that controls the size of the simulated sample	PROC HPCDM	NREPLICATES=
Specifies the parameter perturbation method	PROC HPCDM	PERTURBMETHOD=
Specifies a seed for the internal pseudorandom number generator	PROC HPCDM	SEED=

Table 3.1 *continued*

Description	Statement	Option
Output Preparation Options		
Specifies the variable for the aggregate adjusted loss sample	OUTPUT	ADJSAMPLEVAR=
Specifies the method to compute the percentiles	PROC HPCDM	PCTLDEF=
Specifies the names of the variables for percentiles	OUTSUM	PCTLNAME=
Specifies the decimal precision to form default percentile variable names	OUTSUM	PCTLNDEC=
Specifies percentiles to compute and report	OUTSUM	PCTLPTS=
Specifies that all perturbed samples be written to the OUT= data set	OUTPUT	PERTURBOUT
Specifies the variable for the aggregate loss sample	OUTPUT	SAMPLEVAR=
Specifies the denominator for computing second- and higher-order moments	PROC HPCDM	VARDEF=
Displayed Output and Plotting Options		
Suppresses all displayed and graphical output	PROC HPCDM	NOPRINT
Specifies which graphical output to prepare	PROC HPCDM	PLOTS=
Specifies which displayed output to prepare	PROC HPCDM	PRINT=

PROC HPCDM Statement

PROC HPCDM *options* **;**

The PROC HPCDM statement invokes the procedure. You can specify the following *options*, which are listed in alphabetical order.

ADJUSTEDSEVERITY=*symbol-name*

ADJSEV=*symbol-name*

names the symbol that represents the adjusted severity value in the SAS programming statements that you specify. The *symbol-name* is a SAS name that conforms to the naming conventions of a SAS variable. For more information, see the section "Programming Statements" on page 66.

COUNTSTORE=*SAS-item-store*

names the item store that contains all the information about the frequency (count) model. The COUNTREG procedure generates this item store when you use the STORE statement.

The exogenous variables in the frequency model, if any, are deduced from this item store. The DATA= data set must contain all those variables.

If you specify a BY statement in the PROC COUNTREG step that creates the COUNTSTORE= item store, then you must specify an identical BY statement in the PROC HPCDM step.

You must specify this option if you do not specify the EXTERNALCOUNTS statement. This option is ignored if you specify the EXTERNALCOUNTS statement, because PROC HPCDM does not need to simulate frequency counts internally when you specify externally simulated counts.

If you specify the COUNTSTORE= option and execute the HPCDM procedure in distributed mode, then the distributed data access mode for the DATA= data set must be either client-data (local-data) mode or through-the-client mode—that is, the DATA= data set should not be stored on a distributed database appliance. For more information about data access modes, see the section "Data Access Modes" on page 11.

DATA=*SAS-data-set*

names the input data set that contains the values of regression variables in frequency or severity models and severity adjustment variables that you use in the programming statements.

The DATA= data set specifies information about the scenario for which you want to estimate the aggregate loss distribution. The interpretation of the contents of the data set and the supported distributed data access modes depend on whether you specify the EXTERNALCOUNTS statement. For more information, see the section "Specifying Scenario Data in the DATA= Data Set" on page 67.

MAXCOUNTDRAW=*number*

MAXCOUNT=*number*

specifies an upper limit on the number of loss events (count) that is used for simulating one aggregate loss sample point. If the *number* is equal to $N_{\max}$, then any count that is greater than $N_{\max}$ is assumed to be equal to $N_{\max}$, and only $N_{\max}$ severity draws are made to compute one point in the aggregate loss sample.

If you specify this option and also specify the COUNTSTORE= item store, then the limit is applied to each count that PROC HPCDM randomly draws from the count distribution in the COUNTSTORE= item store. Any count draw that is larger than the *number* is replaced by the *number*.

If you specify this option and also specify the EXTERNALCOUNTS statement, then the limit is applied to each observation in the DATA= data set, and any value of the COUNT= variable that is larger than the *number* is replaced by the *number*.

If you do not specify this option, then a default value of 1,000 is used.

If you specify a *number* that is significantly larger than 1,000, then PROC HPCDM might take a very long time to complete the simulation, especially when some counts are closer to the limit.

NOPRINT

turns off all displayed and graphical output. If you specify this option, then PROC HPCDM ignores any value that you specify for the PRINT= or PLOTS= option.

NPERTURBEDSAMPLES=*number*

NPERTURB=*number*

requests that parameter perturbation analysis be conducted. The model parameters are perturbed the specified *number* of times and a separate full sample is simulated for each set of perturbed parameter values. The summary statistics and percentiles are computed for each such perturbed sample, and their values are aggregated across the samples to compute the mean and standard deviation of each summary statistic and percentile.

The parameter perturbation procedure makes random draws of parameter values from a multivariate normal distribution if the covariance estimates of the parameters are available. For the multivariate normal

distribution of severity model parameters, PROC HPCDM attempts to read the covariance estimates from the SEVERITYEST= data set or the SEVERITYSTORE= item store. For the multivariate normal distribution of count model parameters, PROC HPCDM attempts to read the covariance estimates from the COUNTSTORE= store. If covariance estimates are not available or valid, then for each parameter, a random draw is made from the univariate normal distribution that has mean and standard deviation equal to the point estimate and the standard error, respectively, of that parameter. If neither covariance nor standard error estimates are available, then perturbation analysis is not conducted.

If you specify the PRINT=ALL or PRINT=PERTURBSUMMARY option, then the summary of perturbation analysis is printed for the core summary statistics and the percentiles of the aggregate loss distribution. If you specify the OUTSUM statement, then the requested summary statistics are written to the OUTSUM= data set for each perturbed sample. You can also optionally request that each perturbed sample be written in its entirety to the OUT= data set by specifying the PERTURBOUT option in the OUTPUT statement.

For more information on the parameter perturbation analysis, see the section "Parameter Perturbation Analysis" on page 82.

NREPLICATES=*number*

NREP=*number*

specifies a *number* that controls the size of the compound distribution sample that PROC HPCDM simulates. The *number* is interpreted differently based on whether you specify the EXTERNALCOUNTS statement.

If you do not specify the EXTERNALCOUNTS statement, then the sample size is equal to the *number* that you specify for this option. If you do not specify this option, then a default value of 100,000 is used.

If you specify the EXTERNALCOUNTS statement, then the number of replicates that you specify in the DATA= data set is multiplied by the *number* that you specify for this option to get the total size of the compound distribution sample. If you do not specify this option, then a default value of 1 is used.

PCTLDEF=*percentile-method*

specifies the method to compute the percentiles of the compound distribution. The *percentile-method* can be 1, 2, 3, 4, or 5. The default method is 5. For more information, see the description of the PCTLDEF= option in the UNIVARIATE procedure in the *Base SAS Procedures Guide: Statistical Procedures*.

PERTURBMETHOD=*perturbation-method*

specifies a method to perturb the parameters of the severity and count models. This option has no effect if you do not specify the NPERTURBEDSAMPLES= option. You can specify one of the following *perturbation-method*s:

ASYNC | 0

causes each thread of computation on each grid node to use its own set of perturbed model parameters. In particular, each thread uses its own pseudorandom number generator (PRNG) to perturb the model parameters, which is the same PRNG as the PRNG that the thread uses for making random draws from the severity or count distributions. Because each thread's PRNG starts with a different seed and because random draws that pertain to perturbation are interleaved with random draws that are made from the severity or count distributions, each thread effectively

uses a different set of perturbed models parameters even though it is simulating a subset of the same, global perturbed sample.

This method is computationally slightly more efficient because it does not need to synchronize the set of perturbed parameters among threads and grid nodes. However, each perturbed sample that it produces is conceptually a collection of smaller, distinct perturbed samples that belong to different compound distribution models.

SYNC | 1
specifies that all threads of computation on all grid nodes use the same (synchronized) set of perturbed model parameters. When you specify this option, PROC HPCDM in concept uses a single, dedicated PRNG to perturb the model parameters and shares those parameters with all threads on all grid nodes.

This method ensures that all observations of a particular perturbed sample belong to the same compound distribution model, because each thread on each grid node uses the same set of perturbed model parameters.

It is recommended that you specify the SYNC method. By default, PERTURBMETHOD=ASYNC to ensure that the current release of PROC HPCDM produces, by default, the same perturbation results as releases prior to SAS/ETS 15.1.

PLOTS < (*global-plot-options***) > =***plot-request-option*

PLOTS < (*global-plot-options***) > =(***plot-request-option . . . plot-request-option***)**
specifies the desired graphical output.

By default, the HPCDM procedure produces no graphical output.

You can specify the following *global-plot-option*:

ONLY
turns off the default graphical output and prepares only the requested plots.

If you specify more than one *plot-request-option*, then separate them with spaces and enclose them in parentheses. The following *plot-request-options* are available:

ALL
displays all the graphical output.

CONDITIONALDENSITY (*conditional-density-plot-options***)**

CONDPDF (*conditional-density-plot-options***)**
prepares a group of plots of the conditional density functions estimates. The group contains at most three plots, each conditional on the value of the aggregate loss being in one of the three regions that are defined by the quantiles that you specify in the following *conditional-density-plot-options*:

LEFTQ=*number*
specifies the quantile in the range (0,1) that marks the end of the left-tail region. If you specify a value of l for *number*, then the left-tail region is defined as the set of values that are less than or equal to q_l, where q_l is the lth quantile. For the left-tail region, nonparametric estimates of the conditional probability density function $f_S^l(s) = \Pr[S = s | S \leq q_l]$ are

plotted. The value of q_l is estimated by the $100l$th percentile of the simulated compound distribution sample.

If you do not specify this option or you specify a missing value for this option, then the left-tail region is not plotted.

RIGHTQ=*number*
specifies the quantile in the range (0,1) that marks the beginning of the right-tail region. If you specify a value of r for *number*, then the right-tail region is defined as the set of values that are greater than q_r, where q_r is the rth quantile. For the right-tail region, nonparametric estimates of the conditional probability density function $f_S^r(s) = \Pr[S = s|S > q_r]$ are plotted. The value of q_r is estimated by the $100r$th percentile of the simulated compound distribution sample.

If you do not specify this option or you specify a missing value for this option, then the right-tail region is not plotted.

You must specify nonmissing value for at least one of the preceding two options. For the region between the LEFTQ= and RIGHTQ= quantiles, which is referred to as the central or body region, nonparametric estimates of the conditional probability density function $f_S^c(s) = \Pr[S = s|q_l < S \leq q_r]$ are plotted. If you do not specify a LEFTQ= value, then q_l is assumed to be 0. If you do not specify a RIGHTQ= value, then q_r is assumed to be ∞.

DENSITY
prepares a plot of the nonparametric estimates of the probability density function (in particular, histogram and kernel density estimates) of the compound distribution.

EDF <(*edf-plot-option***)>**
prepares a plot of the nonparametric estimates of the cumulative distribution function of the compound distribution.

You can request that the confidence interval be plotted by specifying the following *edf-plot-option*:

ALPHA=*number*
specifies the confidence level in the (0,1) range that is used for computing the confidence intervals for the EDF estimates. If you specify a value of α for *number*, then the upper and lower confidence limits for the confidence level of $100(1 - \alpha)$ are plotted.

NONE
displays none of the graphical output. If you specify this option, then it overrides all other plot request options. The default graphical output is also suppressed.

Note that if the simulated sample size is large, then it can take a significant amount of time and memory to prepare the plots.

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.

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.

PERCENTILES
displays the percentiles of the compound distribution sample. This includes all the predefined percentiles, percentiles that you request in the OUTSUM statement, and percentiles that you specify for preparing conditional density plots.

PERTURBSUMMARY
displays the mean and standard deviation of the summary statistics and percentiles that are taken across all the samples produced by perturbing the model parameters. This option is valid only if you specify the NPERTURBEDSAMPLES= option in the PROC HPCDM statement.

SUMMARYSTATISTICS | SUMSTAT
displays the summary statistics of the compound distribution sample.

If you do not specify the PRINT= option or the ONLY *global-display-option*, then the default displayed output is equivalent to specifying PRINT=(SUMMARYSTATISTICS).

SEED=*number*
specifies the integer to use as the seed in generating the pseudorandom numbers that are used for simulating severity and frequency values.

If you do not specify the seed or if *number* is negative or 0, then the time of day from the computer's clock is used as the seed.

SEVERITYEST=*SAS-data-set*
names the input data set that contains the parameter estimates for the severity model. The format of this data set must be the same as the OUTEST= data set that is produced by the SEVERITY procedure.

The names of the regression variables in the scale regression model, if any, are deduced from this data set. In particular, PROC HPCDM assumes that all the variables in the SEVERITYEST= data set that do not appear in the following list are scale regression variables:

- BY variables
- _MODEL_, _TYPE_, _NAME_, and _STATUS_ variables
- variables that represent distribution parameters

The DATA= data set must contain all the regressors in the scale regression model.

To ensure that PROC HPCDM correctly matches the values of regressors and the values of regression parameter estimates, you might need to rename the regressors in the DATA= data set so that their

names match the names of the regressors that you specify in the SCALEMODEL statement of the PROC SEVERITY step that fits the severity model.

If you specify a BY statement in the PROC SEVERITY step that creates the SEVERITYEST= data set, then you must specify an identical BY statement in the PROC HPCDM step. Otherwise, PROC HPCDM detects the BY variables as regression variables in the scale regression model, which might produce unexpected results.

SEVERITYSTORE=*SAS-item-store*

SEVSTORE=*SAS-item-store*

specifies the item store that contains the context and estimates of the severity model. A PROC SEVERITY step with the OUTSTORE= option creates this item store.

If your severity model contains classification or interaction effects, then you need to use this option instead of the SEVERITYEST= option to specify the severity model. If you specify this option, you cannot specify the SEVERITYEST= option.

If you specify a BY statement in the PROC SEVERITY step that creates the SEVERITYSTORE= item store, then you must specify an identical BY statement in the PROC HPCDM step.

VARDEF=*divisor*

specifies the divisor to use in the calculation of variance, standard deviation, kurtosis, and skewness of the compound distribution sample. If the sample size is N, then you can specify one of the following values for the *divisor*:

DF

sets the divisor for variance to $N - 1$. This is the default. This also changes the definitions of skewness and kurtosis.

N

sets the divisor to N.

For more information, see the section "Descriptive Statistics" on page 83.

BY Statement

BY *variable-list* **;**

You can use the BY statement in the HPCDM procedure to process the input data set in groups of observations defined by the BY variables.

If you specify the BY statement, then you must specify the DATA= option in order to specify the input data set. PROC HPCDM expects the input data set to be sorted in the order of the BY variables unless you specify the NOTSORTED option.

The BY statement is always supported in the single-machine mode of execution. For the distributed mode, it is supported only when the DATA= data set resides on the client machine. In other words, the BY statement is supported only in the client-data (or local-data) mode of the distributed computing model and not for any of the alongside modes, such as the alongside-the-database or alongside HDFS mode.

DISTBY Statement

DISTBY *replication-id-variable* **;**

A DISTBY statement is necessary if and only if you specify an ID= variable in the EXTERNALCOUNTS statement. In fact, the *replication-id-variable* must be the same as the ID= variable. This is especially important in the distributed mode of execution, because when the observations in the DATA= data set are distributed to the grid nodes, by specifying the *replication-id-variable* as a DISTBY variable, you are instructing PROC HPCDM to make sure that the observations that have the same value for the *replication-id-variable* are always kept together on one grid node. This is required for correct simulation of the CDM in the presence of the ID= variable.

Contrast this to the BY variables that you specify in the BY statement. The observations of a BY group might be split across all the nodes of the grid, but the observations of a DISTBY group, which is nested within a BY group, are never split across the nodes of the grid.

The *replication-id-variable* must not appear in the BY statement. However, the DATA= data set must be sorted as if the *replication-id-variable* were listed after the BY variables in the BY statement.

Even though the DISTBY statement is important primarily in distributed mode, you must also specify it in single-machine mode.

EXTERNALCOUNTS Statement

EXTERNALCOUNTS COUNT=*frequency-variable* < **ID=***replication-id-variable* > **;**

The EXTERNALCOUNTS statement enables you to specify externally simulated frequency counts. By default, PROC HPCDM internally simulates the number of loss events by using the frequency model input (COUNTSTORE= item store). However, if you specify the EXTERNALCOUNTS statement, then PROC HPCDM uses the counts that you specify in the DATA= data set and simulates only the severity values internally.

If you specify more than one EXTERNALCOUNTS statement, only the first one is used.

You must specify the following option in the EXTERNALCOUNTS statement:

COUNT=*count-variable*
specifies the variable that contains the simulated counts. This variable must be present in the DATA= data set.

You can also specify the following option in the EXTERNALCOUNTS statement:

ID=*replication-id-variable*
specifies the variable that contains the replicate identifier. This variable must be present in the DATA= data set. Furthermore, you must specify the DISTBY statement with *replication-id-variable* as the only DISTBY variable to ensure correct simulation.

The observations of DATA= data set must be arranged such that the values of the ID= variable are in increasing order in each BY group or in the entire data set if you do not specify the BY statement.

If you do not specify the ID= option, then PROC HPCDM assumes that each observation represents one replication. In other words, the observation number serves as the default replication identifier.

The simulation process of using the external counts to generate the compound distribution sample is described in the section "Simulation with External Counts" on page 71.

OUTPUT Statement

OUTPUT OUT=*SAS-data-set* < *variable-name-options* > < / *out-option* > **;**

The OUTPUT statement enables you to specify the data set to output the generated compound distribution sample.

If you specify more than one OUTPUT statement, only the first one is used.

You must specify the output data set by using the following option:

OUT=*SAS-data-set*

OUTSAMPLE=*SAS-data-set*

specifies the output data set to contain the simulated compound distribution sample. If you specify programming statements to adjust individual severity values, then this data set contains both unadjusted and adjusted samples.

You can specify the following *variable-name-options* to control the names of the variables created in the OUT= data set:

ADJSAMPLEVAR=*variable-name*

specifies the name of the variable to contain the adjusted compound distribution sample in the OUT= data set. If you do not specify the ADJSAMPLEVAR= option, then "_AGGADJSEV_" is used by default.

This option is ignored if you do not specify the ADJUSTEDSEVERITY= option and the programming statements to adjust the simulated severity values.

SAMPLEVAR=*variable-name*

specifies the name of the variable to contain the simulated sample in the OUT= data set. If you do not specify the SAMPLEVAR= option, then "_AGGSEV_" is used by default.

Further, you can request that the perturbed samples be written to the OUT= data set by specifying the following *out-option*:

PERTURBOUT

specifies that all the perturbed samples be written to the OUT= data set. Each perturbed sample is identified by the _DRAWID_ variable in the OUT= data set. A value of 0 for the _DRAWID_ variable indicates an unperturbed sample.

Separate compound distribution samples are generated for each combination of specified severity and frequency models. The _SEVERITYMODEL_ and _COUNTMODEL_ columns in the OUT= data set identify the severity and frequency models, respectively, that are used to generate the sample in the SAMPLEVAR= and ADJSAMPLEVAR= variables.

OUTSUM Statement

OUTSUM OUT=*SAS-data-set statistic-keyword*< *=variable-name* > < . . . *statistic-keyword*< *=variable-name* > > < *outsum-options* > **;**

The OUTSUM statement enables you to specify the data set in which PROC HPCDM writes the summary statistics of the compound distribution samples.

If you specify more than one OUTSUM statement, only the first one is used.

You must specify the output data set by using the following option:

OUT=*SAS-data-set*

OUTSUM=*SAS-data-set*

specifies the output data set that contains the summary statistics of each of the simulated compound distribution samples. You can control the summary statistics that appear in this data set by specifying different *statistic-keywords* and *outsum-options*.

If you execute the HPCDM procedure in distributed mode, only the client-data (local-data) and through-the-client data access modes are supported for this data set. In other words, the libref that you specify for this data set should not point to a distributed database appliance. For more information about data access modes, see the section "Data Access Modes" on page 11.

You can request that one or more predefined statistics of the compound distribution sample be written to the OUTSUM= data set. For each specification of the form *statistic-keyword*< *=variable-name* >, the statistic that is specified by the *statistic-keyword* is written to a variable named *variable-name*. If you do not specify the *variable-name*, then the statistic is written to a variable named *statistic-keyword*. You can specify the following *statistic-keywords*:

KURTOSIS

KURT

specifies the kurtosis of the compound distribution sample.

MEAN

specifies the mean of the compound distribution sample.

MEDIAN

Q2

P50

specifies the median (the 50th percentile) of the compound distribution sample.

P01

specifies the 1st percentile of the compound distribution sample.

P05

specifies the 5th percentile of the compound distribution sample.

P95

specifies the 95th percentile of the compound distribution sample.

P99
specifies the 99th percentile of the compound distribution sample.

P99_5
P995
specifies the 99.5th percentile of the compound distribution sample.

Q1
P25
specifies the lower or 1st quartile (the 25th percentile) of the compound distribution sample.

Q3
P75
specifies the upper or 3rd quartile (the 75th percentile) of the compound distribution sample.

QRANGE
specifies the interquartile range (Q3–Q1) of the compound distribution sample.

SKEWNESS
SKEW
specifies the skewness of the compound distribution sample.

STDDEV
STD
specifies the standard deviation of the compound distribution sample.

All percentiles are computed by using the method that you specify for the PCTLDEF= option in the PROC HPCDM statement. You can also request additional percentiles to be reported in the OUTSUM= data set by specifying the following *outsum-options*:

PCTLPTS=*percentile-list*
specifies one or more percentiles that you want to be computed and written to the OUTSUM= data set. This option is useful if you need to request percentiles that are not available in the preceding list of *statistic-keyword* values. Each percentile value must belong to the (0,100) open interval. The *percentile-list* is a comma-separated list of numbers. You can also use a list notation of the form "*< number1 > to < number2 > by < increment >*". For example, the following two options are equivalent:

```
pctlpts=10, 20, 99.6, 99.7, 99.8, 99.9
pctlpts=10, 20, 99.6 to 99.9 by 0.1
```

The name of the variable for a given percentile value is decided by the PCTLNAME= option.

PCTLNAME=*percentile-variable-name-list*
specifies the names of the variables that contain the estimates of the percentiles that you request by using the PCTLPTS= option.

If you do not specify the PCTLNAME= option, then each percentile value *t* in the list of values in the PCTLPTS= option is written to the variable named "P*t*," where the decimal point in *t*, if any, is replaced by an underscore.

The *percentile-variable-name-list* is a space-separated list of names. You can also use a shortcut notation of <prefix>*m*–<prefix>*n* for two integers *m* and *n* ($m < n$) to generate the following list of names: <prefix>*m*, <prefix>$m + 1, \ldots,$ and <prefix>*n*. For example, the following two options are equivalent:

```
pctlname=p1 p2 pc5 pc6 pc7 pc8 pc9 pc10
pctlname=p1 p2 pc5-pc10
```

The name in *j*th position of the expanded name list of the PCTLNAME= option is used to create a variable for a percentile value in the *j*th position of the expanded value list of the PCTLPTS= option. If you specify k_n names in the PCTLNAME= option and k_v percentile values in the PCTLPTS= option, and if $k_n < k_v$, then the first k_n percentiles are written to the variables that you specify and the remaining $k_v - k_n$ percentiles are written to the variables that have the name of the form P*t*, where *t* is the text representation of the percentile value that is formed by retaining at most PCTLNDEC= digits after the decimal point and replacing the decimal point with an underscore ('_'). For example, assume you specify the options

```
pctlpts=10, 20, 99.3 to 99.5 by 0.1, 99.995
pctlname=pten ptwenty ninenine3-ninenine5
```

Then PROC HPCDM writes the 10th and 20th percentiles to pten and ptwenty variables, respectively; the 99.3rd through 99.5th percentiles to ninenine3, ninenine4, and ninenine5 variables, respectively; and the remaining 99.995th percentile to the P99_995 variable.

If a percentile value in the PCTLPTS= option matches a percentile value implied by one of the predefined percentile statistics and you specify the corresponding *statistic-keyword*, then the variable name that is implied by the *statistic-keyword*<*=variable-name*> specification takes precedence over the name that you specify in the PCTLNAME= option. For example, assume you specify the predefined percentile statistic of P95 as in the OUTSUM statement

```
outsum out=mypctls p95=ninetyfifth
        pctlpts=95 to 99 by 1 pctlname=pct95-pct99;
```

Then the 95th percentile is written to the ninetyfifth variable instead of the pct95 variable that the PCTLNAME= option implies.

PCTLNDEC=*integer-value*

specifies the maximum number of decimal places to use while creating the names of the variables for the percentile values in the PCTLPTS= option. The default value is 3. For example, for a percentile value of 99.9995, PROC HPCDM creates a variable named P99_999 by default, but if you specify PCTLNDEC=4, then the variable is named P99_9995.

The PCTLNDEC= option is used only for percentile values for which you do not specify a name in the PCTLNAME= option.

Note that all variable names in the OUTSUM= data set have a limit of 32 characters. If a name exceeds that limit, then it is truncated to contain only the first 32 characters. For more information about the variables in the OUTSUM= data set, see the section "Output Data Sets" on page 87.

PERFORMANCE Statement

PERFORMANCE *options* **;**

The PERFORMANCE statement defines performance parameters for distributed and multithreaded computing, passes variables that describe the distributed computing environment, and requests detailed results about the performance characteristics of PROC HPCDM.

You can also use the PERFORMANCE statement to control whether a high-performance analytical procedure executes in single-machine or distributed mode.

For more information about the PERFORMANCE statement, see the section "PERFORMANCE Statement" on page 31.

SEVERITYMODEL Statement

SEVERITYMODEL *severity-model-list* **;**

The SEVERITYMODEL statement specifies one or more severity distribution models that you want to use in simulating a compound distribution sample. The *severity-model-list* is a space-separated list of names of severity models that you would use with PROC SEVERITY's DIST statement. The SEVERITYEST= data set or the SEVERITYSTORE= item store must contain all the severity models in the list. If you specify the SEVERITYEST= data set and you specify a name that does not appear in the _MODEL_ column of the SEVERITYEST= data set, then that name is ignored. Similarly, if you specify the SEVERITYSTORE= item store and a severity model by that name does not appear in the item store, then that name is ignored.

If you specify more than one SEVERITYMODEL statement, only the first one is used.

If you do not specify a SEVERITYMODEL statement, then this is equivalent to specifying all the severity models that appear in the SEVERITYEST= data set or the SEVERITYSTORE= item store.

A compound distribution sample is generated for each of the severity models by compounding that severity model with the frequency model that you specify in the COUNTSTORE= item store or the external frequency model that is encoded by the COUNT= variable that you specify in the EXTERNALCOUNTS statement.

Programming Statements

In PROC HPCDM, you can use a series of programming statements that use variables in the DATA= data set to adjust an individual severity value. The adjusted severity values are aggregated to form a separate adjusted compound distribution sample.

The programming statements are executed for each simulated individual severity value. The observation of the input data set that is used to evaluate the programming statements is determined by the simulation procedure that is described in the section "Simulation Procedure" on page 68.

For more information, see the section "Simulation of Adjusted Compound Distribution Sample" on page 74.

Details: HPCDM Procedure

Specifying Scenario Data in the DATA= Data Set

A scenario represents a state of the world for which you want to estimate the distribution of aggregate losses. The state consists of one or more entities that generate the loss events. For example, an entity might be an individual who has an insurance policy or an organization that has a workers' compensation policy. Each entity has some characteristics of its own and some external factors that affect the frequency with which it generates the losses and the severity of each loss. For example, characteristics of an individual with an automobile insurance policy can include various demographics of the individual and various features of the automobile. Characteristics of an organization with a workers' compensation policy can be the number of employees, revenue, ratio of temporary to permanent employees, and so on. The organization can also be affected by external macroeconomic factors such as GDP and unemployment of the country where the organization operates and factors that affect its industry. You need to quantify and specify all these characteristics as external factors (regressors) when you fit severity and frequency models.

You should specify all the information about a scenario in the DATA= data set that you specify in the PROC HPCDM statement. Each observation in the DATA= data set encodes the characteristics of an entity. For proper simulation of severities, you must specify in the DATA= data set all the characteristics that you use as regressors in the severity scale regression models. When you use the COUNTSTORE= option to specify the frequency model, you must specify in the DATA= data set all the characteristics that you use as regressors in the frequency model in order to properly simulate the counts. All the regressors are expected to have nonmissing values. If any of the regressors have a missing value in an observation, then that observation is ignored.

The information in the DATA= data set is interpreted as follows, based on whether you specify the EXTERNALCOUNTS statement:

- If you do not specify the EXTERNALCOUNTS statement, then all the observations in the data set form a scenario. The observations are used together to compute one random draw from the compound distribution. The total number of draws is equal to the value that you specify in the NREPLICATES= option. The simulation process is described in the section "Simulation with Regressors and No External Counts" on page 69 and illustrated using an example in the section "Illustration of Aggregate Loss Simulation Process" on page 69.

  In this case, the distributed data access mode for the DATA= data set must be either client-data (local-data) mode or through-the-client mode—that is, the DATA= data set should not be stored on a distributed appliance. For more information about data access modes, see the section "Data Access Modes" on page 11.

- If you specify the EXTERNALCOUNTS statement, then the DATA= data set is expected to contain multiple replications (draws) of the frequency counts that you simulate externally for a scenario. The DATA= data set must contain the COUNT= variable that you specify in the EXTERNALCOUNTS statement. The replications are identified by the observation number or the ID= variable that you specify in the EXTERNALCOUNTS statement. For each observation in a given replication, the COUNT= variable is expected to contain the count of losses that are generated by the entity associated with that observation. All the observations of a given replication are used together to compute one random

draw from the compound distribution. The size of the compound distribution sample is equal to the number of distinct replications that you specify in the DATA= data set, multiplied by the value that you specify in the NREPLICATES= option. The simulation process is described in the section "Simulation with External Counts" on page 71 and illustrated using an example in the section "Illustration of the Simulation Process with External Counts" on page 72.

In this case, the distributed data access mode for the DATA= data set can be any of the supported data access modes. For more information about data access modes, see the section "Data Access Modes" on page 11.

In both cases, an observation can also contain severity adjustment variables that you can use to adjust the severity of the losses generated by that entity, based on some policy rules. For more information about simulating the adjusted compound distribution sample, see the section "Simulation of Adjusted Compound Distribution Sample" on page 74.

If you specify severity and frequency models that have no regression effects in them and if you do not specify externally simulated counts in the EXTERNALCOUNTS statement, then you do not need to specify the DATA= data set. This case corresponds to a fixed scenario that is represented entirely by the distribution parameters of the models.

Simulation Procedure

PROC HPCDM selects a simulation procedure based on whether you specify external counts or you request that PROC HPCDM simulate the counts, and whether the severity or frequency models contain regression effects. The following sections describe the process for the different scenarios.

Simulation with No Regressors and No External Counts

If you specify severity and frequency models that have no regression effects in them, and if you do not specify externally simulated counts in the EXTERNALCOUNTS statement, then PROC HPCDM uses the following simulation procedure.

The process is described for one severity distribution, *dist*. If you specify multiple severity distributions in the SEVERITYMODEL statement, then the process is repeated for each specified distribution.

The following steps are repeated M times to generate a compound distribution sample of size M, where M is the value that you specify in the NREPLICATES= option or the default value of that option:

1. Use the frequency model that you specify in the COUNTSTORE= option to draw a value N from the count distribution. N is the number of loss events that are expected to occur in the time period that is being simulated. N is adjusted to conform to the upper limit by setting it equal to $\min(N, N_{\max})$, where $N_{\max}$ is either 1,000 or the value that you specify in the MAXCOUNTDRAW= option.
2. Draw N values, X_j $(j = 1, \ldots, N)$, from the severity distribution *dist* with parameters that you specify in the SEVERITYEST= data set or the SEVERITYSTORE= item store.
3. Add the N severity values that are drawn in step 2 to compute one point S from the compound distribution as

$$S = \sum_{j=1}^{N} X_j$$

Note that although it is more common to fit the frequency model with regressors, PROC COUNTREG enables you to fit a frequency model without regressors. If you do not specify any regressors in the MODEL statement of the COUNTREG procedure, then it fits a model that contains only an intercept.

Simulation with Regressors and No External Counts

If the severity or frequency models have regression effects and if you do not specify externally simulated counts in the EXTERNALCOUNTS statement, then you must specify a DATA= data set to provide values of the regression variables, which together represent a scenario for which you want to simulate the CDM. In this case, PROC HPCDM uses the following simulation procedure.

The process is described for one severity distribution. If you specify multiple severity distributions in the SEVERITYMODEL statement, then the process is repeated for each specified distribution.

Note that you are doing scenario analysis when regression effects are present. Let K denote the number of observations that form the scenario. This is the number of observations either in the current BY group or in the entire DATA= data set if you do not specify the BY statement. If $K > 1$, then you are modeling the scenario for a group of entities. If $K = 1$, then you are modeling the scenario for one entity.

The following steps are repeated M times to generate a compound distribution sample of size M, where M is the value that you specify in the NREPLICATES= option or the default value of that option:

1. For each observation k ($k = 1, \ldots, K$), a count N_k is drawn from the frequency model that you specify in the COUNTSTORE= option. The parameters of this model are determined by the frequency regressors in observation k. N_k represents the number of loss events that are expected to be generated by entity k in the time period that is being simulated. N_k is adjusted to conform to the upper limit by setting it equal to $\min(N_k, N_{\max})$, where $N_{\max}$ is either 1,000 or the value that you specify in the MAXCOUNTDRAW= option.
2. Counts from all observations are added to compute $N = \sum_{k=1}^{K} N_k$. N is the total number of loss events that are expected to occur in the time period that is being simulated.
3. N number of random draws are made from the severity distribution, and they are added to generate one point of the compound distribution sample. Each of the N draws uses one of the K observations. If you specify a scale regression model for the severity distribution, then the scale parameter of the severity distribution is determined by the values of the severity regressors in the observation that is chosen for that draw.

If you specify the BY statement, then a separate sample of size M is created for each BY group in the DATA= data set.

Illustration of Aggregate Loss Simulation Process

As an illustration of the simulation process, consider a very simple example of analyzing the distribution of an aggregate loss that is incurred by a set of policyholders of an automobile insurance company in a period of one year. It is postulated that the frequency and severity distributions depend on three variables: Age, Gender (1: female, 2: male), and CarType (1: sedan, 2: sport utility vehicle). So these variables are used as regressors while you fit the count model and severity scale regression model by using the COUNTREG and SEVERITY procedures, respectively. Now, consider that you want to use the fitted frequency and severity models to estimate the distribution of the aggregate loss that is incurred by a set of five policyholders together. Let the characteristics of the five policyholders be encoded in a SAS data set named Work.Scenario that has the following contents:

Obs	age	gender	carType
1	30	2	1
2	25	1	2
3	45	2	2
4	33	1	1
5	50	1	1

The column Obs contains the observation number. It is shown only for the purpose of illustration. It need not be present in the data set. The following PROC HPCDM step simulates the scenario in the Work.Scenario data set:

```
proc hpcdm data=scenario
        severityest=<severity parameter estimates data set>
        countstore=<count model store> nreplicates=<sample size>;
    severitymodel <severity distribution name(s)>;
run;
```

The following process generates a sample from the aggregate loss distribution for the scenario in the Work.Scenario data set:

1. Use the values Age=30, Gender=2, and CarType=1 in the first observation to draw a count from the count distribution. Let that count be 2. Repeat the process for the remaining four observations. Let the counts be as shown in the Count column in the following table:

Obs	age	gender	carType	count
1	30	2	1	2
2	25	1	2	1
3	45	2	2	2
4	33	1	1	3
5	50	1	1	0

   Note that the Count column is shown for illustration only; it is not added as a variable to the DATA= data set.

2. The simulated counts from all the observations are added to get a value of $N = 8$. This means that for this particular sample point, you expect a total of eight loss events in a year from these five policyholders.

3. For the first observation, the scale parameter of the severity distribution is computed by using the values Age=30, Gender=2, and CarType=1. That value of the scale parameter is used together with estimates of the other parameters from the SEVERITYEST= data set to make two draws from the severity distribution. Each of the draws simulates the magnitude of the loss that is expected from the first policyholder. The process is repeated for the remaining four policyholders. The fifth policyholder does not generate any loss event for this particular sample point, so no severity draws are made by using the fifth observation. Let the severity draws, rounded to integers for convenience, be as shown in the _SEV_ column in the following table:

Obs	age	gender	carType	count	_sev_
1	30	2	1	2	350 2100
2	25	1	2	1	4500

```
3        45       2          2          2        700   4300
4        33       1          1          3        600   1500   950
5        50       1          1          0
```

Note that the _SEV_ column is shown for illustration only; it is not added as a variable to the DATA= data set.

PROC HPCDM adds the severity values of the eight draws to compute an aggregate loss value of 15,000. After recording this amount in the sample, the process returns to step 1 to compute the next point in the aggregate loss sample. For example, in the second iteration, the count distribution of each policyholder might generate one loss event for a total of five loss events, and the five severity draws from the severity distributions that govern each of the policyholders might add up to 5,000. Then, the value of 5,000 is recorded as the second point in the aggregate loss sample. The process continues until M aggregate loss sample points are simulated, where the M is the value that you specify in the NREPLICATES= option.

Simulation with External Counts

If you specify externally simulated counts by using the EXTERNALCOUNTS statement, then each replication in the input data set represents the loss events generated by an entity. An entity can be an individual or organization for which you want to estimate the compound distribution. If an entity has any characteristics that are used as external factors (regressors) in developing the severity scale regression model, then you must specify the values of those factors in the DATA= data set. If you specify the ID= variable, then multiple observations for the same replication ID represent different entities in a group for which you are simulating the CDM.

PROC HPCDM uses the following simulation procedure in the presence of externally simulated counts.

The process is described for one severity distribution. If you specify multiple severity distributions in the SEVERITYMODEL statement, then the process is repeated for each specified distribution.

Let there be M distinct replications in the current BY group of the DATA= data set or in the entire DATA= data set if you do not specify the BY statement. A replication is identified by either the observation number or the value of the ID= variable that you specify in the EXTERNALCOUNTS statement.

For each of the M values of the replication identifier, the following steps are executed R times, where R is the value of the NREPLICATES= option or the default value of that option:

1. Compute the total number of losses, N. If there are K ($K \geq 1$) observations for the current value of the replication identifier, then $N = \sum_{k=1}^{K} N_k$, where N_k is the value of the COUNT= variable for observation k, after it is adjusted to conform to the upper limit of either 1,000 or the value that you specify in the MAXCOUNTDRAW= option.

2. N number of random draws are made from the severity distribution, and they are added to generate one point of the compound distribution sample.

This process generates a compound distribution sample of size $M \times R$. If you specify the BY statement, then a separate sample of size $M \times R$ is created for each BY group in the DATA= data set.

Illustration of the Simulation Process with External Counts

In order to illustrate the simulation process, consider the following simple example. In this example, your severity model does not contain any regressors. An example that uses a severity scale regression model is illustrated later. Assume that you have made 10 random draws from an external count model and recorded them in the ExtCount variable of a SAS data set named Work.Counts1 as follows:

Obs	extCount
1	3
2	2
3	0
4	1
5	3
6	4
7	1
8	2
9	0
10	5

Because the data set does not contain an ID= variable, the observation number that is shown in the Obs column acts as the replicate identifier. The following PROC HPCDM step simulates an aggregate loss sample by using the Work.Counts1 data set:

```
proc hpcdm data=work.counts1 nreplicates=5
        severityest=<severity parameter estimates data set>;
    severitymodel <severity distribution name(s)>;
    externalcounts count=extCount;
run;
```

The simulation process works as follows:

1. For the first replication, which is associated with the first observation, three severity values are drawn from the severity distribution by using the parameter estimates that you specify in the SEVERITYEST= data set. If the severity values are 150, 500, and 320, then their sum of 970 is recorded as the first point of the aggregate loss sample. Because the value of the NREPLICATES= option is 5, this process of drawing three severity values and adding them to form a point of the aggregate loss sample is repeated four more times to generate a total of five sample points that correspond to the first observation.
2. For the second replication, two severity values are drawn from the severity distribution. If the severity values are 450 and 100, then their sum of 550 is recorded as a point of the aggregate loss sample. This process of drawing two severity values and adding them to form a point of the aggregate loss sample is repeated four more times to generate a total of five sample points that correspond to the second observation.
3. The process continues until all the replications, which are observations in this case, are exhausted.

The process results in an aggregate loss sample of size 50, which is equal to the number of replications in the data set (10) multiplied by the value of the NREPLICATES= option (5).

Now, consider an example in which the severity models in the SEVERITYEST= data set are scale regression models. In this case, the severity distribution that is used for drawing the severity value is decided by the values of regressors in the observation that is being processed. Consider that you want to simulate the aggregate loss that is incurred by one policyholder and you have recorded, in the ExtCount variable, the results of 10 random draws from an external count model. The DATA= data set has the following contents:

Obs	age	gender	carType	extCount
1	30	2	1	5
2	30	2	1	2
3	30	2	1	0
4	30	2	1	1
5	30	2	1	3
6	30	2	1	4
7	30	2	1	1
8	30	2	1	2
9	30	2	1	0
10	30	2	1	5

The simulation process in this case is the same as the process in the previous case of no regressors, except that the severity distribution that is used for drawing the severity values has a scale parameter that is determined by the values of the regressors Age, Gender, and CarType in the observation that is being processed. In this particular example, all observations have the same value for all regressors, indicating that you are modeling a scenario in which the characteristics of the policyholder do not change during the time for which you have simulated the number of events. You can also model a scenario in which the characteristics of the policyholder change by recording those changes in the values of the appropriate regressors.

Extending this example further, consider that you want to analyze the distribution of the aggregate loss that is incurred by a group of policyholders, as in the example in the section "Illustration of Aggregate Loss Simulation Process" on page 69. Let the Work.Counts2 data set record multiple replications of the number of losses that might be generated by each policyholder. The contents of the Work.Counts2 data set are as follows:

Obs	replicateId	age	gender	carType	extCount
1	1	30	2	1	2
2	1	25	1	2	1
3	1	45	2	2	3
4	1	33	1	1	5
5	1	50	1	1	1
6	2	30	2	1	3
7	2	25	1	2	2
8	1	45	2	2	0
9	2	33	1	1	4
10	2	50	1	1	1

The ReplicateId variable records the identifier for the replication. Each replication contains multiple observations, such that each observation represents one of the policyholders that you are analyzing. For simplicity, only the first two replications are shown here.

The following PROC HPCDM step simulates an aggregate loss sample by using the Work.Counts2 data set:

```
proc hpcdm data=work.counts2 nreplicates=3
        severityest=<severity parameter estimates data set>;
    severitymodel <severity distribution name(s)>;
    distby replicateId;
    externalcounts count=extCount id=replicateId;
    output out=aggloss samplevar=totalLoss;
run;
```

When you specify an ID= variable in the EXTERNALCOUNTS statement, you must specify the same ID= variable in the DISTBY statement in order for the procedure to work correctly in a distributed computing environment. Further, the DATA= set must be sorted in ascending order of the ID= variable values.

The simulation process works as follows:

1. First, the five observations of the first replication (ReplicateId=1 are analyzed. For the first observation (Obs=1), the scale parameter of the severity distribution is computed by using the values Age=30, Gender=2, and CarType=1. That value of the scale parameter is used together with estimates of the other parameters from the SEVERITYEST= data set to make two draws from the severity distribution. Next, the regressor values of the second observation are used to compute the scale parameter of the severity distribution, which is used to make one severity draw. The process continues such that the regressor values in the third, fourth, and fifth observations are used to decide the severity distribution to make three, five, and one draws from, respectively. Let the severity values that are drawn from the observations of this replication be as shown in the _SEV_ column in the following table, where the _SEV_ column is shown for illustration only; it is not added as a variable to the DATA= data set:

Obs	replicateId	age	gender	carType	extCount	_sev_
1	1	30	2	1	2	700 500
2	1	25	1	2	1	5000
3	1	45	2	2	3	900 1400 300
4	1	33	1	1	5	350 2000 150 800 600
5	1	50	1	1	1	250

   The values of all 12 severity draws are added to compute and record the value of 12,950 as the first point of the aggregate loss sample. Because you specify NREPLICATES=3 in the PROC HPCDM step, this process of making 12 severity draws from the respective observations is repeated two more times to generate a total of three sample points for the first replication.

2. The five observations of the second replication (ReplicateId=2) are analyzed next to draw three, two, four, and one severity values from the severity distributions, with scale parameters that are decided by the regressor values in the sixth, seventh, ninth, and tenth observations, respectively. The 10 severity values are added to form a point of the aggregate loss sample. This process of making 10 severity draws from the respective observations is repeated two more times to generate a total of three sample points for the second replication.

If your Work.Counts2 data set contains 10,000 distinct values of ReplicateId, then 30,000 observations are written to the Work.AggLoss data set that you specify in the OUTPUT statement of the preceding PROC HPCDM step. Because you specify SAMPLEVAR=TotalLoss in the OUTPUT statement, the aggregate loss sample is available in the TotalLoss column of the Work.AggLoss data set.

Simulation of Adjusted Compound Distribution Sample

If you specify programming statements that adjust the severity value, then a separate adjusted compound distribution sample is also generated.

Your programming statements are expected to implement an adjustment function f that uses the unadjusted severity value, X_j, to compute and return an adjusted severity value, X_j^a. To compute X_j^a, you might also use the sum of unadjusted severity values and the sum of adjusted severity values.

Formally, if N denotes the number of loss events that are to be simulated for the current replication of the simulation process, then for the severity draw, X_j, of the jth loss event ($j = 1, \ldots, N$), the adjusted severity value is

$$X_j^a = f(X_j, S_{j-1}, S_{j-1}^a)$$

where $S_{j-1} = \sum_{l=1}^{j-1} X_l$ is the aggregate unadjusted loss before X_j is generated and $S_{j-1}^a = \sum_{l=1}^{j-1} X_l^a$ is the aggregate adjusted loss before X_j is generated. The initial values of both types of aggregate losses are set to 0. In other words, $S_0 = 0$ and $S_0^a = 0$.

The aggregate adjusted loss for the replication is S_N^a, which is denoted by S^a for simplicity, and is defined as

$$S^a = \sum_{j=1}^{N} X_j^a$$

In your programming statements that implement f, you can use the following keywords as placeholders for the input arguments of the function f:

SEV
: indicates the placeholder for X_j, the unadjusted severity value. PROC HPCDM generates this value as described in the section "Simulation with No Regressors and No External Counts" on page 68 (step 2) or the section "Simulation with Regressors and No External Counts" on page 69 (step 3). PROC HPCDM supplies this value to your program.

CUMSEV
: indicates the placeholder for S_{j-1}, the sum of unadjusted severity values that PROC HPCDM generates before X_j is generated. PROC HPCDM supplies this value to your program.

CUMADJSEV
: indicates the placeholder for S_{j-1}^a, the sum of adjusted severity values that are computed by your programming statements before X_j is generated and adjusted. PROC HPCDM supplies this value to your program.

In your programming statements, you must assign the value of X_j^a, the output of function f, to a symbol that you specify in the ADJUSTEDSEVERITY= option in the PROC HPCDM statement. PROC HPCDM uses the final assigned value of this symbol as the value of X_j^a.

You can use most DATA step statements and functions in your program. The DATA step file and the data set I/O statements (for example, INPUT, FILE, SET, and MERGE) are not available. However, some functionality of the PUT statement is supported. For more information, see the section "PROC FCMP and DATA Step Differences" in *Base SAS Procedures Guide*.

The simulation process that generates the aggregate adjusted loss sample is identical to the process that is described in the section "Simulation with Regressors and No External Counts" on page 69 or the section "Simulation with External Counts" on page 71, except that after making each of the N severity draws, PROC HPCDM executes your severity adjustment programming statements to compute the adjusted severity (X_j^a). All the N adjusted severity values are added to compute S^a, which forms a point of the aggregate adjusted loss sample. The process is illustrated using an example in the section "Illustration of Aggregate Adjusted Loss Simulation Process" on page 77.

Using Severity Adjustment Variables

If you do not specify the DATA= data set, then your ability to adjust the severity value is limited, because you can use only the current severity draw, sums of unadjusted and adjusted severity draws that are made before the current draw, and some constant numbers to encode your adjustment policy. That is sufficient if you want to estimate the distribution of aggregate adjusted loss for only one entity. However, if you are simulating a scenario that contains more than one entity, then it might be more useful if the adjustment policy depends on factors that are specific to each entity that you are simulating. To do that, you must specify the DATA= data set and encode such factors as *adjustment variables* in the DATA= data set. Let A denote the set of values of the adjustment variables. Then, the form of the adjustment function f that computes the adjusted severity value becomes

$$X_j^a = f(X_j, S_{j-1}, S_{j-1}^a, A)$$

PROC HPCDM reads the values of adjustment variables from the DATA= data set and supplies the set of those values (A) to your severity adjustment program. For an invocation of f with an unadjusted severity value of X_j, the values in set A are read from the same observation that is used to simulate X_j.

All adjustment variables that you use in your program must be present in the DATA= data set. You must not use any keyword for a placeholder symbol as a name of any variable in the DATA= data set, whether the variable is a severity adjustment variable or a regressor in the frequency or severity model. Further, the following restrictions apply to the adjustment variables:

- You can use only numeric-valued variables in PROC HPCDM programming statements. This restriction also implies that you cannot use SAS functions or call routines that require character-valued arguments, unless you pass those arguments as constant (literal) strings or characters.
- You cannot use functions that create lagged versions of a variable in PROC HPCDM programming statements. If you need lagged versions, then you can use a DATA step before the PROC HPCDM step to add those versions to the input data set.

The use of adjustment variables is illustrated using an example in the section "Illustration of Aggregate Adjusted Loss Simulation Process" on page 77.

Aggregate Adjusted Loss Simulation for a Multi-entity Scenario

If you are simulating a scenario that consists of multiple entities, then you can use some additional pieces of information in your severity adjustment program. Let the scenario consist of K entities and let N_k denote the number of loss events that are incurred by kth entity ($k = 1, \ldots, K$) in the current iteration of the simulation process. Each value of N_k is adjusted to conform to the upper limit of either 1,000 or the value that you specify in the MAXCOUNTDRAW= option. The total number of severity draws that need to be made is $N = \sum_{k=1}^{K} N_k$. The aggregate adjusted loss is now defined as

$$S^a = \sum_{k=1}^{K} \sum_{j=1}^{N_k} X_{k,j}^a$$

where $X_{k,j}^a$ is an adjusted severity value of the jth draw ($j = 1, \ldots, N_k$) for the kth entity, and the form of the adjustment function f that computes $X_{k,j}^a$ is

$$X_{k,j}^a = f(X_{k,j}, S_{k,j-1}, S_{k,j-1}^a, S_{n-1}, S_{n-1}^a, A)$$

where $X_{k,j}$ is the value of the jth draw of unadjusted severity for the kth entity. $S_{k,j-1} = \sum_{l=1}^{j-1} X_{k,l}$ and $S^a_{k,j-1} = \sum_{l=1}^{j-1} X^a_{k,l}$ are the aggregate unadjusted loss and the aggregate adjusted loss, respectively, for the kth entity before $X_{k,j}$ is generated. The index n ($n = 1, \ldots, N$) keeps track of the total number of severity draws, across all entities, that are made before $X_{k,j}$ is generated. So $S_{n-1} = \sum_{l=1}^{n-1} X_l$ and $S^a_{n-1} = \sum_{l=1}^{n-1} X^a_l$ are the aggregate unadjusted loss and aggregate adjusted loss, respectively, for all the entities that are processed before $X_{k,j}$ is generated. Note that S_{n-1} and S^a_{n-1} include the $j-1$ draws that are made for the kth entity before $X_{k,j}$ is generated.

The initial values of all types of aggregate losses are set to 0. In other words, $S_0 = 0$, $S^a_0 = 0$, and for all values of k, $S_{k,0} = 0$ and $S^a_{k,0} = 0$.

PROC HPCDM uses the final value that you assign to the ADJUSTEDSEVERITY= symbol in your programming statements as the value of $X^a_{k,j}$.

In your severity adjustment program, you can use the following two additional placeholder keywords:

CUMSEVFOROBS
: indicates the placeholder for $S_{k,j-1}$, which is the total loss that is incurred by the kth entity before the current loss event. PROC HPCDM supplies this value to your program.

CUMADJSEVFOROBS
: indicates the placeholder for $S^a_{k,j-1}$, which is the total adjusted loss that is incurred by the kth entity before the current loss event. PROC HPCDM supplies this value to your program.

The previously described placeholder symbols _CUMSEV_ and _CUMADJSEV_ represent S_{n-1} and S^a_{n-1}, respectively. If you have only one entity in the scenario ($K = 1$), then the values of _CUMSEVFOROBS_ and _CUMADJSEVFOROBS_ are identical to the values of _CUMSEV_ and _CUMADJSEV_, respectively.

There is one caveat when a scenario consists of more than one entity ($K > 1$) and when you use any of the symbols for cumulative severity values (_CUMSEV_, _CUMADJSEV_, _CUMSEVFOROBS_, or _CUMADJSEVFOROBS_) in your programming statements. In this case, to make the simulation realistic, it is important to randomize the order of N severity draws across K entities. For more information, see the section "Randomizing the Order of Severity Draws across Observations of a Scenario" on page 80.

Illustration of Aggregate Adjusted Loss Simulation Process

This section continues the example in the section "Simulation with Regressors and No External Counts" on page 69 to illustrate the simulation of aggregate adjusted loss.

Recall that the earlier example simulates a scenario that consists of five policyholders. Assume that you want to compute the distribution of the aggregate amount paid to all the policyholders in a year, where the payment for each loss is decided by a deductible and a per-payment limit. To begin with, you must record the deductible and limit information in the input DATA= data set. The following table shows the DATA= data set from the earlier example, extended to include two variables, Deductible and Limit:

Obs	age	gender	carType	deductible	limit
1	30	2	1	250	5000
2	25	1	2	500	3000
3	45	2	2	100	2000
4	33	1	1	500	5000
5	50	1	1	200	2000

The variables Deductible and Limit are referred to as severity adjustment variables, because you need to use them to compute the adjusted severity. Let AmountPaid represent the value of adjusted severity that you are interested in. Further, let the following SAS programming statements encode your logic of computing the value of AmountPaid:

```
amountPaid = MAX(_sev_ - deductible, 0);
amountPaid = MIN(amountPaid, MAX(limit - _cumadjsevforobs_, 0));
```

PROC HPCDM supplies your program with values of the placeholder symbols _SEV_ and _CUMADJSEVFOROBS_, which represent the value of the current unadjusted severity draw and the sum of adjusted severity values from the previous draws, respectively, for the observation that is being processed. The use of _CUMADJSEVFOROBS_ helps you ensure that the payment that is made to a given policyholder in a year does not exceed the limit that is recorded in the Limit variable.

In order to simulate a sample for the aggregate of AmountPaid, you need to submit a PROC HPCDM step whose structure is like the following:

```
proc hpcdm data=<data set name> adjustedseverity=amountPaid
       severityest=<severity parameter estimates data set>
       countstore=<count model store>;
    severitymodel <severity distribution name(s)>;

    amountPaid = MAX(_sev_ - deductible, 0);
    amountPaid = MIN(amountPaid, MAX(limit - _cumadjsevforobs_, 0));
run;
```

The simulation process of one replication that generates one point of the aggregate loss sample and the corresponding point of the aggregate adjusted loss sample is as follows:

1. Use the values Age=30, Gender=2, and CarType=1 in the first observation to draw a count from the count distribution. Let that count be 3. Repeat the process for the remaining four observations. Let the counts be as shown in the Count column in the following table:

Obs	age	gender	carType	deductible	limit	count
1	30	2	1	250	5000	2
2	25	1	2	500	3000	1
3	45	2	2	100	2000	2
4	33	1	1	500	5000	3
5	50	1	1	200	2000	0

   Note that the Count column is shown for illustration only; it is not added as a variable to the DATA= data set.

2. The simulated counts from all the observations are added to get a value of $N = 8$. This means that for this particular replication, you expect a total of eight loss events in a year from these five policyholders.

3. For the first observation, the scale parameter of the severity distribution is computed by using the values Age=30, Gender=2, and CarType=1. That value of the scale parameter is used together with estimates of the other parameters from the SEVERITYEST= data set to make two draws from the severity distribution. The process is repeated for the remaining four policyholders. The fifth policyholder does not generate any loss event for this particular replication, so no severity draws are made by using the fifth observation. Let the severity draws, rounded to integers for convenience, be as shown in the

SEV column in the following table, where the _SEV_ column is shown for illustration only; it is not added as a variable to the DATA= data set:

Obs	age	gender	carType	deductible	limit	count	_sev_		
1	30	2	1	250	5000	2	350	2100	
2	25	1	2	500	3000	1	4500		
3	45	2	2	100	2000	2	700	4300	
4	33	1	1	200	5000	3	600	1500	950
5	50	1	1	200	2000	0			

The sample point for the aggregate unadjusted loss is computed by adding the severity values of eight draws, which gives an aggregate loss value of 15,000. The unadjusted aggregate loss is also referred to as the ground-up loss.

For each of the severity draws, your severity adjustment programming statements are executed to compute the adjusted severity, which is the value of AmountPaid in this case. For the draws in the preceding table, the values of AmountPaid are as follows:

Obs	deductible	limit	_sev_	_cumadjsevforobs_	amountPaid
1	250	5000	350	0	100
1	250	5000	2100	100	1850
2	500	3000	4500	0	3000
3	100	2000	700	0	600
3	100	2000	4300	600	1400
4	200	5000	600	0	400
4	200	5000	1500	400	1300
4	200	5000	950	1700	750

The adjusted severity values are added to compute the cumulative payment value of 9,400, which forms the first sample point for the aggregate adjusted loss.

After recording the aggregate unadjusted and aggregate adjusted loss values in their respective samples, the process returns to step 1 to compute the next sample point unless the specified number of sample points have been simulated.

In this particular example, you can verify that the order in which the 8 loss events are simulated does not affect the aggregate adjusted loss. As a simple example, consider the following order of draws that is different from the consecutive order that was used in the preceding table:

Obs	deductible	limit	_sev_	_cumadjsevforobs_	amountPaid
4	200	5000	600	0	400
3	100	2000	4300	0	2000
1	250	5000	350	0	100
3	100	2000	700	2000	0
4	200	5000	950	400	750
1	250	5000	2100	100	1850
2	500	3000	4500	0	3000
4	200	5000	1500	1150	1300

Although the payments that are made for individual loss events differ, the aggregate adjusted loss is still 9,400.

However, in general, when you use a cumulative severity value such as _CUMADJSEVFOROBS_ in your program, the order in which the draws are processed affects the final value of aggregate adjusted loss. For more information, see the sections "Randomizing the Order of Severity Draws across Observations of a Scenario" on page 80 and "Illustration of the Need to Randomize the Order of Severity Draws" on page 81.

Randomizing the Order of Severity Draws across Observations of a Scenario

If you specify a scenario that consists of a group of more than one entity, then it is assumed that each entity generates its loss events independently from other entities. In other words, the time at which the loss event of one entity is generated or recorded is independent of the time at which the loss event of another entity is generated or recorded. If entity k generates N_k loss events, where N_k is adjusted to conform to the upper limit of either 1,000 or the value that you specify in the MAXCOUNTDRAW= option, then the total number of loss events for a group of K entities is $N = \sum_{k=1}^{K} N_k$. To simulate the aggregate loss for this group, N severity draws are made and aggregated to compute one point of the compound distribution sample. However, to honor the assumption of independence among entities, the order of those N severity draws must be randomized across K entities such that no entity is preferred over another.

The K entities are represented by K observations of the scenario in the DATA= data set. If you specify external counts, the K observations correspond to the observations that have the same replication identifier value. If you do not specify the external counts, then the K observations correspond to all the observations in the BY group or in the entire DATA= set if you do not specify the BY statement.

The randomization process over K observations is implemented as follows. First, one of the K observations is chosen at random and one severity value is drawn from the severity distribution implied by that observation, then another observation is chosen at random and one severity value is drawn from its implied severity distribution, and so on. In each step, the total number of events that are simulated for the selected observation k is incremented by 1. When all N_k events for an observation k are simulated, observation k is retired and the process continues with the remaining observations until a total of N severity draws are made. Let $k(j)$ denote a function that implements this randomization by returning an observation k ($k = 1, \ldots, K$) for the jth draw ($j = 1, \ldots, N$). The aggregate loss computation can then be formally written as

$$S = \sum_{j=1}^{N} X_{k(j)}$$

where $X_{k(j)}$ denotes the severity value that is drawn by using observation $k(j)$.

If you do not specify a scale regression model for severity, then all severity values are drawn from the same severity distribution. However, if you specify a scale regression model for severity, then the severity draw is made from the severity distribution that is determined by the values of regressors in observation k. In particular, the scale parameter of the distribution depends on the values of regressors in observation k. If $R(l)$ denotes the scale regression model for observation l and $X_{R(l)}$ denotes the severity value drawn from scale regression model $R(l)$, then the aggregate loss computation can be formally written as

$$S = \sum_{j=1}^{N} X_{R(k(j))}$$

This randomization process is especially important in the context of simulating an adjusted compound distribution sample when your severity adjustment program uses the aggregate adjusted severity observed so

far to adjust the next severity value. For an illustration of the need to randomize in such cases, see the next section.

Illustration of the Need to Randomize the Order of Severity Draws

This section uses the example of the section "Illustration of Aggregate Adjusted Loss Simulation Process" on page 77, but with the following PROC HPCDM step:

```
proc hpcdm data=<data set name> adjustedseverity=amountPaid
       severityest=<severity parameter estimates data set>
       countstore=<count model store>;
    severitymodel <severity distribution name(s)>;

    if (_cumadjsev_ > 15000) then
        amountPaid = 0;
    else do;
        penaltyFactor = MIN(3, 15000/(15000 - _cumadjsev_));
        amountPaid = MAX(0, _sev_ - deductible * penaltyFactor);
    end;
run;
```

The severity adjustment statements in the preceding steps compute the value of AmountPaid by using the following provisions in the insurance policy:

- There is a limit of 15,000 on the total amount that can be paid in a year to the group of policyholders that is being simulated. The amount of payment for each loss event depends on the total amount of payments before that loss event.
- The penalty for incurring more losses is imposed in the form of an increased deductible. In particular, the deductible is increased by the ratio of the maximum cumulative payment (15,000) to the amount that remains available to pay for future losses in the year. The factor by which the deductible can be raised has a limit of three.

This example illustrates only step 3 of the simulation process, where randomization is done. It assumes that step 2 of the simulation process is identical to the step 2 in the example in the section "Illustration of Aggregate Adjusted Loss Simulation Process" on page 77. At the beginning of step 3, let the severity draws from all the observations be as shown in the _SEV_ column in the following table:

Obs	age	gender	carType	deductible	count	_sev_		
1	30	2	1	250	2	350	2100	
2	25	1	2	500	1	4500		
3	45	2	2	100	2	700	4300	
4	33	1	1	200	3	600	1500	950
5	50	1	1	200	0			

If the order of these eight draws is not randomized, then all the severity draws for the first observation are adjusted before all the severity draws of the second observation, and so on. The execution of the severity adjustment program leads to the following sequence of values for AmountPaid:

Obs	deductible	_sev_	_cumadjsev_	penaltyFactor	amountPaid
1	250	350	0	1	100
1	250	2100	100	1.0067	1848.32
2	500	4500	1948.32	1.1493	3925.36

3	100	700	5873.68	1.6436	535.64
3	100	4300	6409.32	1.7461	4125.39
4	200	600	10534.72	3	0
4	200	1500	10534.72	3	900
4	200	950	11434.72	3	350

The preceding sequence of simulating loss events results in a cumulative payment of 11,784.72.

If the sequence of draws is randomized over observations, then the computation of the cumulative payment might proceed as follows for one instance of randomization:

Obs	deductible	_sev_	_cumadjsev_	penaltyFactor	amountPaid
2	500	4500	0	1	4000
1	250	350	4000	1.3636	9.09
3	100	700	4009.09	1.3648	563.52
4	200	950	4572.61	1.4385	662.30
4	200	1500	5234.91	1.5361	1192.78
1	250	2100	6427.69	1.7498	1662.54
4	200	600	8090.24	2.1708	165.83
3	100	4300	8256.07	2.2242	4077.58

In this example, a policyholder is identified by the value in the Obs column. As the table indicates, PROC HPCDM randomizes the order of loss events not only across policyholders but also across the loss events that a given policyholder incurs. The particular sequence of loss events that is shown in the table results in a cumulative payment of 12,333.65. This differs from the cumulative payment that results from the previously considered nonrandomized sequence of loss events, which tends to penalize the fourth policyholder by always processing her payments after all other payments, with a possibility of underestimating the total paid amount. This comparison not only illustrates that the order of randomization affects the aggregate adjusted loss sample but also corroborates the arguments about the importance of order randomization that are made at the beginning of the section "Randomizing the Order of Severity Draws across Observations of a Scenario" on page 80.

Parameter Perturbation Analysis

It is important to realize that most of the parameters of the frequency and severity models are estimated and there is uncertainty associated with the parameter estimates. Any compound distribution estimate that is computed by using these uncertain parameter estimates is inherently uncertain. The aggregate loss sample that is simulated by using the mean estimates of the parameters is just one possible sample from the compound distribution. If information about parameter uncertainty is available, then it is recommended that you conduct parameter perturbation analysis that generates multiple samples of the compound distribution, in which each sample is simulated by using a set of perturbed parameter estimates. You can use the NPERTURBEDSAMPLES= option in the PROC HPCDM statement to specify the number of perturbed samples to be generated. The set of perturbed parameter estimates is created by making a random draw of the parameter values from their joint probability distribution. If you specify NPERTURBEDSAMPLES=*P*, then PROC HPCDM creates *P* sets of perturbed parameters and each set is used to simulate a full aggregate sample. The summary analysis of *P* such aggregate loss samples results in a set of *P* estimates for each summary statistic and percentile of the compound distribution. The mean and standard deviation of this set of *P* estimates quantify the uncertainty that is associated with the compound distribution.

The parameter uncertainty information is available in the form of either the variance-covariance matrix of the parameter estimates or standard errors of the parameters estimates. If the variance-covariance matrix

is available and is positive definite, then PROC HPCDM assumes that the joint probability distribution of the parameter estimates is a multivariate normal distribution, $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$, where the mean vector $\boldsymbol{\mu}$ is the set of point parameter estimates and Σ is the variance-covariance matrix. If the variance-covariance matrix is not available or is not positive definite, then PROC HPCDM assumes that each parameter has a univariate normal distribution, $\mathcal{N}(\mu, \sigma^2)$, where μ is the point estimate of the parameter and σ is the standard error of the parameter estimate.

To make the random draws from the multivariate normal distribution of all parameters or the univariate distributions of individual parameters, PROC HPCDM uses a pseudorandom number generator (PRNG) that is controlled by the PERTURBMETHOD= option as follows:

- PERTURBMETHOD=ASYNC is the legacy method that releases prior to SAS/ETS 15.1 used and is the default for the current release. This method allows each thread on each grid node to use a different PRNG for perturbation, which in fact is the same PRNG that the thread uses for making random draws from the severity or frequency distributions. Using different PRNGs and interleaving perturbation-related random draws with severity and count random draws causes each thread to use a different set of perturbed parameters while generating a subset of the same perturbed sample; in turn, this leads to a perturbed sample that is a heterogeneous collection of smaller perturbed samples, each of which is generated from a different compound distribution model.
- PERTURBMETHOD=SYNC method is the recommended method because it uses a single PRNG to perturb the parameters and synchronizes the set of perturbed parameters across all threads on all grid nodes. This makes each perturbed sample a homogeneous sample that corresponds to a single compound distribution model.

If you specify the severity models by using the SEVERITYEST= data set, then the point parameter estimates are expected to be available in the SEVERITYEST= data set in observations for which _TYPE_='EST', the standard errors are expected to be available in the SEVERITYEST= data set in observations for which _TYPE_='STDERR', and the variance-covariance matrix is expected to be available in the SEVERITYEST= data set in observations for which _TYPE_='COV'. If you use the SEVERITY procedure to create the SEVERITYEST= data set, then you need to specify the COVOUT option in the PROC SEVERITY statement to make the variance-covariance estimates available in the SEVERITYEST= data set.

If you specify the severity models by using the SEVERITYSTORE= item store, then you need to specify the OUTSTORE= option in the PROC SEVERITY statement to create that item store, which includes the point parameter estimates and standard errors by default. In addition, you need to specify the COVOUT option in the PROC SEVERITY statement to make the variance-covariance estimates available in the SEVERITYSTORE= item store.

For the frequency model, you must use the COUNTREG procedure to create the COUNTSTORE= item store, which always contains the point estimates, standard errors, and variance-covariance matrix of the parameters.

If you specify the ADJUSTEDSEVERITY= option in the PROC HPCDM statement, then a separate perturbation analysis is conducted for the distribution of the aggregate adjusted loss.

Descriptive Statistics

This section provides computational details for the descriptive statistics that are computed for each aggregate loss sample. You can also save these statistics in an OUTSUM= data set by specifying appropriate keywords in the OUTSUM statement.

This section gives specific details about the moment statistics. For more information about the methods of computing percentile statistics, see the description of the PCTLDEF= option in the UNIVARIATE procedure in the *Base SAS Procedures Guide: Statistical Procedures*.

Standard algorithms (Fisher 1973) are used to compute the moment statistics. The computational methods that the HPCDM procedure uses are consistent with those that other SAS procedures use for calculating descriptive statistics.

Mean

The sample mean is calculated as

$$\bar{y} = \frac{\sum_{i=1}^{n} y_i}{n}$$

where n is the size of the generated aggregate loss sample and y_i is the ith value of the aggregate loss.

Standard Deviation

The standard deviation is calculated as

$$s = \sqrt{\frac{1}{d}\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

where n is the size of the generated aggregate loss sample, y_i is the ith value of the aggregate loss, $\bar{y}$ is the sample mean, and d is the divisor controlled by the VARDEF= option in the PROC HPCDM statement:

$$d = \begin{cases} n-1 & \text{if VARDEF=DF (default)} \\ n & \text{if VARDEF=N} \end{cases}$$

Skewness

The sample skewness, which measures the tendency of the deviations to be larger in one direction than in the other, is calculated as

$$\frac{1}{d_s}\sum_{i=1}^{n}\left(\frac{y_i - \bar{y}}{s}\right)^3$$

where n is the size of the generated aggregate loss sample, y_i is the ith value of the aggregate loss, $\bar{y}$ is the sample mean, s is the sample standard deviation, and d_s is the divisor controlled by the VARDEF= option in the PROC HPCDM statement:

$$d_s = \begin{cases} \frac{(n-1)(n-2)}{n} & \text{if VARDEF=DF (default)} \\ n & \text{if VARDEF=N} \end{cases}$$

If VARDEF=DF, then n must be greater than 2.

The sample skewness can be positive or negative; it measures the asymmetry of the data distribution and estimates the theoretical skewness $\sqrt{\beta_1} = \mu_3 \mu_2^{-\frac{3}{2}}$, where μ_2 and μ_3 are the second and third central moments. Observations that are normally distributed should have a skewness near zero.

Kurtosis

The sample kurtosis, which measures the heaviness of tails, is calculated as in Table 3.2 depending on the value that you specify in the VARDEF= option.

Table 3.2 Formulas for Kurtosis

VARDEF= Value	Formula
DF (default)	$\frac{n(n+1)}{(n-1)(n-2)(n-3)} \sum_{i=1}^{n} \left(\frac{y_i - \bar{y}}{s} \right)^4 - \frac{3(n-1)^2}{(n-2)(n-3)}$
N	$\frac{1}{n} \sum_{i=1}^{n} \left(\frac{y_i - \bar{y}}{s} \right)^4 - 3$

In these formulas, n is the size of the generated aggregate loss sample, y_i is the ith value of the aggregate loss, $\bar{y}$ is the sample mean, and s is the sample standard deviation. If VARDEF=DF, then n must be greater than 3.

The sample kurtosis measures the heaviness of the tails of the data distribution. It estimates the adjusted theoretical kurtosis denoted as $\beta_2 - 3$, where $\beta_2 = \frac{\mu_4}{\mu_2{}^2}$ and μ_4 is the fourth central moment. Observations that are normally distributed should have a kurtosis near zero.

Input Specification

PROC HPCDM accepts the DATA= and SEVERITYEST= data sets and the COUNTSTORE= and SEVERITYSTORE= item stores as input. This section details the information that they are expected to contain.

DATA= Data Set

If you specify the BY statement, then the DATA= data set must contain all the BY variables that you specify in the BY statement and the data set must be sorted by the BY variables unless the BY statement includes the NOTSORTED option.

If the severity models in the SEVERITYEST= data set or the SEVERITYSTORE= item store contain any scale regressors, then all those regressors must be present in the DATA= data set.

If you specify the programming statements to compute an aggregate adjusted loss, and if your specified ADJUSTEDSEVERITY= symbol depends on severity adjustment variables, then the DATA= data set must contain all such variables.

The rest of the contents of the DATA= data set depends on whether you specify the EXTERNALCOUNTS statement. If you specify the EXTERNALCOUNTS statement, then the DATA= data set is expected to contain the COUNT= and ID= variables that you specify in the EXTERNALCOUNTS statement. If you do not specify the EXTERNALCOUNTS statement, then the DATA= data set must contain all the regressors, including zero model regressors, that are present in the count model that the COUNTSTORE= item store contains.

You do not need to specify the DATA= data set if *all* the following conditions are true:

- You do not specify the BY statement.
- You specify the severity models such that none of them are scale regression models.
- You do not specify the EXTERNALCOUNTS statement.
- You specify a COUNTSTORE= item store such that the count model contains no count regressors.
- Your severity adjustment programming statements, if you specify any, do not use any external input.

If you specify the BY statement, then PROC HPCDM analyzes only the BY groups that are present in the input source of the severity and count models. If neither the severity models nor the count models contain regression effects, then the DATA= data set must contain BY variables and one row for each BY group that you want PROC HPCDM to analyze.

SEVERITYEST= Data Set

The SEVERITYEST= data set is expected to contain the parameter estimates of the severity models. This is a required data set; you must specify it whenever you use PROC HPCDM.

The SEVERITYEST= data set must have the same format as the OUTEST= data set that is created by the SEVERITY procedure. For more information, see the description of the OUTEST= data set in the SEVERITY procedure in the *SAS/ETS User's Guide*.

If you specify the BY statement, then the SEVERITYEST= data set must contain all the BY variables that you specify in the BY statement. If you do not specify the NOTSORTED option in the BY statement, then the SEVERITYEST= data set must be sorted by the BY variables.

SEVERITYSTORE= Item Store

The SEVERITYSTORE= item store is expected to be created by using the OUTSTORE= option in a PROC SEVERITY statement. For more information, see the description of the OUTSTORE= option in the SEVERITY procedure in the *SAS/ETS User's Guide*.

You must specify this item store when you do not specify the SEVERITYEST= data set. Also, if your severity model is a scale regression model that contains classification or interaction effects, then you cannot use the SEVERITYEST= data set. You must specify such severity models by specifying the SEVERITYSTORE= item store.

If you specify the BY statement, then the SEVERITYSTORE= item store must have been created by using a PROC SEVERITY step that uses an identical BY statement.

COUNTSTORE= Item Store

The COUNTSTORE= item store is expected to be created by using the STORE statement in the COUNTREG procedure. You must specify the COUNTSTORE= item store when you do not specify the EXTERNAL-COUNTS statement. For more information, see the description of the STORE statement in the COUNTREG procedure in the *SAS/ETS User's Guide*.

If you specify the BY statement, then the COUNTSTORE= item store must have been created by using a PROC COUNTREG step that uses an identical BY statement.

Output Data Sets

PROC HPCDM writes the output data sets that you specify in the OUT= option of the OUTPUT and OUTSUM statements. The contents of these output data sets are described in the sections "OUTSAMPLE= Data Set" on page 87 and "OUTSUM= Data Set" on page 87, respectively.

OUTSAMPLE= Data Set

The OUTSAMPLE= data set records the full sample of the aggregate loss and aggregate adjusted loss.

If you specify the BY statement, then the data are organized in BY groups and the data set contains variables that you specify in the BY statement. In addition, the OUTSAMPLE= data set contains the following variables:

SEVERITYMODEL
: indicates the name of the severity distribution model.

COUNTMODEL
: indicates the name of the count model. If you specify the EXTERNALCOUNTS statement, then the value of this variable is "_EXTERNAL_". If you specify the COUNTSTORE= option, then the value of this variable is "_COUNTSTORE_".

<unadjusted sample variable>
: indicates the value of the unadjusted aggregate loss. The name of this variable is the value of the SAMPLEVAR= option in the OUTPUT statement. If you do not specify the SAMPLEVAR= option, then the variable is named _AGGSEV_.

<adjusted sample variable>
: indicates the value of the adjusted aggregate loss. This variable is created only when you specify the programming statements and the ADJUSTEDSEVERITY= option in the PROC HPCDM statement. The name of this variable is the value of the ADJSAMPLEVAR= option in the OUTPUT statement. If you do not specify the ADJSAMPLEVAR= option, then the variable is named _AGGADJSEV_.

DRAWID
: indicates the identifier for the perturbed sample. This variable is created only when you specify the NPERTURBEDSAMPLES= option in the PROC HPCDM statement. The value of this variable identifies the perturbed sample. A value of 0 for the _DRAWID_ variable indicates an unperturbed sample.

OUTSUM= Data Set

The OUTSUM= data set records the summary statistics and percentiles of the compound distributions of aggregate loss and aggregate adjusted loss. Only the estimates that you request in the OUTSUM statement are written to the OUTSUM= data set. For more information about the method of naming the variables that correspond to the summary statistics or percentiles, see the description of the OUTSUM statement.

If you specify the BY statement, then the data are organized in BY groups and the data set contains variables that you specify in the BY statement. In addition, the OUTSUM= data set contains the following variables:

SEVERITYMODEL
: indicates the name of the severity distribution model.

COUNTMODEL
: indicates the name of the count model. If you specify the EXTERNALCOUNTS statement, then the value of this variable is "_EXTERNAL_". If you specify the COUNTSTORE= option, then the value of this variable is "_COUNTSTORE_".

SAMPLEVAR
: indicates the name of the aggregate loss sample. For an unadjusted sample, the value of the variable is the value of the SAMPLEVAR= option that you specify in the OUTPUT statement or the default value of _AGGSEV_. For an adjusted sample, the value of the variable is the value of the ADJSAMPLEVAR= option that you specify in the OUTPUT statement or the default value of _AGGADJSEV_.

DRAWID
: indicates the identifier for the perturbed sample. This variable is created only when you specify the NPERTURBEDSAMPLES= option in the PROC HPCDM statement. The value of this variable identifies the perturbed sample. A value of 0 for _DRAWID_ indicates an unperturbed sample.

Displayed Output

The HPCDM procedure optionally produces displayed output by using the Output Delivery System (ODS). All output is controlled by the PRINT= option in the PROC HPCDM statement. Table 3.3 relates the PRINT= options to ODS tables.

Table 3.3 ODS Tables Produced in PROC HPCDM

ODS Table Name	Description	Option
CompoundInfo	Compound distribution information	Default
DataSummary	Input data summary	Default
Percentiles	Percentiles of the aggregate loss sample	PRINT=PERCENTILES
PerformanceInfo	Execution environment information that pertains to the computational performance	Default
PerturbedPctlSummary	Perturbation analysis of percentiles	PRINT=PERTURBSUMMARY and NPERTURBEDSAMPLES > 0
PerturbedSummary	Perturbation analysis of summary statistics	PRINT=PERTURBSUMMARY and NPERTURBEDSAMPLES > 0
SummaryStatistics	Summary statistics of the aggregate loss sample	PRINT=SUMMARYSTATISTICS
Timing	Timing information for various computational stages of the procedure	DETAILS (PERFORMANCE statement)

PRINT= Option

This section provides detailed descriptions of the tables that are displayed by using different PRINT= options.

- If you do not specify the PRINT= option and if you do not specify the NOPRINT or PRINT=NONE options, then by default PROC HPCDM produces the CompoundInfo, DataSummary, and SummaryStatistics ODS tables.

  The "Compound Distribution Information" table (ODS name: CompoundInfo) displays the information about the severity and count models.

  The "Input Data Summary" table (ODS name: DataSummary) is displayed when you specify the DATA= data set. The table displays the total number of observations and the valid number of observations in the data set. If you specify the EXTERNALCOUNTS statement, then the table also displays the number of replications and total number of loss events across all replications.

- If you specify PRINT=PERCENTILES, the "Percentiles" table (ODS name: Percentiles) is displayed for the distribution of the aggregate loss. The table contains estimates of all the predefined percentiles in addition to the percentiles that you request in the OUTSUM statement.

  If you specify the programming statements and the ADJUSTEDSEVERITY= symbol, then an additional table is displayed for the distribution of the aggregate adjusted loss. This table also contains estimates of all the predefined percentiles in addition to the percentiles that you request in the OUTSUM statement.

- If you specify PRINT=PERTURBSUMMARY, two tables are displayed for the distribution of the aggregate loss. The "Perturbed Summary Statistics" table (ODS name: PerturbedSummary) displays the summary of the effect of perturbing model parameters on the following five summary statistics of the distribution: mean, standard deviation, variance, skewness, and kurtosis. The "Perturbed Percentiles" table (ODS name: PerturbedPctlSummary) displays the perturbation summary for all the predefined percentiles in addition to the percentiles that you request in the OUTSUM statement.

  The tables are displayed only if you specify a value greater than 0 for the NPERTURBEDSAMPLES= option.

  If you specify a value of P for the NPERTURBEDSAMPLES= option, then for each summary statistic and percentile, an average and standard error of the set of P values of that summary statistic or percentile are displayed in the respective perturbation summary tables.

  If you specify the programming statements and the ADJUSTEDSEVERITY= symbol, then additional perturbation summary tables are displayed for the distribution of the aggregate adjusted loss.

- If you specify PRINT=SUMMARYSTATISTICS, the "Summary Statistics" table (ODS name: SummaryStatistics) is displayed for the distribution of the aggregate loss. The table contains estimates of the following summary statistics: the number of observations in the sample, maximum value in the sample, minimum value in the sample, mean, median, standard deviation, interquartile range, variance, skewness, and kurtosis.

  If you specify the programming statements and the ADJUSTEDSEVERITY= symbol, then an additional table of summary statistics is displayed for the distribution of the aggregate adjusted loss.

Performance Information

The "Performance Information" table (ODS name: PerformanceInfo) is produced by default. It displays information about the execution mode. For single-machine mode, the table displays the number of threads that are used. For distributed mode, the table displays the grid mode (symmetric or asymmetric), the number of compute nodes, and the number of threads per node.

If you specify the DETAILS option in the PERFORMANCE statement, PROC HPCDM also produces a "Timing" table (ODS name: Timing) that displays elapsed times (absolute and relative) for the main tasks of the procedure.

ODS Graphics

Statistical procedures use ODS Graphics to create graphs as part of their output. ODS Graphics is described in detail in Chapter 21, "Statistical Graphics Using ODS" (*SAS/STAT User's Guide*).

Before you create graphs, ODS Graphics must be enabled (for example, with the ODS GRAPHICS ON statement). For more information about enabling and disabling ODS Graphics, see the section "Enabling and Disabling ODS Graphics" in that chapter.

The overall appearance of graphs is controlled by ODS styles. Styles and other aspects of using ODS Graphics are discussed in the section "A Primer on ODS Statistical Graphics" in that chapter.

This section describes the use of ODS for creating graphics with the HPCDM procedure.

NOTE: If you request simulation of an aggregate loss sample of large size, either by specifying a large value for the NREPLICATES= option or by including a large number of replicates in the DATA= data set that you specify in conjunction with the EXTERNALCOUNTS statement, then it is recommended that you not request any plots, because creating plots that have large numbers of points can require a very large amount of hardware resources and can take a very long time. You can disable the generation of plots either by submitting the ODS GRAPHICS OFF statement before submitting the PROC HPCDM step or by specifying the PLOTS=NONE option in the PROC HPCDM statement. It is recommended that you request plots only when the sample size is less than 100,000.

ODS Graph Names

PROC HPCDM assigns a name to each graph that it creates by using ODS. You can use these names to selectively refer to the graphs. The names are listed in Table 3.4.

Table 3.4 ODS Graphics Produced by PROC HPCDM

ODS Graph Name	Plot Description	PLOTS= Option
ConditionalDensityPlot	Conditional density plot	CONDITIONALDENSITY
DensityPlot	Probability density function plot	DENSITY
EDFPlot	Empirical distribution function plot	EDF

Conditional Density Plot

The conditional density plot helps you visually analyze two or three regions of the compound distribution by displaying a density function estimate that is conditional on the values of the aggregate loss that fall in those regions. You can specify the region boundaries in terms of quantiles by using the LEFTQ= and RIGHTQ= suboptions of the PLOTS=CONDITIONALDENSITY option. This is especially useful if you want to see the distribution of aggregate loss values in the right- and left-tail regions.

If you specify the programming statements and the ADJUSTEDSEVERITY= symbol, then a separate set of conditional density plots are displayed for the aggregate adjusted loss.

Probability Density Function Plot

The probability density function (PDF) plot shows the nonparametric estimates of the PDF of the aggregate loss distribution. This plot includes histogram and kernel density estimates.

If you specify the programming statements and the ADJUSTEDSEVERITY= symbol, then a separate density plot is displayed for the aggregate adjusted loss.

Empirical Distribution Function Plot

The empirical density function (EDF) plot shows the nonparametric estimate of the cumulative distribution function of the aggregate loss distribution. You can specify the ALPHA= suboption of the PLOTS=EDF option to request that the upper and lower confidence limits be plotted for each EDF estimate. By default, the confidence interval is not plotted.

If you specify the programming statements and the ADJUSTEDSEVERITY= symbol, then a separate EDF plot is displayed for the aggregate adjusted loss.

Examples: HPCDM Procedure

Example 3.1: Estimating the Probability Distribution of Insurance Payments

The primary outcome of running PROC HPCDM is the estimate of the compound distribution of aggregate loss, given the distributions of frequency and severity of the individual losses. This aggregate loss is often referred to as the ground-up loss. If you are an insurance company or a bank, you are also interested in acting on the ground-up loss by computing an entity that is derived from the ground-up loss. For example, you might want to estimate the distribution of the amount that you are expected to pay for the losses or the distribution of the amount that you can offload onto another organization, such as a reinsurance company. PROC HPCDM enables you to specify a severity adjustment program, which is a sequence of SAS programming statements that adjust the severity of the individual loss event to compute the entity of interest. Your severity adjustment program can use external information that is recorded as variables in the observations of the DATA= data set in addition to placeholder symbols for information that PROC HPCDM generates internally, such as the severity of the current loss event (_SEV_) and the sum of the adjusted severity values of the events that have been simulated thus far for the current sample point (_CUMADJSEV_). If you are doing a scenario analysis such that a scenario contains more than one observation, then you can also access the cumulative

severity and cumulative adjusted severity for the current observation by using the _CUMSEVFOROBS_ and _CUMADJSEVFOROBS_ symbols.

This example continues the example of the section "Scenario Analysis" on page 44 to illustrate how you can estimate the distribution of the aggregate amount that is paid to a group of policyholders. Let the amount that is paid to an individual policyholder be computed by using what is usually referred to as a *disappearing deductible* (Klugman, Panjer, and Willmot 1998, Ch. 2). If X denotes the ground-up loss that a policyholder incurs, d denotes the lower limit on the deductible, d' denotes the upper limit on the deductible, and u denotes the limit on the total payments that are made to a policyholder in a year, then Y, the amount that is paid to the policyholder for each loss event, is defined as follows:

$$Y = \begin{cases} 0 & X \le d \\ d' \frac{X-d}{d'-d} & d < X \le d' \\ X & d' < X \le u \\ u & X > u \end{cases}$$

You can encode this logic by using a set of SAS programming statements.

Extend the Work.GroupOfPolicies data set in the example in the section "Scenario Analysis" on page 44 to include the following three additional variables for each policyholder: LowDeductible to record d, HighDeductible to record d', and Limit to record u. The data set contains the observations as shown in Output 3.1.1.

Output 3.1.1 Scenario Analysis Data for Multiple Policyholders with Policy Provisions

policyholderId	age	gender	carType	annualMiles	education	carSafety	income
1	1.18	2	1	2.2948	3	0.99532	1.59870
2	0.66	2	2	2.8148	1	0.05625	0.67539
3	0.82	1	2	1.6130	2	0.84146	1.05940
4	0.44	1	1	1.2280	3	0.14324	0.24110
5	0.44	1	1	0.9670	2	0.08656	0.65979

lowDeductible	highDeductible	limit	annualLimit
400	1400	7500	10000
300	1300	2500	20000
100	1100	5000	10000
300	800	5000	20000
100	1100	5000	20000

The following PROC HPCDM step estimates the compound distributions of the aggregate loss and the aggregate amount that is paid to the group of policyholders in the Work.GroupOfPolicies data set by using the count model that is stored in the Work.CountregModel item store and the lognormal severity model that is stored in the Work.SevRegEst data set:

```
/* Simulate the aggregate loss distribution and aggregate adjusted
   loss distribution for the scenario with multiple policyholders */
proc hpcdm data=groupOfPolicies nreplicates=10000 seed=13579 print=all
           countstore=work.countregmodel severityest=work.sevregest
           plots=(edf pdf) nperturbedSamples=50
           adjustedseverity=amountPaid;
   severitymodel logn;
```

```
   if (_sev_ <= lowDeductible) then
      amountPaid = 0;
   else do;
      if (_sev_ <= highDeductible) then
         amountPaid = highDeductible *
            (_sev_-lowDeductible)/(highDeductible-lowDeductible);
      else
         amountPaid = MIN(_sev_, limit); /* imposes per-loss payment limit */
   end;
run;
```

The preceding step uses a severity adjustment program to compute the value of the symbol AmountPaid and specifies that symbol in the ADJUSTEDSEVERITY= option in the PROC HPCDM step. The program is executed for each simulated loss event. The PROC HPCDM supplies your program with the value of the severity in the _SEV_ placeholder symbol.

The "Sample Summary Statistics" table in Output 3.1.2 shows the summary statistics of the compound distribution of the aggregate ground-up loss. The "Adjusted Sample Summary Statistics" table shows the summary statistics of the compound distribution of the aggregate AmountPaid. The average aggregate payment is about 4,361, as compared to the average aggregate ground-up loss of 5,906.

Output 3.1.2 Summary Statistics of Compound Distributions of the Total Loss and Total Amount Paid

The HPCDM Procedure
Severity Model: Logn
Count Model: NegBin(p=2)

Compound Distribution Information	
Severity Model	Lognormal Distribution
Scale Model Regressors	carType carSafety income
Count Model	NegBin(p=2) Model in Item Store WORK.COUNTREGMODEL

Sample Summary Statistics			
Mean	5906.2	**Median**	4727.7
Standard Deviation	4801.7	**Interquartile Range**	5227.0
Variance	23056465.3	**Minimum**	0
Skewness	2.25016	**Maximum**	64811.8
Kurtosis	10.01578	**Sample Size**	10000

Adjusted Sample Summary Statistics			
Mean	4361.0	**Median**	3762.9
Standard Deviation	3181.7	**Interquartile Range**	4133.6
Variance	10123000.5	**Minimum**	0
Skewness	1.11692	**Maximum**	23657.4
Kurtosis	1.64518	**Sample Size**	10000

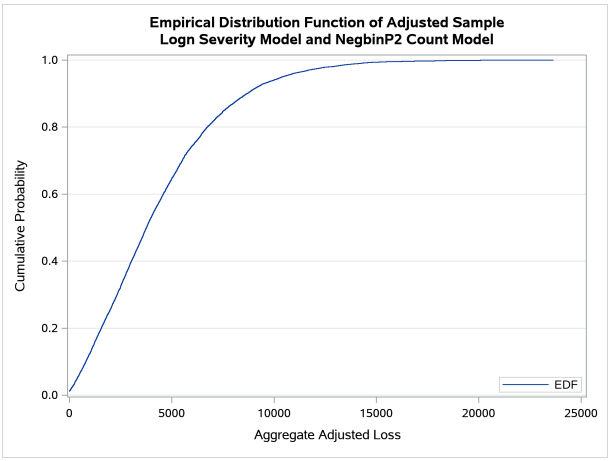
The perturbation summary of the distribution of AmountPaid is shown in Output 3.1.3. It shows that you can expect to pay a median of 3,796 $\pm$ 271 to this group of five policyholders in a year. Also, if the 99.5th percentile defines the worst case, then you can expect to pay 15,573 $\pm$ 859 in the worst-case.

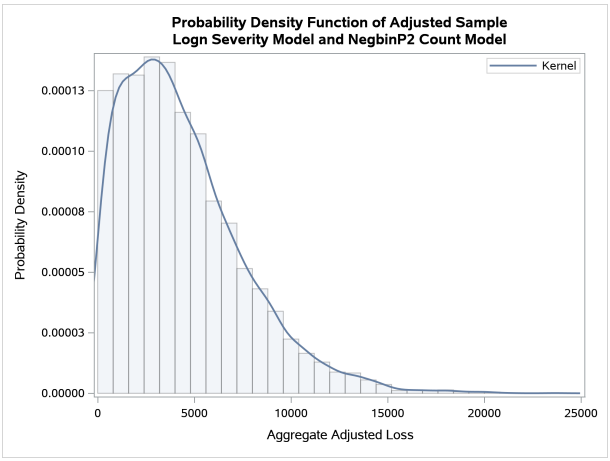
Output 3.1.3 Perturbation Summary of the Total Amount Paid

Adjusted Sample Percentile Perturbation Analysis		
Percentile	Estimate	Standard Error
1	0.94036	6.15322
5	386.17494	65.79399
25	1988.9	188.59978
50	3796.0	271.27093
75	6153.8	393.13966
95	10435.2	621.20756
99	14108.5	827.74239
99.5	15573.1	858.66726
Number of Perturbed Samples = 50		
Size of Each Sample = 10000		

The empirical distribution function (EDF) and probability density function plots of the aggregate adjusted loss are shown in Output 3.1.4. Both plots indicate a heavy-tailed distribution of the total amount paid.

Output 3.1.4 PDF and EDF Plots of the Compound Distribution of the Total Amount Paid





Now consider that, in the future, you want to modify the policy provisions to add a limit on the total amount of payment that is made to an individual policyholder in one year and to impose a group limit of 15,000 on the total amount of payments that are made to the group as a whole in one year. You can analyze the effects of these modified policy provisions on the distribution of the aggregate paid amount by recording the individual policyholder's annual limit in the AnnualLimit variable of the input data set and then modifying your severity adjustment program by using the placeholder symbols _CUMADJSEVFOROBS_ and _CUMADJSEV_ as shown in the following PROC HPCDM step:

```
/* Simulate the aggregate loss distribution and aggregate adjusted
   loss distribution for the modified set of policy provisions */
proc hpcdm data=groupOfPolicies nreplicates=10000 seed=13579 print=all
           countstore=work.countregmodel severityest=work.sevregest
           plots=none nperturbedSamples=50
           adjustedseverity=amountPaid;
```

```
   severitymodel logn;

   if (_sev_ <= lowDeductible) then
      amountPaid = 0;
   else do;
      if (_sev_ <= highDeductible) then
         amountPaid = highDeductible *
            (_sev_-lowDeductible)/(highDeductible-lowDeductible);
      else
         amountPaid = MIN(_sev_, limit); /* imposes per-loss payment limit */

      /* impose policyholder's annual limit */
      amountPaid = MIN(amountPaid, MAX(0,annualLimit - _cumadjsevforobs_));

      /* impose group's annual limit */
      amountPaid = MIN(amountPaid, MAX(0,15000 - _cumadjsev_));
   end;
run;
```

The results of the perturbation analysis for these modified policy provisions are shown in Output 3.1.5. When compared to the results of Output 3.1.3, the additional policy provisions of restricting the total payment to the policyholder and the group have reduced the median payment slightly, but the provisions have reduced the worst-case payment (99.5th percentile) to 14,755 $\pm$ 392 from 15,573 $\pm$ 859.

Output 3.1.5 Perturbation Summary of the Total Amount Paid for Modified Policy Provisions

The HPCDM Procedure
Severity Model: Logn
Count Model: NegBin(p=2)

Adjusted Sample Percentile Perturbation Analysis

Percentile	Estimate	Standard Error
1	0.46949	2.23897
5	382.04931	56.96535
25	1953.8	167.85926
50	3733.5	250.62840
75	6054.8	348.38707
95	10272.4	520.34620
99	13728.9	631.17302
99.5	14754.8	392.25491

Number of Perturbed Samples = 50

Size of Each Sample = 10000

Example 3.2: Using Externally Simulated Count Data

The COUNTREG procedure enables you to estimate count regression models that are based on the most commonly used discrete distributions, such as the Poisson, negative binomial (both $p = 1$ and $p = 2$), and Conway-Maxwell-Poisson distributions. PROC COUNTREG also enables you to fit zero-inflated models that are based on Poisson, negative binomial ($p = 2$), and Conway-Maxwell-Poisson distributions. However,

there might be situations in which you want to use some other method of fitting count regression models. For example, if you are modeling the number of loss events that are incurred by two financial instruments such that there is some dependency between the two, then you might use some multivariate frequency modeling methods and simulate the counts for each instrument by using the dependency structure between the count model parameters of the two instruments. As another example, you might want to use different types of count models for different BY groups in your data; this is not possible in PROC COUNTREG. So you need to simulate the counts for such BY groups externally. PROC HPCDM enables you to supply externally simulated counts by using the EXTERNALCOUNTS statement. PROC HPCDM then does not need to simulate the counts internally; it simulates only the severity of each loss event by using the severity model estimates that you specify in the SEVERITYEST= data set or the SEVERITYSTORE= item store. The simulation process is described and illustrated in the section "Simulation with External Counts" on page 71.

Consider that you are a bank, and as part of quantifying your operational risk, you want to estimate the aggregate loss distributions for two lines of business, retail banking and commercial banking, by using some key risk indicators (KRIs). Assume that your model fitting and model selection process has determined that the Poisson regression model and negative binomial regression model are the best-fitting count models for number of loss events that are incurred in the retail banking and commercial banking businesses, respectively. Let CorpKRI1, CorpKRI2, CbKRI1, CbKRI2, and CbKRI3 be the KRIs that are used in the count regression model of the commercial banking business, and let CorpKRI1, RbKRI1, and RbKRI2 be the KRIs that are used in the count regression model of the retail banking business. Some examples of corporate-level KRIs (CorpKRI1 and CorpKRI2 in this example) are the ratio of temporary to permanent employees and the number of security breaches that are reported during a year. Some examples of KRIs that are specific to the commercial banking business (CbKRI1, CbKRI2, and CbKRI3 in this example) are number of credit defaults, proportion of financed assets that are movable, and penalty claims against your bank because of processing delays. Some examples of KRIs that are specific to the retail banking business (RbKRI1 and RbKRI2 in this example) are number of credit cards that are reported stolen, fraction of employees who have not undergone fraud detection training, and number of forged drafts and checks that are presented in a year.

Let the severity of each loss event in the commercial banking business be dependent on two KRIs, CorpKRI1 and CbKRI2. Let the severity of each loss event in the retail banking business be dependent on three KRIs, CorpKRI2, RbKRI1, and RbKRI3. Note that for each line of business, the set of KRIs that are used for the severity model is different from the set of KRIs that are used for the count model, although there is some overlap between the two sets. Further, the severity model for retail banking includes a new regressor (RbKRI3) that is not used for any of the count models. Such use of different sets of KRIs for count and severity models is typical of real-world applications.

Let the parameter estimates of the negative binomial and Poisson regression models, as determined by PROC COUNTREG, be available in the Work.CountEstEx2NB2 and Work.CountEstEx2Poisson data sets, respectively. These data sets are produced by using the OUTEST= option in the respective PROC COUNTREG statements. Let the parameter estimates of the best-fitting severity models, as determined by PROC SEVERITY, be available in the Work.SevEstEx2Best data set. You can find the code to prepare these data sets in the PROC HPCDM sample program *hcdmex02.sas*.

Now, consider that you want to estimate the distribution of the aggregate loss for a scenario, which is represented by a specific set of KRI values. The following DATA step illustrates one such scenario:

```
/* Generate a scenario data set for a single operating condition */
data singleScenario (keep=corpKRI1 corpKRI2 cbKRI1 cbKRI2 cbKRI3
                         rbKRI1 rbKRI2 rbKRI3);
   array x{8} corpKRI1 corpKRI2 cbKRI1 cbKRI2 cbKRI3 rbKRI1 rbKRI2 rbKRI3;
   call streaminit(5151);
```

```
   do i=1 to dim(x);
      x(i) = rand('NORMAL');
   end;
   output;
run;
```

The Work.SingleScenario data set contains all the KRIs that are included in the count and severity models of both business lines. Note that if you standardize or scale the KRIs while fitting the count and severity models, then you must apply the same standardization or scaling method to the values of the KRIs that you specify in the scenario. In this particular example, all KRIs are assumed to be standardized.

The following DATA step uses the scenario in the Work.SingleScenario data set to simulate 10,000 replications of the number of loss events that you might observe for each business line and writes the simulated counts to the NumLoss variable of the Work.LossCounts1 data set:

```
/* Simulate multiple replications of the number of loss events that
   you can expect in the scenario being analyzed */
data lossCounts1 (keep=line corpKRI1 corpKRI2 cbKRI2 rbKRI1 rbKRI3 numloss);
   array cxR{3} corpKRI1 rbKRI1 rbKRI2;
   array cbetaR{4} _TEMPORARY_;
   array cxC{5} corpKRI1 corpKRI2 cbKRI1 cbKRI2 cbKRI3;
   array cbetaC{6} _TEMPORARY_;

   retain theta;
   if _n_ = 1 then do;
      call streaminit(5151);
      * read count model estimates *;
      set countEstEx2NB2(where=(line='CommercialBanking' and _type_='PARM'));
      cbetaC(1) = Intercept;
      do i=1 to dim(cxC);
         cbetaC(i+1) = cxC(i);
      end;
      alpha = _Alpha;
      theta = 1/alpha;

      set countEstEx2Poisson(where=(line='RetailBanking' and _type_='PARM'));
      cbetaR(1) = Intercept;
      do i=1 to dim(cxR);
         cbetaR(i+1) = cxR(i);
      end;
   end;

   set singleScenario;
   do iline=1 to 2;
      if (iline=1) then line = 'CommercialBanking';
      else line = 'RetailBanking';
      do repid=1 to 10000;
         * draw from count distribution *;
         if (iline=1) then do;
            xbeta = cbetaC(1);
            do i=1 to dim(cxC);
               xbeta = xbeta + cxC(i) * cbetaC(i+1);
            end;
            Mu = exp(xbeta);
```

```
            p = theta/(Mu+theta);
            numloss = rand('NEGB',p,theta);
         end;
         else do;
            xbeta = cbetaR(1);
            do i=1 to dim(cxR);
               xbeta = xbeta + cxR(i) * cbetaR(i+1);
            end;
            numloss = rand('POISSON', exp(xbeta));
         end;
         output;
      end;
   end;
run;
```

The Work.LossCounts1 data set contains the NumLoss variable in addition to the KRIs that are used by the severity regression model, which are needed by PROC HPCDM to simulate the aggregate loss.

By default, PROC HPCDM computes an aggregate loss distribution by using each of the severity models that you specify in the SEVERITYMODEL statement. However, you can restrict PROC HPCDM to use only a subset of the severity models for a given BY group by modifying the SEVERITYEST= data set to include only the estimates of the desired severity models in each BY group, as illustrated in the following DATA step:

```
/* Keep only the best severity model for each business line
   and set coefficients of unused regressors in each model to 0 */
data sevestEx2Best;
    set sevestEx2;
    if ((line = 'CommercialBanking' and _model_ = 'Logn')) then do;
        corpKRI2 = 0; rbKRI1 = 0; rbKRI3 = 0;
        output;
    end;
    else if ((line = 'RetailBanking' and _model_ = 'Gamma')) then do;
        corpKRI1 = 0; cbKRI2 = 0;
        output;
    end;
run;
```

Note that the preceding DATA step also sets the coefficients of the unused regressors in each model to 0. This is important because PROC HPCDM uses all the regressors that it detects from the SEVERITYEST= data set for each severity model.

Now, you are ready to estimate the aggregate loss distribution for each line of business by submitting the following PROC HPCDM step, in which you specify the EXTERNALCOUNTS statement to request that external counts in the NumLoss variable of the DATA= data set be used for simulation of the aggregate loss:

```
/* Estimate the distribution of the aggregate loss for both
   lines of business by using the externally simulated counts */
proc hpcdm data=lossCounts1 seed=13579 print=all
           severityest=sevestEx2Best;
   by line;
   externalcounts count=numloss;
   severitymodel logn gamma;
run;
```

Each observation in the Work.LossCounts1 data set represents one replication of the external counts simulation process. For each such replication, the preceding PROC HPCDM step makes as many severity draws from the severity distribution as the value of the NumLoss variable and adds the severity values from those draws to compute one sample point of the aggregate loss. The severity distribution that is used for making the severity draws has a scale parameter value that is decided by the KRI values in the given observation and the regression parameter values that are read from the Work.SevEstEx2Best data set.

The summary statistics and percentiles of the aggregate loss distribution for the commercial banking business, which uses the lognormal severity model, are shown in Output 3.2.1. The "Input Data Summary" table indicates that each of the 10,000 observations in the BY group is treated as one replication and that there are a total of 19,028 loss events produced by all the replications together. For the scenario in the Work.SingleScenario data set, you can expect the commercial banking business to incur an average aggregate loss of 643 units, as shown in the "Sample Summary Statistics" table, and the chance that the loss will exceed 4,762 units is 0.5%, as shown in the "Sample Percentiles" table.

Output 3.2.1 Aggregate Loss Summary for Commercial Banking Business

The HPCDM Procedure

line=CommercialBanking

Input Data Summary	
Name	WORK.LOSSCOUNTS1
Observations	10000
Valid Observations	10000
Replications	10000
Total Count	19028

line=CommercialBanking

Sample Summary Statistics			
Mean	643.24599	**Median**	363.33564
Standard Deviation	843.56959	**Interquartile Range**	842.66329
Variance	711609.7	**Minimum**	0
Skewness	2.66370	**Maximum**	8807.3
Kurtosis	11.00174	**Sample Size**	10000

line=CommercialBanking

Sample Percentiles	
Percentile	**Value**
1	0
5	0
25	51.29272
50	363.33564
75	893.95601
95	2291.3
99	3990.7
99.5	4762.4
Percentile Method = 5	

For the retail banking business, which uses the gamma severity model, the "Sample Percentiles" table in Output 3.2.2 indicates that the median operational loss of that business is about 69 units and the chance that the loss will exceed 391 units is about 1%.

Output 3.2.2 Aggregate Loss Percentiles for Retail Banking Business

line=RetailBanking

Sample Percentiles	
Percentile	**Value**
1	0
5	0
25	0
50	69.26829
75	140.27686
95	273.61767
99	391.15896
99.5	439.23312
Percentile Method = 5	

When you conduct the simulation and estimation for a scenario that contains only one observation, you assume that the operating environment does not change over the period of time that is being analyzed. That assumption might be valid for shorter durations and stable business environments, but often the operating environments change, especially if you are estimating the aggregate loss over a longer period of time. So you might want to include in your scenario all the possible operating environments that you expect to see during the analysis time period. Each environment is characterized by its own set of KRI values. For example, the operating conditions might change from quarter to quarter, and you might want to estimate the aggregate loss distribution for the entire year. You start the estimation process for such scenarios by creating a scenario data set. The following DATA step creates the Work.MultiConditionScenario data set, which consists of four operating environments, one for each quarter:

```
/* Generate a scenario data set for multiple operating conditions */
data multiConditionScenario (keep=opEnvId corpKRI1 corpKRI2
      cbKRI1 cbKRI2 cbKRI3 rbKRI1 rbKRI2 rbKRI3);
   array x{8} corpKRI1 corpKRI2 cbKRI1 cbKRI2 cbKRI3 rbKRI1 rbKRI2 rbKRI3;
   call streaminit(5151);
   do opEnvId=1 to 4;
      do i=1 to dim(x);
         x(i) = rand('NORMAL');
      end;
      output;
   end;
run;
```

All four observations of the Work.MultiConditionScenario data set together form one scenario. When simulating the external counts for such multi-entity scenarios, one replication consists of the possible number of loss events that can occur as a result of each of the four operating environments. In any given replication, some operating environments might not produce any loss event or all four operating environments might produce some loss events. Assume that you use a DATA step to create the Work.LossCounts2 data set that contains, for each business line, 10,000 replications of the loss counts and that you identify each replication

by using the RepId variable. You can find the DATA step code to prepare the Work.LossCounts2 data set in the PROC HPCDM sample program *hcdmex02.sas*.

Output 3.2.3 shows some observations of the Work.LossCounts2 data set for each business line. For the first replication (RepId=1) of the commercial banking business, only operating environments 3 and 4 incur loss events, whereas the other environments incur no loss events. For the second replication (RepId=2), all operating environments incur at least one loss event. For the first replication (RepId=1) of the retail banking business, operating environments 2, 3, and 4 incur two, one, and three loss events, respectively.

Output 3.2.3 Snapshot of the External Counts Data with Replication Identifier

line	opEnvId	corpKRI1	corpKRI2	cbKRI2	rbKRI1	rbKRI3	repid	numloss
CommercialBanking	1	0.45224	0.40661	-0.33680	-1.08692	-2.20557	1	0
CommercialBanking	2	-0.03799	0.98670	-0.03752	1.94589	1.22456	1	0
CommercialBanking	3	-0.29120	-0.45239	0.98855	-0.37208	-1.51534	1	3
CommercialBanking	4	0.87499	-0.67812	-0.04839	-1.44881	0.78221	1	1
CommercialBanking	1	0.45224	0.40661	-0.33680	-1.08692	-2.20557	2	2
CommercialBanking	2	-0.03799	0.98670	-0.03752	1.94589	1.22456	2	5
CommercialBanking	3	-0.29120	-0.45239	0.98855	-0.37208	-1.51534	2	12
CommercialBanking	4	0.87499	-0.67812	-0.04839	-1.44881	0.78221	2	12
RetailBanking	1	0.45224	0.40661	-0.33680	-1.08692	-2.20557	1	0
RetailBanking	2	-0.03799	0.98670	-0.03752	1.94589	1.22456	1	2
RetailBanking	3	-0.29120	-0.45239	0.98855	-0.37208	-1.51534	1	1
RetailBanking	4	0.87499	-0.67812	-0.04839	-1.44881	0.78221	1	3
RetailBanking	1	0.45224	0.40661	-0.33680	-1.08692	-2.20557	2	2
RetailBanking	2	-0.03799	0.98670	-0.03752	1.94589	1.22456	2	2
RetailBanking	3	-0.29120	-0.45239	0.98855	-0.37208	-1.51534	2	0
RetailBanking	4	0.87499	-0.67812	-0.04839	-1.44881	0.78221	2	1

You can now use this simulated count data to estimate the distribution of the aggregate loss that is incurred in all four operating environments by submitting the following PROC HPCDM step, in which you specify the replication identifier variable RepId in the ID= option of the EXTERNALCOUNTS statement:

```
/* Estimate the distribution of the aggregate loss for both
   lines of business by using the externally simulated counts
   for the multiple operating environments */
proc hpcdm data=lossCounts2 seed=13579 print=all
           severityest=sevestEx2Best plots=density;
   by line;
   distby repid;
   externalcounts count=numloss id=repid;
   severitymodel logn gamma;
run;
```

Note that when you specify the ID= variable in the EXTERNALCOUNTS statement, you must also specify that variable in the DISTBY statement. Within each BY group, for each value of the RepId variable, one point of the aggregate loss sample is simulated by using the process that is described in the section "Simulation with External Counts" on page 71.

The summary statistics and percentiles of the distribution of the aggregate loss, which is the aggregate of the losses across all four operating environments, are shown in Output 3.2.4 for the commercial banking business. The "Input Data Summary" table indicates that there are 10,000 replications in the BY group and that a total of 145,721 loss events are generated across all replications. The "Sample Percentiles" table indicates that you can expect a median aggregate loss of 4,761 units and a worst-case loss, as defined by the 99.5th percentile, of 17,051 units from the commercial banking business when you combine losses that result from all four operating environments.

Output 3.2.4 Aggregate Loss Summary for the Commercial Banking Business in Multiple Operating Environments

The HPCDM Procedure

line=CommercialBanking

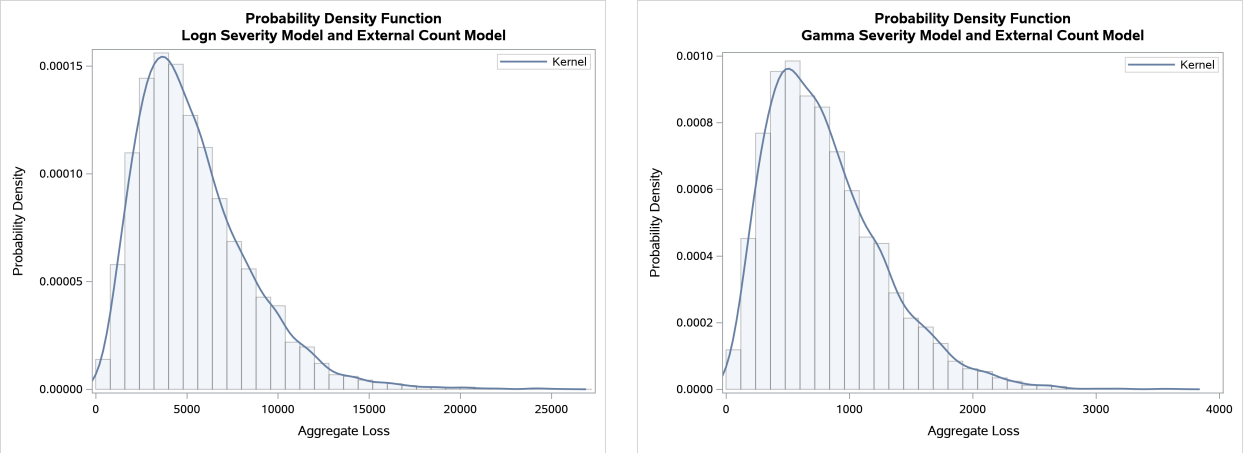
Input Data Summary	
Name	WORK.LOSSCOUNTS2
Observations	40000
Valid Observations	40000
Replications	10000
Total Count	145721

line=CommercialBanking

Sample Percentiles	
Percentile	**Value**
1	762.69922
5	1521.1
25	3127.8
50	4760.8
75	6963.0
95	11180.1
99	15302.7
99.5	17050.8
Percentile Method = 5	

The probability density functions of the aggregate loss for the commercial and retail banking businesses are shown in Output 3.2.5. In addition to the difference in scales of the losses in the two businesses, you can see that the aggregate loss that is incurred in the commercial banking business has a heavier right tail than the aggregate loss that is incurred in the retail banking business.

Output 3.2.5 Density Plots of the Aggregate Losses for Commercial Banking (left) and Retail Banking (right) Businesses



Example 3.3: Scenario Analysis with Rich Regression Effects and BY Groups

This example illustrates scenario analysis when frequency and severity models use regression models that contain classification and interaction effects. It also illustrates how you can analyze scenarios for multiple groups of observations in one PROC HPCDM step without your having to simulate counts externally.

The example in the section "Scenario Analysis" on page 44 encodes the discrete-valued, nominal (nonordinal) variables Gender, CarType, and Education as numerical variables with an implied order. For example, a high school diploma is assigned a smaller number than an advanced degree. This method of forcing an order on otherwise nonordinal (categorical) variables is not natural and might lead to biased estimates. A more accurate approach is to treat such variables as classification variables that enter the statistical analysis or model not through their values but through their levels. For example, when you specify Education as a classification variable, the modeling process creates different parameters for the Education = 'High School' and Education = 'Advanced Degree' levels and estimates a regression coefficient for each. When you specify such variables in the CLASS statement of PROC COUNTREG and PROC SEVERITY, those procedures perform the appropriate levelization for you, which is the process of finding and transforming levels into regression parameters. For more information, see the description of the CLASS statement in Chapter 28, "The SEVERITY Procedure" (*SAS/ETS User's Guide*).

In addition to specifying nominal variables as classification (CLASS) variables, you can include interaction effects in severity and frequency models. For example, you might want to evaluate how the distribution of losses that are incurred by a policyholder with a college degree who drives an SUV differs from that of a policyholder with an advanced degree who drives a sedan. You can do this by including an interaction between CarType and Education in your severity model. Similarly, if you want to evaluate how the number of losses that a policyholder incurs per year varies by the number of annual miles for different types of cars, you can include an interaction between CarType and AnnualMiles in your frequency model. Analyzing such a rich set of regression effects can help you make more accurate predictions about the frequency and severity distributions of losses. PROC HPCDM is designed to use such rich models to simulate a more accurate distribution of the aggregate loss.

As an example of the process, first, let the following programming statements fit the severity and count models that contain a certain set of regression effects:

```
proc severity data=losses(where=(not(missing(lossAmount))))
        covout outstore=work.sevstore print=all plots=none;
    by region;
    loss lossAmount;
    class carType gender education;
    scalemodel carType gender carSafety income education*carType
               income*gender carSafety*income;
    dist logn burr;
run;

proc countreg data=losscounts covout;
   by region;
   class gender carType education;
   model numloss = age income gender carType*annualmiles education / dist=negbin;
   zeromodel numloss ~ age income carType education;
   store cstore;
run;
```

Note the following points about these statements:

- You can find the code that prepares the Work.Losses and Work.LossCounts data sets in the PROC HPCDM sample program *hcdmex03.sas*. The data sets are organized in groups of observations that represent data from two regions, East and West. You can analyze both groups at once by specifying the BY statement with Region as the BY variable.
- Both severity and count models use three CLASS variables. The severity model includes three interaction effects (Education*CarType, Income*Gender, and CarSafety*Income) and four main effects. PROC SEVERITY uses the same set of regression effects in the scale regression model of each of the two distributions that you specify in the DIST statement, which are LOGN and BURR in this example.
- The count model is a mixture of two models: a model to estimate the occurrence of zero loss events and a model to estimate nonzero counts. The zero model is a regression model with four main effects and the default logistic link function. The model for nonzero counts is a negative binomial model with one interaction effect (CarType*AnnualMiles) and four main effects.

The "Parameter Estimates" table of the lognormal severity model in Output 3.3.1 for the Region='East' BY group shows that Income*Gender and CarSafety*Income effects are not statistically significant. The "Parameter Estimates" table in Output 3.3.2 shows that those two effects are not statistically significant for the Burr severity model also.

Output 3.3.1 Parameter Estimates for LOGN Severity Model for Region=East

region=East

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Mu	1	4.98253	0.02861	174.16	<.0001
Sigma	1	0.48894	0.00535	91.41	<.0001
carType SUV	1	0.51772	0.03648	14.19	<.0001
carType Sedan	0	0	.	.	.
gender F	1	1.16690	0.03082	37.86	<.0001
gender M	0	0	.	.	.
carSafety	1	-0.71517	0.04599	-15.55	<.0001
income	1	-0.28528	0.03652	-7.81	<.0001
carType*education SUV Advanced Degree	1	0.44599	0.06245	7.14	<.0001
carType*education SUV College	1	0.67852	0.04416	15.36	<.0001
carType*education SUV High School	0	0	.	.	.
carType*education Sedan Advanced Degree	1	-0.49680	0.02689	-18.47	<.0001
carType*education Sedan College	1	-0.26310	0.01849	-14.23	<.0001
carType*education Sedan High School	0	0	.	.	.
income*gender F	1	0.00988	0.04010	0.25	0.8054
income*gender M	0	0	.	.	.
carSafety*income	1	-0.09390	0.06166	-1.52	0.1278

Output 3.3.2 Parameter Estimates for BURR Severity Model for Region=East

region=East

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Theta	1	145.63709	5.74371	25.36	<.0001
Alpha	1	0.99783	0.06470	15.42	<.0001
Gamma	1	3.58743	0.09362	38.32	<.0001
carType SUV	1	0.51648	0.03701	13.96	<.0001
carType Sedan	0	0	.	.	.
gender F	1	1.16664	0.03083	37.84	<.0001
gender M	0	0	.	.	.
carSafety	1	-0.71636	0.04590	-15.61	<.0001
income	1	-0.29522	0.03639	-8.11	<.0001
carType*education SUV Advanced Degree	1	0.43696	0.06385	6.84	<.0001
carType*education SUV College	1	0.68049	0.04501	15.12	<.0001
carType*education SUV High School	0	0	.	.	.
carType*education Sedan Advanced Degree	1	-0.50160	0.02672	-18.77	<.0001
carType*education Sedan College	1	-0.26483	0.01840	-14.39	<.0001
carType*education Sedan High School	0	0	.	.	.
income*gender F	1	0.01268	0.03986	0.32	0.7504
income*gender M	0	0	.	.	.
carSafety*income	1	-0.07713	0.06162	-1.25	0.2107

The "Parameter Estimates" table of the count model in Output 3.3.3 shows that the income and Inf_income parameters are insignificant. This implies that the income effect is not significant for the main and zero inflation parts of the count model.

The results for the Region='West' BY group are not shown here, but you can execute the sample program *hcdmex03.sas* to verify that the same parameters are statistically insignificant in severity and count models of that BY group as well. However, in general, you might find that some effects are significant for some BY groups but insignificant for other BY groups. In such cases, for more accurate results, it is recommended that you create a separate data set for each set of similar BY groups and invoke the SEVERITY, COUNTREG, and HPCDM procedures on each data set to separately analyze each set of similar BY groups.

Output 3.3.3 Count Model Parameter Estimates for Region=East

region=East

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	1.156626	0.130641	8.85	<.0001
age	1	0.734798	0.112299	6.54	<.0001
income	1	-0.040744	0.081573	-0.50	0.6174
gender F	1	-0.999094	0.053170	-18.79	<.0001
gender M	0	0	.	.	.
annualmiles*carType SUV	1	-1.266452	0.045996	-27.53	<.0001
annualmiles*carType Sedan	1	-0.632281	0.027818	-22.73	<.0001
education Advanced Degree	1	0.418651	0.099414	4.21	<.0001
education College	1	0.709479	0.069596	10.19	<.0001
education High School	0	0	.	.	.
Inf_Intercept	1	-0.501240	0.353072	-1.42	0.1557
Inf_age	1	-0.945657	0.329950	-2.87	0.0042
Inf_income	1	-0.173541	0.233461	-0.74	0.4573
Inf_carType SUV	1	-0.693427	0.369119	-1.88	0.0603
Inf_carType Sedan	0	0	.	.	.
Inf_education Advanced Degree	1	0.668613	0.291821	2.29	0.0220
Inf_education College	1	0.474212	0.232499	2.04	0.0414
Inf_education High School	0	0	.	.	.
_Alpha	1	0.790839	0.103522	7.64	<.0001

The following modified PROC SEVERITY and PROC COUNTREG steps refit the severity and count models, respectively, after removing the insignificant effects:

```
/* Re-fit models after removing insignificant effects. */
proc severity data=losses(where=(not(missing(lossAmount))))
        covout outstore=work.sevstore print=all plots=none;
    by region;
    loss lossAmount;
    class carType gender education;
    scalemodel carType gender carSafety income education*carType;
    dist logn burr;
run;

proc countreg data=losscounts covout;
   by region;
   class gender carType education;
   model numloss = age gender carType*annualmiles education / dist=negbin;
   zeromodel numloss ~ age carType education;
   store cstore;
run;
```

Note that the PROC SEVERITY step uses the OUTSTORE= option to store the parameter estimates in an item store. When your scale regression model contains classification or interaction effects, you must store the parameter estimates in an item store instead of storing them in an OUTEST= data set, because PROC HPCDM cannot obtain the necessary information about classification or interaction effects from an OUTEST= data set.

The "Parameter Estimates" tables in Output 3.3.4 and Output 3.3.5 show that all parameters are now statistically significant, most at the 95% confidence level and a few at the 90% confidence level. If you want every parameter to be significant at the 95% confidence level, then you might want to continue the process by removing the carType effect with a *p*-value of 0.0607 from the ZEROMODEL statement and refitting the count model. However, for the purpose of this example, the preceding models are declared to be satisfactory, and the effect selection process stops here.

You need to follow this process of model inspection and effect selection before you use the severity and count models with the HPCDM procedure. For count models, you can use the automatic effect (variable) selection feature of PROC COUNTREG. For more information, see the description of the SELECT= option in the MODEL statement of Chapter 11, "The COUNTREG Procedure" (*SAS/ETS User's Guide*). For severity models, you need to perform effect selection manually by inspecting the estimates and refitting the model after removing one or a few insignificant effects at a time until you find the final set of significant effects. Although it is not shown in this example, you can also decide which set of effects is better by comparing the fit statistics of two models; the better model might contain certain effects at lower confidence levels than the usual 95% or 90% confidence levels. In fact, the SELECT=INFO option of PROC COUNTREG uses the AIC or BIC of the entire model to select the set of effects instead of using the *p*-values of individual parameters. You might also want to use some domain knowledge to retain certain effects in the model even if their confidence level is not very high.

Output 3.3.4 Final LOGN Severity Model Parameter Estimates for Region=East

region=East

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Mu	1	5.00845	0.02135	234.61	<.0001
Sigma	1	0.48908	0.00535	91.43	<.0001
carType SUV	1	0.51556	0.03642	14.16	<.0001
carType Sedan	0	0	.	.	.
gender F	1	1.17291	0.01726	67.96	<.0001
gender M	0	0	.	.	.
carSafety	1	-0.77273	0.02614	-29.56	<.0001
income	1	-0.32702	0.01962	-16.67	<.0001
carType*education SUV Advanced Degree	1	0.44870	0.06223	7.21	<.0001
carType*education SUV College	1	0.68360	0.04404	15.52	<.0001
carType*education SUV High School	0	0	.	.	.
carType*education Sedan Advanced Degree	1	-0.49572	0.02688	-18.44	<.0001
carType*education Sedan College	1	-0.26234	0.01848	-14.19	<.0001
carType*education Sedan High School	0	0	.	.	.

Output 3.3.5 Final Count Model Parameter Estimates for Region=East

region=East

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	1.136175	0.124786	9.10	<.0001
age	1	0.737805	0.112339	6.57	<.0001
gender F	1	-1.001311	0.052996	-18.89	<.0001
gender M	0	0	.	.	.
annualmiles*carType SUV	1	-1.263178	0.045809	-27.57	<.0001
annualmiles*carType Sedan	1	-0.631419	0.027728	-22.77	<.0001
education Advanced Degree	1	0.400307	0.092060	4.35	<.0001
education College	1	0.703436	0.067935	10.35	<.0001
education High School	0	0	.	.	.
Inf_Intercept	1	-0.585662	0.338796	-1.73	0.0839
Inf_age	1	-0.928294	0.324629	-2.86	0.0042
Inf_carType SUV	1	-0.658089	0.350886	-1.88	0.0607
Inf_carType Sedan	0	0	.	.	.
Inf_education Advanced Degree	1	0.588511	0.269195	2.19	0.0288
Inf_education College	1	0.446600	0.228151	1.96	0.0503
Inf_education High School	0	0	.	.	.
_Alpha	1	0.785018	0.101327	7.75	<.0001

For severity models, you also need to inspect the "All Fit Statistics" table to decide which severity distributions you want to use for aggregate loss modeling. The table in Output 3.3.6 shows that the lognormal distribution is the best according to the majority of fit statistics, so you can choose that. However, in some cases, you might see that the likelihood-based fit statistics (–2 log likelihood, AIC, AICC, BIC) choose one distribution and

the EDF-based statistics (KS, AD, CvM) choose another distribution. In such cases, it is recommended that before making your final decision, you conduct aggregate loss simulation by using both severity distributions and compare the summary statistics and percentiles that each severity distribution produces.

Output 3.3.6 Comparison of Severity Distributions for Region=East

region=East

All Fit Statistics

Distribution	-2 Log Likelihood		AIC		AICC		BIC		KS		AD		CvM	
Logn	45280	*	45300	*	45300	*	45364	*	10.31771	*	613.78765		46.37913	*
Burr	45346		45368		45368		45437		10.90815		519.83495	*	49.71973	

Note: The asterisk (*) marks the best model according to each column's criterion.

After you have satisfactorily estimated the severity and frequency models, it is time to estimate the distribution of the aggregate loss by using the HPCDM procedure. The scenario data set must contain the final set of regressors that are used in both the severity model and the frequency model. Note that even if your models contain interaction effects, your scenario data set needs to contain only the columns for individual variables of the effects. PROC HPCDM internally performs *levelization* of each observation, which is the process of expanding the variable values to match them with the parameters of each effect. A typical scenario for an insurance application might consist of a large number of policyholders, but for illustration purposes, this example uses a small scenario of only a few policyholders per region. Output 3.3.7 shows the contents of the Work.Scenario data set, and the following PROC HPCDM step simulates the aggregate losses for that scenario:

```
proc hpcdm data=scenario nreplicates=10000 seed=123 print=all
           severitystore=work.sevstore countstore=work.cstore
           nperturb=30;
   by region;
   severitymodel logn;
   outsum out=agglossStats mean stddev skewness kurtosis pctlpts=(90 97.5 99.5);
run;
```

Output 3.3.7 Work.Scenario Data Set for BY-Group Processing

Obs	region	gender	carType	education	age	annualmiles	carSafety	income
1	East	F	SUV	High School	1.16	2.1540	0.29288	0.26090
2	East	F	Sedan	High School	0.86	2.3978	0.69844	0.15000
3	East	F	Sedan	Advanced Degree	0.78	1.9926	0.59421	0.58808
4	West	M	Sedan	College	0.82	1.8550	0.66849	0.15000
5	West	M	SUV	College	0.40	3.6240	0.23194	1.25274
6	West	M	Sedan	High School	0.62	3.6162	0.86477	0.42597
7	West	F	Sedan	College	0.32	3.4598	0.66294	0.36132
8	West	M	Sedan	Advanced Degree	0.90	3.2580	0.37172	0.15000

The SEVERITYSTORE= and COUNTSTORE= options specify the item stores that contain the effect information and parameter estimates of the severity and counts models, respectively, for both BY groups. The COVOUT option in the preceding PROC SEVERITY and PROC COUNTREG steps ensures that the respective item stores include the covariance estimates that are needed for the perturbation analysis that the NPERTURB= option requests.

Output 3.3.8 Aggregate Loss Simulation Results for Region=East

The HPCDM Procedure
Severity Model: Logn
Count Model: ZINB

region=East

Sample Percentile Perturbation Analysis

Percentile	Estimate	Standard Error
1	0	0
5	0	0
25	0	0
50	151.62052	20.57120
75	492.04365	33.55686
90	917.18029	51.54978
95	1233.3	63.95801
97.5	1553.5	78.97273
99	1981.2	111.13102
99.5	2308.0	127.42680

Number of Perturbed Samples = 30

Size of Each Sample = 10000

Output 3.3.9 Aggregate Loss Simulation Results for Region=West

region=West

Sample Percentile Perturbation Analysis

Percentile	Estimate	Standard Error
1	0	0
5	0	0
25	134.16405	16.72670
50	417.89498	27.34826
75	863.13053	48.13708
90	1453.7	74.88636
95	1913.2	101.60492
97.5	2368.8	140.43218
99	2979.5	190.75595
99.5	3462.9	242.14530

Number of Perturbed Samples = 30

Size of Each Sample = 10000

Output 3.3.8 and Output 3.3.9 show the summary of the perturbation analysis for the two regions. You can deduce that for the collection of three policyholders in the eastern region of the specified scenario, the 97.5th percentile of their collective aggregate loss is 1553.5 ± 79 units, and for the collection of five policyholders in the western region of the specified scenario, the 99.5th percentile of their collective aggregate loss is 3462.9 ± 242.2.

References

Fisher, R. A. (1973). *Statistical Methods for Research Workers*. 14th ed. New York: Hafner Publishing.

Klugman, S. A., Panjer, H. H., and Willmot, G. E. (1998). *Loss Models: From Data to Decisions*. New York: John Wiley & Sons.

Chapter 4
The HPCOPULA Procedure

Contents

Overview: HPCOPULA Procedure

The HPCOPULA procedure is a high-performance version of the SAS/ETS COPULA procedure, which simulates data from a specified copula. Unlike the COPULA procedure, which can be run only on an individual workstation, the HPCOPULA procedure takes advantage of a computing environment in which the optimization task can be distributed 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 provides a dramatic gain in performance.

You can use the HPCOPULA procedure to read and write data in distributed form and perform analyses either in single-machine mode or in distributed mode. For more information about the execution mode of SAS High-Performance Analytics procedures, see the section "Processing Modes" on page 6.

The HPCOPULA procedure is specifically designed to operate in the high-performance distributed environment. By default, PROC HPCOPULA performs computations in multiple threads.

PROC HPCOPULA Features

The HPCOPULA 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

Getting Started: HPCOPULA Procedure

This example illustrates the use of PROC HPCOPULA. 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. The PERFORMANCE statement requests that the analytic computations use two nodes in the distributed computing environment and two threads in each node. Note that this syntax does not require the DATA= option.

```
/* Copula simulation of uniforms */
proc hpcopula;
   var ret_ibm ret_msft ret_bp ret_ko ret_duk;
   define cop normal (corr = estimates);
   simulate cop / ndraws     = 1000000
                  outuniform = simulated_uniforms;
   PERFORMANCE nodes=2 nthreads=2 details;
run;
```

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

Syntax: HPCOPULA Procedure

The following statements are available in the HPCOPULA procedure:

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

Functional Summary

Table 4.1 summarizes the statements and options that the HPCOPULA procedure uses.

Table 4.1 Functional Summary

Description	Statement	Option
Data Set Options		
Specifies the input data set that contains the correlation matrix for elliptical copulas	DEFINE	CORR=
Declaring the Role of Variables		
Specifies the names of the variables to use in copula fitting or in simulation	VAR	
Copula Simulation Options		
Specifies the random sample size	SIMULATE	NDRAWS=
Specifies the random number generator seed	SIMULATE	SEED=
Output Control Options		
Specifies the output data set to contain the random samples from the simulation with uniform marginal distribution	SIMULATE	OUTUNIFORM=

PROC HPCOPULA Statement

PROC HPCOPULA ;

The PROC HPCOPULA statement invokes the HPCOPULA procedure.

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 be 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: HPCOPULA Procedure" on page 118:

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 4.1.1 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. You can use this option when MARGINALS=UNIFORM or MARGINALS=EMPIRICAL. If MARGINALS=EMPIRICAL, then this option enables you to obtain the samples that are simulated from the joint distribution specified by the copula, where all marginal distributions are uniform. 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.

PERFORMANCE Statement

PERFORMANCE < *performance-options* > **;**

The PERFORMANCE statement specifies *performance-options* to control the multithreaded and distributed computing environment and requests detailed performance results of the HPCOPULA procedure. You can also use the PERFORMANCE statement to control whether the HPCOPULA procedure executes in SMP or MPP mode. You can specify the following *performance-options*:

DETAILS
requests a table that shows a timing breakdown of the PROC HPCOPULA steps.

NODES=*n*
specifies the number of nodes in the distributed computing environment, provided that the data are not processed alongside the database.

NTHREADS=*n*
specifies the number of threads for analytic computations and overrides the SAS system option THREADS | NOTHREADS. If you do not specify the NTHREADS= option, PROC HPCOPULA creates one thread per CPU for the analytic computations.

For more information about the PERFORMANCE statement, see the section "PERFORMANCE Statement" on page 31.

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 HPCOPULA 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: HPCOPULA 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 $C : [0, 1]^m \rightarrow [0, 1]$ 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 $\mathrm{Ran}F_1 \times \cdots \times \mathrm{Ran}F_m$, where $\mathrm{Ran}F_j = F_j([-\infty, \infty])$ 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

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

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):

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

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

  $$\lambda_u(X_1, X_2) \equiv \lim_{q->1^-} P(X_2 > F_2^{-1}(q) | X_1 > F_1^{-1}(q))$$

  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 Σ be the correlation matrix, where $m(m-1)/2$ parameters satisfy the positive semidefiniteness constraint. The normal copula can be written as

$$C_\Sigma(u_1, u_2, \ldots, u_m) = \mathbf{\Phi}_\Sigma\Big(\Phi^{-1}(u_1), \ldots, \Phi^{-1}(u_m)\Big)$$

where Φ is the distribution function of a standard normal random variable and $\mathbf{\Phi}_\Sigma$ is the m-variate standard normal distribution with mean vector 0 and covariance matrix Σ. That is, the distribution $\mathbf{\Phi}_\Sigma$ is $N_m(0, \Sigma)$.

Simulation

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

1. Generate a multivariate normal vector $\boldsymbol{Z} \sim N(0, \Sigma)$, where Σ is an m-dimensional correlation matrix.
2. Transform the vector $\boldsymbol{Z}$ into $\boldsymbol{U} = (\Phi(Z_1), \ldots, \Phi(Z_m))^T$, where Φ 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 $\tilde{\boldsymbol{Z}} \sim N(0, I)$, then $L\tilde{\boldsymbol{Z}} \sim N(0, \Sigma)$.

Student's *t* Copula

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

The Student's t copula can be written as

$$C_\Theta(u_1, u_2, \ldots, u_m) = \boldsymbol{t}_{\nu,\Sigma}\left(t_\nu^{-1}(u_1), t_\nu^{-1}(u_2), \ldots, t_\nu^{-1}(u_m)\right)$$

where $\boldsymbol{t}_{\nu,\Sigma}$ is the multivariate Student's t distribution that has a correlation matrix Σ with ν degrees of freedom.

Simulation

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

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

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

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 ϕ^{-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 HPCOPULA 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 $\theta \in (-\infty, \infty)\backslash\{0\}$ 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 ϕ. 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 $\boldsymbol{U} = (\tilde{F}(-\log(X_1)/V), \ldots, \tilde{F}(-\log(X_m)/V))^T$.

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 θ^{-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 $\mathrm{St}(\theta^{-1}, 1, \gamma, 0)$, 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.

Examples: HPCOPULA Procedure

Example 4.1: 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 4.1.1 show that the correlation parameter is set at 0.8:

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

Output 4.1.1 Copula Correlation Matrix

Obs	Y1	Y2
1	1.0	0.8
2	0.8	1.0

The following statements use PROC HPCOPULA to simulate the data:

```
/* simulate the data from bivariate normal copula */
proc hpcopula;
   var Y1-Y2;
   define cop normal (corr=inparm);
   simulate cop /
            ndraws     = 1000000
            seed       = 1234
            outuniform = normal_unifdata;
   PERFORMANCE nodes=4 nthreads=4 details
               host="&GRIDHOST" install="&GRIDINSTALLLOC";
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 PERFORMANCE statement requests that the analytic computations be performed on four nodes in the distributed computing environment and four threads on each node. Output 4.1.2 shows the run time of this particular simulation experiment.

Output 4.1.2 Run-Time Performance

Performance Information	
Host Node	<< your grid host >>
Install Location	<< your grid install location >>
Execution Mode	Distributed
Number of Compute Nodes	4
Number of Threads per Node	4

Procedure Task Timing		
Task	**Seconds**	**Percent**
Communication to client	0.00	0.02%
Simulation of Model	0.03	20.63%
Writing of output data	0.13	79.35%

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.

```
/* default time has exponential marginal distribution with parameter 0.5 */
data default;
   set normal_unifdata;
   array arr{2} Y1-Y2;
   array time{2} time1-time2;
   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 4.1.3 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 4.1.3 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	2.00042	1.99724	1.38664	1.78961E-6	28.39277
time2	1000000	2.00190	2.00064	1.38787	2.24931E-6	30.50949

Pearson Correlation Coefficients, N = 1000000
Prob > |r| under H0: Rho=0

	time1	time2
time1	1.00000	0.76950 <.0001
time2	0.76950 <.0001	1.00000

Kendall Tau b Correlation Coefficients, N = 1000000
Prob > |tau| under H0: Tau=0

	time1	time2
time1	1.00000	0.58998 <.0001
time2	0.58998 <.0001	1.00000

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 4.1.4.

Output 4.1.4 Probabilities of Default

The FREQ Procedure

survive	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	852314	85.23	852314	85.23
1	147686	14.77	1000000	100.00

survive1	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	776565	77.66	776565	77.66
1	223435	22.34	1000000	100.00

survive2	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	776382	77.64	776382	77.64
1	223618	22.36	1000000	100.00

Output 4.1.4 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

Devroye, L. (1986). *Non-uniform Random Variate Generation*. New York: Springer-Verlag. http://luc.devroye.org/rnbookindex.html.

Marshall, A. W., and Olkin, I. (1988). "Families of Multivariate Distributions." *Journal of the American Statistical Association* 83:834–841.

Nolan, J. P. (2010). *Stable Distributions: Models for Heavy Tailed Data*. Boston: Birkhäuser.

Sklar, A. (1959). "Fonctions de répartition à *n* dimensions et leurs marges." *Publications de l'Institut de Statistique de L'Université de Paris* 8:229–231.

Chapter 5
The HPCOUNTREG Procedure

Contents

Overview: HPCOUNTREG Procedure

The HPCOUNTREG procedure is a high-performance version of the COUNTREG procedure in SAS/ETS software. Like the COUNTREG procedure, the HPCOUNTREG procedure fits regression models in which the dependent variable takes on nonnegative integer or count values. Unlike the COUNTREG procedure, which can be run only on an individual workstation, the HPCOUNTREG procedure takes advantage of a computing environment that enables it to distribute the optimization task among one or more nodes. In addition, each node can use one or more threads to carry out the optimization on its subset of the data. When several nodes are employed, with each node using several threads to carry out its part of the work, the result is a highly parallel computation that provides a dramatic gain in performance.

The HPCOUNTREG procedure enables you to read and write data in distributed form and perform analyses in distributed mode and single-machine mode. For information about how to affect the execution mode of SAS high-performance analytical procedures, see the section "Processing Modes" on page 6.

The HPCOUNTREG procedure is specifically designed to operate in the high-performance distributed environment. By default, PROC HPCOUNTREG performs computations in multiple threads.

PROC HPCOUNTREG Features

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

The HPCOUNTREG 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
- 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 HPCOUNTREG procedure:

- can perform analysis on a massively parallel high-performance appliance
- reads input data in parallel and writes output data in parallel when the data source is the appliance database
- is highly multithreaded during all phases of analytic execution
- has model-building syntax that uses CLASS and effect-based MODEL statements familiar from SAS/ETS 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

Getting Started: HPCOUNTREG Procedure

Except for its ability to operate in the high-performance distributed environment, the HPCOUNTREG 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:

```
proc hpcountreg 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 HPCOUNTREG. 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 5.1.

Figure 5.1 Article Count Data

Obs	art	fem	mar	kid5	phd	ment
1	3	0	1	2	1.38000	8.0000
2	0	0	0	0	4.29000	7.0000
3	4	0	0	0	3.85000	47.0000
4	1	0	1	1	3.59000	19.0000
5	1	0	1	0	1.81000	0.0000
6	1	0	1	1	3.59000	6.0000
7	0	0	1	1	2.12000	10.0000
8	0	0	1	0	4.29000	2.0000
9	3	0	1	2	2.58000	2.0000
10	3	0	1	1	1.80000	4.0000

The following SAS statements estimate the Poisson regression model. The model is executed in the distributed computing environment with two threads and four nodes.

```
/*-- Poisson Regression --*/
proc hpcountreg data=long97data;
   model art = fem mar kid5 phd ment / dist=poisson method=quanew;
   performance nthreads=2 nodes=4 details;
run;
```

The "Model Fit Summary" table that is shown in Figure 5.2 lists several details about the model. By default, the HPCOUNTREG 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 5.2 Estimation Summary Table for a Poisson Regression

The HPCOUNTREG Procedure

Model Fit Summary	
Dependent Variable	art
Number of Observations	915
Data Set	WORK.LONG97DATA
Model	Poisson
Log Likelihood	-1651
Maximum Absolute Gradient	0.0002080
Number of Iterations	13
Optimization Method	Quasi-Newton
AIC	3314
SBC	3343

Figure 5.3 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 5.3 Parameter Estimates of Poisson Regression

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	0.3046	0.1030	2.96	0.0031
fem	1	-0.2246	0.05461	-4.11	<.0001
mar	1	0.1552	0.06137	2.53	0.0114
kid5	1	-0.1849	0.04013	-4.61	<.0001
phd	1	0.01282	0.02640	0.49	0.6271
ment	1	0.02554	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.

```
/*-- Negative Binomial Regression --*/
proc hpcountreg data=long97data;
   model art = fem mar kid5 phd ment / dist=negbin(p=2) method=quanew;
   performance nthreads=2 nodes=4 details;
run;
```

Figure 5.4 shows the fit summary and Figure 5.5 shows the parameter estimates.

Figure 5.4 Estimation Summary Table for a Negative Binomial Regression

The HPCOUNTREG Procedure

Model Fit Summary

Dependent Variable	art
Number of Observations	915
Data Set	WORK.LONG97DATA
Model	NegBin
Log Likelihood	-1561
Maximum Absolute Gradient	0.0000666
Number of Iterations	16
Optimization Method	Quasi-Newton
AIC	3136
SBC	3170

Figure 5.5 Parameter Estimates of Negative Binomial Regression

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	0.2561	0.1386	1.85	0.0645
fem	1	-0.2164	0.07267	-2.98	0.0029
mar	1	0.1505	0.08211	1.83	0.0668
kid5	1	-0.1764	0.05306	-3.32	0.0009
phd	1	0.01527	0.03604	0.42	0.6718
ment	1	0.02908	0.003470	8.38	<.0001
_Alpha	1	0.4416	0.05297	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.

Syntax: HPCOUNTREG Procedure

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

PROC HPCOUNTREG <*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*> ;
  PERFORMANCE *performance-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. The ZEROMODEL statement, if used, must appear after the MODEL statement. If a FREQ or WEIGHT statement is specified more than once, the variable specified in the first instance is used.

Functional Summary

Table 5.1 summarizes the statements and options used with the HPCOUNTREG procedure.

Table 5.1 Functional Summary

Description	Statement	Option
Data Set Options		
Specifies the input data set	HPCOUNTREG	DATA=
Specifies the identification variable for panel data analysis	HPCOUNTREG	GROUPID=
Writes parameter estimates to an output data set	HPCOUNTREG	OUTEST=
Writes estimates to an output data set	OUTPUT	OUT=
Specifies BY-group processing	BY	
Specifies an optional frequency variable	FREQ	
Specifies an optional weight variable	WEIGHT	
Printing Control Options		
Prints the correlation matrix of the estimates	HPCOUNTREG	CORRB
Prints the covariance matrix of the estimates	HPCOUNTREG	COVB
Suppresses the normal printed output	HPCOUNTREG	NOPRINT
Requests all printing options	HPCOUNTREG	PRINTALL
Options to Control the Optimization Process		
Specifies maximum number of iterations allowed	HPCOUNTREG	MAXITER=
Selects the iterative minimization method to use	HPCOUNTREG	METHOD=
Specifies maximum number of iterations allowed	HPCOUNTREG	MAXITER=
Specifies maximum number of function calls	HPCOUNTREG	MAXFUNC=
Specifies the upper limit of CPU time in seconds	HPCOUNTREG	MAXTIME=
Specifies absolute function convergence criterion	HPCOUNTREG	ABSCONV=
Specifies absolute function convergence criterion	HPCOUNTREG	ABSFCONV=
Specifies absolute gradient convergence criterion	HPCOUNTREG	ABSGCONV=
Specifies relative function convergence criterion	HPCOUNTREG	FCONV=
Specifies relative gradient convergence criterion	HPCOUNTREG	GCONV=
Specifies absolute parameter convergence criterion	HPCOUNTREG	ABSXCONV=
Specifies matrix singularity criterion	HPCOUNTREG	SINGULAR=
Sets boundary restrictions on parameters	BOUNDS	
Sets initial values for parameters	INIT	
Sets linear restrictions on parameters	RESTRICT	
Model Estimation Options		
Specifies the dispersion variables	DISPMODEL	
Specifies the type of model	HPCOUNTREG	DIST=
Specifies the type of covariance matrix	HPCOUNTREG	COVEST=
Specifies the type of error components model for panel data	MODEL	ERRORCOMP=

Table 5.1 *continued*

Description	Statement	Option
Suppresses the intercept parameter	MODEL	NOINT
Specifies the offset variable	MODEL	OFFSET=
Specifies the parameterization for the Conway-Maxwell-Poisson (CMP) model	MODEL	PARAMETER=
Specifies the zero-inflated offset variable	ZEROMODEL	OFFSET=
Specifies the zero-inflated link function	ZEROMODEL	LINK=
Output Control Options		
Includes covariances in the OUTEST= data set	HPCOUNTREG	COVOUT
Includes correlations in the OUTEST= data set	HPCOUNTREG	CORROUT
Outputs SAS variables to the output data set	OUTPUT	COPYVAR=
Outputs the estimates of dispersion for the CMP model	OUTPUT	DISPERSION
Outputs the estimates of GDelta $= \mathbf{g}_i'\boldsymbol{\delta}$ for the CMP model	OUTPUT	GDELTA=
Outputs the estimates of λ for the CMP model	OUTPUT	LAMBDA=
Outputs the estimates of ν for the CMP model	OUTPUT	NU=
Outputs the estimates of μ for the CMP model	OUTPUT	MU=
Outputs the estimates of mode for the CMP model	OUTPUT	MODE=
Outputs the probability that the response variable will take the current value	OUTPUT	PROB=
Outputs probabilities for particular response values	OUTPUT	PROBCOUNT( )
Outputs expected value of response variable	OUTPUT	PRED=
Outputs the estimates of variance for the CMP model	OUTPUT	VARIANCE=
Outputs estimates of XBeta $= \mathbf{x}_i'\boldsymbol{\beta}$	OUTPUT	XBETA=
Outputs estimates of ZGamma $= \mathbf{z}_i'\boldsymbol{\gamma}$	OUTPUT	ZGAMMA=
Outputs probability of a zero value as a result of the zero-generating process	OUTPUT	PROBZERO=
Performance Options		
Requests a table that shows a timing breakdown	PERFORMANCE	DETAILS
Specifies the number of threads to use	PERFORMANCE	NTHREADS=
Specifies the number of nodes to use on the SAS appliance	PERFORMANCE	NODES=

PROC HPCOUNTREG Statement

PROC HPCOUNTREG <*options*> **;**

The following *options* can be used in the PROC HPCOUNTREG statement.

Input Data Set Options

DATA=*SAS-data-set*
specifies the input SAS data set. If the DATA= option is not specified, PROC HPCOUNTREG 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.

Output Data Set Options

OUTEST=*SAS-data-set*
writes the parameter estimates to the specified output data set.

CORROUT
writes the correlation matrix for the parameter estimates to the OUTEST= data set. This option is valid only if the OUTEST= option is specified.

COVOUT
writes the covariance matrix for the parameter estimates to the OUTEST= data set. This option is valid only if the OUTEST= option is specified.

Printing Options

You can specify the following options in either the PROC HPCOUNTREG 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.

Estimation Control Options

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

ABSCONV=*r*

ABSTOL=*r*

specifies an absolute function value convergence criterion by which minimization stops when $f(\theta^{(k)}) \leq r$. 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 by which minimization stops when the function value has a small change in successive iterations:

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

The default is 0.

ABSGCONV=*r*

ABSGTOL=*r*

specifies an absolute gradient convergence criterion. Optimization stops when the maximum absolute gradient element is small:

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

The default is 1E–5.

ABSXCONV=*r*

ABSXTOL=*r*

specifies an absolute parameter convergence criterion. Optimization stops when the Euclidean distance between successive parameter vectors is small:

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

The default is 0.

FCONV=*r*

FTOL=*r*

specifies a relative function convergence criterion. Optimization stops when a relative change of the function value in successive iterations is small:

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

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

GCONV=*r*

GTOL=*r*

specifies a relative gradient convergence criterion. For all techniques except CONGRA, optimization stops when the normalized predicted function reduction is small:

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

For the CONGRA technique (where a reliable Hessian estimate H is not available), the following criterion is used:

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

The default is 1E–8.

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.

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.

SINGULAR=*r*
specifies the general singularity criterion that is applied by the HPCOUNTREG procedure in sweeps and inversions. The default is 1E–8.

BOUNDS Statement

BOUNDS *bound1* [*, bound2 . . .*] **;**

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. 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 159.

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 144.

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 `x1` through `x10` to be between 0 and 1, and the parameter for `x1` in the zero-inflation model to be less than 1.

```
bounds z < 0,
       0 < x1-x10 < 1,
       Inf_x1 < 1;
```

BY Statement

BY *variables* **;**

A BY statement can be used with PROC HPCOUNTREG 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 HPCOUNTREG 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

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

DISPMODEL *dependent-variable* ∼ < *dispersion-related-regressors* > **;**

The DISPMODEL statement specifies the *dispersion-related-regressors* that are used to model dispersion. This statement is ignored unless you specify DIST=CMPOISSON 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 (∼) has a parameter to be estimated in the regression. For example, let $\mathbf{g}_i'$ be the ith 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 $\mathbf{g}_i'\boldsymbol{\delta}$, where $\boldsymbol{\delta}$ is the $(q+1)\times 1$ vector of parameters to be estimated, the dispersion model intercept is δ_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 δ_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 hpcountreg 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 HPCOUNTREG 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. 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 159.

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 HPCOUNTREG rounds any positive noninteger count value to the nearest integer. PROC HPCOUNTREG 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 (/):

DIST=*value*
specifies a type of model to be analyzed. You can specify the following *values*:

POISSON | P	specifies the Poisson regression model.
CMPOISSON | C | CMP	specifies a Conway-Maxwell-Poisson regression model.
NEGBIN(P=1)	specifies the negative binomial regression model that uses a linear variance function.
NEGBIN(P=2) | NEGBIN	specifies the negative binomial regression model that uses a quadratic variance function.
ZIPOISSON | ZIP	specifies zero-inflated Poisson regression.
ZICMPOISSON | ZICMP	specifies a zero-inflated Conway-Maxwell-Poisson regression. The ZERO-MODEL statement must be specified when this model type is specified.
ZINEGBIN | ZINB	specifies zero-inflated negative binomial regression.

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

ERRORCOMP=FIXED | RANDOM
specifies a type of conditional panel model to be analyzed. You can specify the following model types:

FIXED	specifies a fixed-effect error component regression model.
RANDOM	specifies a random-effect error component regression model.

NOINT
suppresses the intercept parameter.

OFFSET=*offset-variable*
specifies a variable in the input data set to be used as an offset variable. The *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."

PARAMETER=MU | LAMBDA
specifies the parameterization for the Conway-Maxwell-Poisson model. The following parameterizations are supported:

LAMBDA	estimates the original Conway-Maxwell-Poisson model (Shmueli et al. 2005).
MU	reparameterizes λ as documented by Guikema and Coffelt (2008), where $\mu = \lambda^{1/\nu}$ and the integral part of μ 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 HPCOUNTREG 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

OUTPUT < *output-options* > **;**

The OUTPUT statement creates a new SAS data set that includes variables created by the *output-options*. These variables include the estimates of $\mathbf{x}_i'\boldsymbol{\beta}$, 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 $\mathbf{z}_i'\boldsymbol{\gamma}$ 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 $\mathbf{g}_i'\boldsymbol{\delta}$, λ, ν, μ, 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*

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

XBETA=*name*
names the variable to contain estimates of $\mathbf{x}_i'\boldsymbol{\beta}$.

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, 4, 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 $\mathbf{z}_i'\boldsymbol{\gamma}$.

PROBZERO=*name*
names the variable to contain the value of φ_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 $\mathbf{g}_i'\boldsymbol{\delta}$ for the Conway-Maxwell-Poisson distribution.

LAMBDA=*name*
assigns a name to the variable that contains the estimate of λ for the Conway-Maxwell-Poisson distribution.

NU=*name*
assigns a name to the variable that contains the estimate of ν for the Conway-Maxwell-Poisson distribution.

MU=*name*
assigns a name to the variable that contains the estimate of μ for the Conway-Maxwell-Poisson distribution.

MODE=*name*
assigns a name to the variable that contains the integral part of μ (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.

PERFORMANCE Statement

PERFORMANCE < *performance-options* > **;**

The PERFORMANCE statement specifies options to control the multithreaded and distributed computing environment and requests detailed results about the performance characteristics of the HPCOUNTREG procedure. You can also use the PERFORMANCE statement to control whether the HPCOUNTREG procedure executes in single-machine or distributed mode. The most commonly used *performance-options* in the PERFORMANCE statement are as follows:

DETAILS
requests a table that shows a timing breakdown of the procedure steps.

NODES=*n*
specifies the number of nodes in the distributed computing environment, provided that the data are not processed alongside the database.

NTHREADS=*n*
specifies the number of threads for analytic computations and overrides the SAS system option THREADS | NOTHREADS. If you do not specify the NTHREADS= option, PROC HPCOUNTREG creates one thread per CPU for the analytic computations.

For more information about the PERFORMANCE statement for high-performance analytical procedures, see the section "PERFORMANCE Statement" on page 31.

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 (−). Parameter names are as shown in the Effect column of the "Parameter Estimates" table. 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. 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 159.

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 $\mu + \mu^2$:

```
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 HPCOUNTREG uses the unquoted *label*. If you specify neither an unquoted *label* nor a quoted *string*, PROC HPCOUNTREG 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. 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 159. 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:

```
proc hpcountreg;
   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 HPCOUNTREG. 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 (φ_i) of a zero count. Each of these q variables has a parameter to be estimated in the regression. For example, let $\mathbf{z}_i'$ be the ith 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 φ_i is a function of $\mathbf{z}_i'\boldsymbol{\gamma}$, where $\boldsymbol{\gamma}$ is the $(q+1) \times 1$ vector of parameters to be estimated. (The zero-inflated intercept is γ_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 γ_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 (φ_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: HPCOUNTREG 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 HPCOUNTREG and not used in the model fit. PROC HPCOUNTREG 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 HPCOUNTREG 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 $\mathbf{x}_i'\boldsymbol{\beta}$, 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 $\mathbf{z}_i'\boldsymbol{\gamma}$, 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 $\mathbf{x}_i$, is independently Poisson distributed with

$$P(Y_i = y_i|\mathbf{x}_i) = \frac{e^{-\mu_i}\mu_i^{y_i}}{y_i!}, \quad y_i = 0, 1, 2, \ldots$$

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

$$\mu_i = \exp(\mathbf{x}_i'\boldsymbol{\beta})$$

where $\boldsymbol{\beta}$ is a $(k+1)\times 1$ parameter vector. (The intercept is β_0; the coefficients for the k regressors are $\beta_1, \ldots, \beta_k$.) Taking the exponential of $\mathbf{x}_i'\boldsymbol{\beta}$ ensures that the mean parameter μ_i is nonnegative. It can be shown that the conditional mean is given by

$$E(y_i|\mathbf{x}_i) = \mu_i = \exp(\mathbf{x}_i'\boldsymbol{\beta})$$

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

$$V(y_i|\mathbf{x}_i) = E(y_i|\mathbf{x}_i) = \mu_i$$

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

$$\mathcal{L} = \sum_{i=1}^{N}(-\mu_i + y_i \ln \mu_i - \ln y_i!) = \sum_{i=1}^{N}(-e^{\mathbf{x}_i'\boldsymbol{\beta}} + y_i\mathbf{x}_i'\boldsymbol{\beta} - \ln y_i!)$$

For more information about the Poisson regression model, see the section "Poisson Regression" (Chapter 11, *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 $\mathbf{x}_i$, is independently Poisson-distributed as

$$P(Y_i = y_i|\mathbf{x}_i) = \frac{e^{-\lambda_i}\lambda_i^{y_i}}{y_i!}, \quad y_i = 0, 1, 2, \ldots$$

The Conway-Maxwell-Poisson distribution is defined as

$$P(Y_i = y_i|\mathbf{x}_i, \mathbf{z}_i) = \frac{1}{Z(\lambda_i, \nu_i)}\frac{\lambda_i^{y_i}}{(y_i!)^{\nu_i}}, \quad y_i = 0, 1, 2, \ldots$$

where the normalization factor is

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

and

$$\lambda_i = \exp(\mathbf{x}_i'\boldsymbol{\beta})$$

$$\nu_i = -\exp(\mathbf{g}_i'\delta)$$

The $\boldsymbol{\beta}$ vector is a $(k + 1) \times 1$ parameter vector. (The intercept is β_0, and the coefficients for the k regressors are $\beta_1, \ldots, \beta_k$.) The δ vector is an $(m + 1) \times 1$ parameter vector. (The intercept is represented by δ_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:

$$E(y_i|\mathbf{x}_i) = V(y_i|\mathbf{x}_i) = \lambda_i = \exp(\mathbf{x}_i'\boldsymbol{\beta})$$

The CMP distribution overcomes this restriction by defining an additional parameter, ν, 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, ν, 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

$$P(Y = y_i; \lambda_i, \nu_i = 0) = (1 - \lambda_i)\lambda_i^{y_i}$$

The geometric distribution represents a case of severe overdispersion.

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

The mean and the 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 λ and ν. 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:

$$E(Y) = \frac{1}{Z(\lambda, \nu)} \sum_{j=0}^{\infty} \frac{j\lambda^j}{(j!)^\nu}$$

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

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

$$\begin{aligned} L(y_1, y_2, \ldots, y_n|\lambda, \nu) &= \frac{\prod_{i=1}^{n} \lambda^{y_i}}{(\prod_{i=1}^{n} y_i!)^{\nu}} Z(\lambda, \nu)^{-n} \\ &= \lambda^{\sum_{i=1}^{n} y_i} \exp\left(-\nu \sum_{i=1}^{n} \ln(y_i!)\right) Z(\lambda, \nu)^{-n} \\ &= \lambda^{S_1} \exp(-\nu S_2) Z(\lambda, \nu)^{-n} \end{aligned}$$

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

$$\mathcal{L}_\beta = \left(\sum_{k=1}^{N} y_k - n \frac{\lambda Z(\lambda, \nu)_\lambda}{Z(\lambda, \nu)}\right) \mathbf{x}$$

$$\mathcal{L}_\delta = \left(\sum_{k=1}^{N} \ln(y_k!) - n \frac{Z(\lambda, \nu)_\nu}{Z(\lambda, \nu)}\right) \nu \mathbf{z}$$

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

$$S(\mu, \nu) = \sum_{j=0}^{\infty} \left(\frac{\mu^j}{j!}\right)^{\nu}$$

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

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

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

They can be approximated as

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

$$V[Y] \approx \frac{\mu}{\nu}$$

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

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

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

In terms of the new parameter μ, 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

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

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

By default, the HPCOUNTREG 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 ν 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 μ representing the mode, which is a reasonable approximation for the mean. The dispersion can be written as

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

When $\nu < 1$, 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 $\nu = 1$ and the mean and variance are equal, the dispersion is equal to 1 (Poisson model). When $\nu > 1$, 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 HPCOUNTREG procedure are parameterized in terms of dispersion, where

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

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

$$E(y_i|\mathbf{x}_i, \tau_i) = \mu_i \tau_i = e^{\mathbf{x}_i'\boldsymbol{\beta}+\epsilon_i}$$

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

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

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

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

An analytical solution to this integral exists when τ_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 τ_i follows a gamma(θ, θ) 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 θ is a positive parameter. Then, the density of y_i given $\mathbf{x}_i$ is derived as

$$\begin{aligned} f(y_i|\mathbf{x}_i) &= \int_0^\infty f(y_i|\mathbf{x}_i, \tau_i) g(\tau_i) d\tau_i \\ &= \frac{\theta^\theta \mu_i^{y_i}}{y_i! \Gamma(\theta)} \int_0^\infty e^{-(\mu_i+\theta)\tau_i} \tau_i^{\theta+y_i-1} d\tau_i \\ &= \frac{\theta^\theta \mu_i^{y_i} \Gamma(y_i+\theta)}{y_i! \Gamma(\theta)(\theta+\mu_i)^{\theta+y_i}} \\ &= \frac{\Gamma(y_i+\theta)}{y_i! \Gamma(\theta)} \left(\frac{\theta}{\theta+\mu_i}\right)^\theta \left(\frac{\mu_i}{\theta+\mu_i}\right)^{y_i} \end{aligned}$$

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

$$f(y_i|\mathbf{x}_i) = \frac{\Gamma(y_i+\alpha^{-1})}{y_i! \Gamma(\alpha^{-1})} \left(\frac{\alpha^{-1}}{\alpha^{-1}+\mu_i}\right)^{\alpha^{-1}} \left(\frac{\mu_i}{\alpha^{-1}+\mu_i}\right)^{y_i}, \quad y_i = 0, 1, 2, \ldots$$

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

$$E(y_i|\mathbf{x}_i) = \mu_i = e^{\mathbf{x}_i'\boldsymbol{\beta}}$$

and the conditional variance

$$V(y_i|\mathbf{x}_i) = \mu_i[1 + \frac{1}{\theta}\mu_i] = \mu_i[1 + \alpha\mu_i] > E(y_i|\mathbf{x}_i)$$

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 $V(y_i|\mathbf{x}_i) = \mu_i + \alpha\mu_i^2$, 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 $\alpha = \frac{1}{\theta_i} = 0$. A Wald test of this hypothesis is provided (it is the reported t statistic for the estimated α in the negative binomial model).

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

$$\begin{aligned} \mathcal{L} = \sum_{i=1}^N \Bigg\{ & \sum_{j=0}^{y_i-1} \ln(j+\alpha^{-1}) - \ln(y_i!) \\ & -(y_i+\alpha^{-1}) \ln(1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta})) + y_i \ln(\alpha) + y_i \mathbf{x}_i'\boldsymbol{\beta} \Bigg\} \end{aligned}$$

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 μ_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 $f(y_i|\mathbf{x}_i)$, except that α^{-1} is replaced everywhere by $\alpha^{-1}\mu^{2-p}$. The negative binomial model NEGBIN1, which sets $p = 1$, has the variance function $V(y_i|\mathbf{x}_i) = \mu_i + \alpha\mu_i$, 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

$$\begin{aligned}\mathcal{L} &= \sum_{i=1}^{N}\left\{\sum_{j=0}^{y_i-1}\ln\left(j+\alpha^{-1}\exp(\mathbf{x}_i'\boldsymbol{\beta})\right)\right.\\ &\quad\left.-\ln(y_i!)-\left(y_i+\alpha^{-1}\exp(\mathbf{x}_i'\boldsymbol{\beta})\right)\ln(1+\alpha)+y_i\ln(\alpha)\right\}\end{aligned}$$

For more information about the negative binomial regression model, see the section "Negative Binomial Regression" (Chapter 11, *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 φ_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,

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

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

$$\begin{aligned} P(y_i = 0|\mathbf{x}_i) &= \varphi_i + (1-\varphi_i)g(0) \\ P(y_i|\mathbf{x}_i) &= (1-\varphi_i)g(y_i), \quad y_i > 0 \end{aligned}$$

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

If the probability φ_i depends on the characteristics of observation i, then φ_i is written as a function of $\mathbf{z}_i'\boldsymbol{\gamma}$, where $\mathbf{z}_i'$ is the $1 \times (q+1)$ vector of zero-inflated covariates and $\boldsymbol{\gamma}$ is the $(q+1) \times 1$ vector of zero-inflated coefficients to be estimated. (The zero-inflated intercept is γ_0; the coefficients for the q zero-inflated covariates are $\gamma_1, \ldots, \gamma_q$.) The function F that relates the product $\mathbf{z}_i'\boldsymbol{\gamma}$ (which is a scalar) to the probability φ_i is called the zero-inflated link function,

$$\varphi_i = F_i = F(\mathbf{z}_i'\boldsymbol{\gamma})$$

In the HPCOUNTREG 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,

$$F(\mathbf{z}_i'\boldsymbol{\gamma}) = \Lambda(\mathbf{z}_i'\boldsymbol{\gamma}) = \frac{\exp(\mathbf{z}_i'\boldsymbol{\gamma})}{1+\exp(\mathbf{z}_i'\boldsymbol{\gamma})}$$

or the standard normal cumulative distribution function (also called the probit function),

$$F(\mathbf{z}_i'\boldsymbol{\gamma}) = \Phi(\mathbf{z}_i'\boldsymbol{\gamma}) = \int_0^{\mathbf{z}_i'\boldsymbol{\gamma}} \frac{1}{\sqrt{2\pi}} \exp(-u^2/2)du$$

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 $\mu_i = e^{\mathbf{x}_i'\boldsymbol{\beta}}$. Thus the ZIP model is defined as

$$\begin{aligned} P(y_i = 0|\mathbf{x}_i, \mathbf{z}_i) &= F_i + (1-F_i)\exp(-\mu_i) \\ P(y_i|\mathbf{x}_i, \mathbf{z}_i) &= (1-F_i)\frac{\exp(-\mu_i)\mu_i^{y_i}}{y_i!}, \quad y_i > 0 \end{aligned}$$

The conditional expectation and conditional variance of y_i are given by

$$E(y_i|\mathbf{x}_i, \mathbf{z}_i) = \mu_i(1-F_i)$$

$$V(y_i|\mathbf{x}_i, \mathbf{z}_i) = E(y_i|\mathbf{x}_i, \mathbf{z}_i)(1+\mu_i F_i)$$

Note that the ZIP model (in addition to the ZINB model) exhibits overdispersion because $V(y_i|\mathbf{x}_i, \mathbf{z}_i) > E(y_i|\mathbf{x}_i, \mathbf{z}_i)$.

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

$$\mathcal{L} = \sum_{i=1}^{N} \ln\left[P(y_i|\mathbf{x}_i, \mathbf{z}_i)\right]$$

After a specific link function (either logistic or standard normal) for the probability φ_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 φ_i is expressed by a logistic link function, namely

$$\varphi_i = \frac{\exp(\mathbf{z}_i'\boldsymbol{\gamma})}{1+\exp(\mathbf{z}_i'\boldsymbol{\gamma})}$$

The log-likelihood function is

$$\begin{aligned}\mathcal{L} &= \sum_{\{i:y_i=0\}} \ln\left[\exp(\mathbf{z}_i'\boldsymbol{\gamma}) + \exp(-\exp(\mathbf{x}_i'\boldsymbol{\beta}))\right] \\ &\quad + \sum_{\{i:y_i>0\}} \left[y_i\mathbf{x}_i'\boldsymbol{\beta} - \exp(\mathbf{x}_i'\boldsymbol{\beta}) - \sum_{k=2}^{y_i}\ln(k)\right] \\ &\quad - \sum_{i=1}^{N}\ln\left[1+\exp(\mathbf{z}_i'\boldsymbol{\gamma})\right]\end{aligned}$$

ZIP Model with Standard Normal Link Function

Next, consider the ZIP model in which the probability φ_i is expressed by a standard normal link function: $\varphi_i = \Phi(\mathbf{z}_i'\boldsymbol{\gamma})$. The log-likelihood function is

$$\begin{aligned}\mathcal{L} &= \sum_{\{i:y_i=0\}} \ln\left\{\Phi(\mathbf{z}_i'\boldsymbol{\gamma}) + \left[1-\Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right]\exp(-\exp(\mathbf{x}_i'\boldsymbol{\beta}))\right\} \\ &\quad + \sum_{\{i:y_i>0\}} \left\{ \ln\left[\left(1-\Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right)\right] - \exp(\mathbf{x}_i'\boldsymbol{\beta}) + y_i\mathbf{x}_i'\boldsymbol{\beta} - \sum_{k=2}^{y_i}\ln(k)\right\}\end{aligned}$$

For more information about the zero-inflated Poisson regression model, see the section "Zero-Inflated Poisson Regression" (Chapter 11, *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

$$P(Y_i = y_i|\mathbf{x}_i,\mathbf{z}_i) = \frac{1}{Z(\lambda_i,\nu_i)}\frac{\lambda_i^{y_i}}{(y_i!)^{\nu_i}}, \quad y_i = 0,1,2,\ldots$$

where the normalization factor is

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

and

$$\lambda_i = \exp(\mathbf{x}_i'\boldsymbol{\beta})$$

$$\nu_i = -\exp(\mathbf{g}_i'\delta)$$

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

$$\begin{aligned} P(y_i|\mathbf{x}_i,\mathbf{z}_i) &= F_i + (1-F_i)\frac{1}{Z(\lambda_i,\nu_i)}, \quad y_i = 0 \\ P(y_i|\mathbf{x}_i,\mathbf{z}_i) &= (1-F_i)\frac{1}{Z(\lambda_i,\nu_i)}\frac{\lambda_i^{y_i}}{(y_i!)^{\nu_i}}, \quad y_i > 0 \end{aligned}$$

The conditional expectation and conditional variance of y_i are given respectively by

$$E(y_i|\mathbf{x}_i,\mathbf{z}_i) = (1-F_i)\frac{1}{Z(\lambda,\nu)}\sum_{j=0}^{\infty}\frac{j\lambda^j}{(j!)^\nu}$$

$$V(y_i|\mathbf{x}_i,\mathbf{z}_i) = (1-F_i)\frac{1}{Z(\lambda,\nu)}\sum_{j=0}^{\infty}\frac{j^2\lambda^j}{(j!)^\nu} - E(y_i|\mathbf{x}_i,\mathbf{z}_i)^2$$

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

$$\mathcal{L} = \sum_{i=1}^{N} w_i \ln\left[P(y_i|\mathbf{x}_i,\mathbf{z}_i)\right]$$

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

For this model, the probability φ_i is expressed by using a logistic link function as

$$\varphi_i = \Lambda(\mathbf{z}_i'\boldsymbol{\gamma}) = \frac{\exp(\mathbf{z}_i'\boldsymbol{\gamma})}{1+\exp(\mathbf{z}_i'\boldsymbol{\gamma})}$$

The log-likelihood function is

$$\begin{aligned} \mathcal{L} &= \sum_{\{i:y_i=0\}} w_i \ln\left\{\Lambda(\mathbf{z}_i'\boldsymbol{\gamma}) + \left[1-\Lambda(\mathbf{z}_i'\boldsymbol{\gamma})\right]\frac{1}{Z(\lambda_i,\nu_i)}\right\} \\ &+ \sum_{\{i:y_i>0\}} w_i \left\{\ln\left[\left(1-\Lambda(\mathbf{z}_i'\boldsymbol{\gamma})\right)\right] - ln(Z(\lambda,\nu)) + (y_i\ln(\lambda) - \nu\ln(y_i!)\right\} \end{aligned}$$

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

For this model, the probability φ_i is specified by using the standard normal distribution function (probit function): $\varphi_i = \Phi(\mathbf{z}_i'\boldsymbol{\gamma})$.

The log-likelihood function is written as

$$\begin{aligned} \mathcal{L} &= \sum_{\{i:y_i=0\}} w_i \ln\left\{\Phi(\mathbf{z}_i'\boldsymbol{\gamma}) + \left[1-\Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right]\frac{1}{Z(\lambda_i,\nu_i)}\right\} \\ &+ \sum_{\{i:y_i>0\}} w_i \left\{\ln\left[\left(1-\Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right)\right] - ln(Z(\lambda,\nu)) + (y_i\ln(\lambda) - \nu\ln(y_i!)\right\} \end{aligned}$$

Zero-Inflated Negative Binomial Regression

The zero-inflated negative binomial (ZINB) model in PROC HPCOUNTREG is based on the negative binomial model that has a quadratic variance function (when DIST=NEGBIN in the MODEL or PROC HPCOUNTREG 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

$$\begin{aligned} P(y_i = 0|\mathbf{x}_i, \mathbf{z}_i) &= F_i + (1-F_i)\,(1+\alpha\mu_i)^{-\alpha^{-1}} \\ P(y_i|\mathbf{x}_i, \mathbf{z}_i) &= (1-F_i)\,\frac{\Gamma(y_i+\alpha^{-1})}{y_i!\Gamma(\alpha^{-1})}\left(\frac{\alpha^{-1}}{\alpha^{-1}+\mu_i}\right)^{\alpha^{-1}} \\ &\times \left(\frac{\mu_i}{\alpha^{-1}+\mu_i}\right)^{y_i}, \quad y_i > 0 \end{aligned}$$

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

$$E(y_i|\mathbf{x}_i, \mathbf{z}_i) = \mu_i(1-F_i)$$

$$V(y_i|\mathbf{x}_i, \mathbf{z}_i) = E(y_i|\mathbf{x}_i, \mathbf{z}_i)\,[1 + \mu_i(F_i + \alpha)]$$

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 φ_i is given by the logistic function, namely

$$\varphi_i = \frac{\exp(\mathbf{z}_i'\boldsymbol{\gamma})}{1+\exp(\mathbf{z}_i'\boldsymbol{\gamma})}$$

The log-likelihood function is

$$\begin{aligned} \mathcal{L} &= \sum_{\{i:y_i=0\}} \ln\left[\exp(\mathbf{z}_i'\boldsymbol{\gamma}) + (1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))^{-\alpha^{-1}}\right] \\ &+ \sum_{\{i:y_i>0\}}\sum_{j=0}^{y_i-1} \ln(j+\alpha^{-1}) \\ &+ \sum_{\{i:y_i>0\}} \left\{-\ln(y_i!) - (y_i+\alpha^{-1})\ln(1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta})) + y_i\ln(\alpha) + y_i\mathbf{x}_i'\boldsymbol{\beta}\right\} \\ &- \sum_{i=1}^{N} \ln\left[1+\exp(\mathbf{z}_i'\boldsymbol{\gamma})\right] \end{aligned}$$

ZINB Model with Standard Normal Link Function

For this model, the probability φ_i is expressed by the standard normal distribution function (probit function): $\varphi_i = \Phi(\mathbf{z}_i'\boldsymbol{\gamma})$. The log-likelihood function is

$$\begin{aligned}
\mathcal{L} &= \sum_{\{i:y_i=0\}} \ln\left\{\Phi(\mathbf{z}_i'\boldsymbol{\gamma}) + \left[1-\Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right](1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta}))^{-\alpha^{-1}}\right\} \\
&+ \sum_{\{i:y_i>0\}} \ln\left[1-\Phi(\mathbf{z}_i'\boldsymbol{\gamma})\right] \\
&+ \sum_{\{i:y_i>0\}} \sum_{j=0}^{y_i-1} \left\{\ln(j+\alpha^{-1})\right\} \\
&- \sum_{\{i:y_i>0\}} \ln(y_i!) \\
&- \sum_{\{i:y_i>0\}} (y_i+\alpha^{-1})\ln(1+\alpha\exp(\mathbf{x}_i'\boldsymbol{\beta})) \\
&+ \sum_{\{i:y_i>0\}} y_i\ln(\alpha) \\
&+ \sum_{\{i:y_i>0\}} y_i\mathbf{x}_i'\boldsymbol{\beta}
\end{aligned}$$

For more information about the zero-inflated negative binomial regression model, see the section "Zero-Inflated Negative Binomial Regression" (Chapter 11, *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 OUTEST= data set.

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

```
model y = x1 x2 x5;
```

where x1 through x5 are continuous variables. If you want to restrict the parameter associated with the regressor x5 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 Inf_ to the name of the regressor. Suppose your MODEL and ZEROMODEL statements are as follows:

```
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 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 Inf_, it refers to the parameter associated with that regressor in the ZEROMODEL statement. The parameter associated with the intercept in the ZEROMODEL is named Inf_Intercept.

In a similar way, you can form the name of a parameter associated with a regressor in the DISPMODEL statement by prefixing Dsp_ to the name of the regressor. The parameter associated with the intercept in the DISPMODEL is named Dsp_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:

```
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:

```
model y = x1 x2 x3*x4;
```

You can form the name of the parameter associated with the interaction regressor x3*x4 by replacing the multiplication sign with an underscore. Thus, x3_x4 refers to the parameter that is associated with the interaction regressor x3*x4.

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:

```
class C;
model y = x1 x2 C*x3;
```

The interaction between the continuous variable x3 and the classification variable C introduces three additional parameters, which are named x3_C_0, x3_C_1, and x3_C_2. Note how, although the order of the terms in the interaction is C followed by x3, the name of the parameter associated with the interaction is formed by placing the name of the continuous variable x3 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:

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

The nested interaction between the continuous variable x3 and the classification variable C introduces three additional parameters, which are named x3_C__0, x3_C__1, and x3_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:

```
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:

```
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:

```
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:

```
RESTRICT _Alpha > 0.2;
```

Computational Resources

The time and memory that PROC HPCOUNTREG 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 HPCOUNTREG 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 HPCOUNTREG 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 $8 * (p + p * (p + 1)/2)$ 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 HPCOUNTREG executes. You can alter the convergence criteria by using the nonlinear optimization options available in the PROC HPCOUNTREG statement. For a list of all the nonlinear optimization options, see "Optimization Control Options" on page 136.

Covariance Matrix Types

The COVEST= option in the PROC HPCOUNTREG 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 HPCOUNTREG produces the following displayed output.

Model Fit Summary

The "Model Fit Summary" table contains the following information:

- dependent (count) variable name
- number of observations used
- number of missing values in data set, if any
- data set name
- type of model that was fit
- parameterization for the Conway-Maxwell-Poisson model
- offset variable name, if any

- zero-inflated link function, if any
- zero-inflated offset variable name, if any
- log-likelihood value at solution
- maximum absolute gradient at solution
- number of iterations
- AIC value at solution (smaller value indicates better fit)
- SBC value at solution (smaller value indicates better fit)

A line in 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 HPCOUNTREG or MODEL statement, the HPCOUNTREG 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 HPCOUNTREG or MODEL statement, the HPCOUNTREG 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 $\mathbf{x}_i'\boldsymbol{\beta}$, 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 $\mathbf{z}_i'\boldsymbol{\gamma}$, 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 $\mathbf{g}_i'\boldsymbol{\delta}$, λ, ν, μ, 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.

OUTEST= Data Set

The OUTEST= data set is made up of at least two rows: the first row (with _TYPE_='PARM') contains each of the parameter estimates in the model, and the second row (with _TYPE_='STD') contains the standard errors for the parameter estimates in the model.

If you use the COVOUT option in the PROC HPCOUNTREG statement, the OUTEST= data set also contains the covariance matrix for the parameter estimates. The covariance matrix appears in the observations with _TYPE_='COV', and the _NAME_ variable labels the rows with the parameter names.

ODS Table Names

PROC HPCOUNTREG 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 5.2.

Table 5.2 ODS Tables Produced in PROC HPCOUNTREG

ODS Table Name	Description	Option
ODS Tables Created by the MODEL Statement		
FitSummary	Summary of nonlinear estimation	Default
ConvergenceStatus	Convergence status	Default
ParameterEstimates	Parameter estimates	Default
CovB	Covariance of parameter estimates	COVB
CorrB	Correlation of parameter estimates	CORRB

Examples: The HPCOUNTREG Procedure

Example 5.1: High-Performance Zero-Inflated Poisson Model

This example shows the use of the HPCOUNTREG procedure with an emphasis on large data set processing and the performance improvements that are achieved by executing in the high-performance distributed environment.

The following DATA step generates one million replicates from the zero-inflated Poisson (ZIP) model. The model contains seven variables and three variables that correspond to the zero-inflated process.

```
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=2;
   z_intercept=-1;
   theta=0.5;

   do i=1 to 1000000;
      sum_xb=0;
      sum_gz=0;
      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 x1-x7 z1-z3;
run;
```

The following statements estimate a zero-inflated Poisson model:

```
option set=GRIDHOST="&GRIDHOST";
option set=GRIDINSTALLLOC="&GRIDINSTALLLOC";

proc hpcountreg data=simulate dist=zip;
   performance nthreads=2 nodes=1 details
           host="&GRIDHOST" install="&GRIDINSTALLLOC";
   model y_p=x1-x7;
   zeromodel y_p ~ z1-z3;
run;
```

The model is executed in the distributed computing environment on two threads and only one node. These settings are used to obtain a hypothetical environment that might resemble running the HPCOUNTREG procedure on a desktop workstation with a dual-core CPU. To run these statements successfully, you need to set the macro variables GRIDHOST and GRIDINSTALLLOC to resolve to appropriate values, or you can replace the references to the macro variables in the example with the appropriate values. Output 5.1.1 shows the "Performance Information" table for this hypothetical scenario.

Output 5.1.1 Performance Information with One Node and One Thread

Performance Information	
Host Node	<< your grid host >>
Install Location	<< your grid install location >>
Execution Mode	Distributed
Number of Compute Nodes	1
Number of Threads per Node	2

Output 5.1.2 shows the results for the zero-inflated Poisson model. The "Model Fit Summary" table shows detailed information about the model and indicates that all one million 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. The optimization of the model that contains one million observations took 52.22 seconds.

Output 5.1.2 Zero-Inflated Poisson Model Execution on One Node and Two Threads

Model Fit Summary	
Dependent Variable	y_p
Number of Observations	1000000
Data Set	WORK.SIMULATE
Model	ZIP
ZI Link Function	Logistic
Log Likelihood	-2215238
Maximum Absolute Gradient	2.07042E-8
Number of Iterations	7
Optimization Method	Newton-Raphson
AIC	4430500
SBC	4430642

Convergence criterion (FCONV=2.220446E-16) satisfied.

Output 5.1.2 *continued*

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	2.0005	0.000492	4069.80	<.0001
x1	1	0.2995	0.000352	850.17	<.0001
x2	1	0.3998	0.000353	1132.23	<.0001
x3	1	0.2008	0.000352	570.27	<.0001
x4	1	0.3994	0.000353	1132.85	<.0001
x5	1	-0.2995	0.000353	-848.95	<.0001
x6	1	-0.5000	0.000353	-1414.9	<.0001
x7	1	-0.3002	0.000352	-852.14	<.0001
Inf_Intercept	1	-0.9993	0.002521	-396.45	<.0001
Inf_z1	1	-0.6024	0.002585	-233.02	<.0001
Inf_z2	1	0.2976	0.002454	121.25	<.0001
Inf_z3	1	0.1974	0.002430	81.20	<.0001

Procedure Task Timing

Task	Seconds	Percent
Reading and Levelizing Data	0.45	0.84%
Communication to Client	0.08	0.16%
Optimization	52.22	98.82%
Post-Optimization	0.09	0.17%

In the following statements, the PERFORMANCE statement is modified to use a grid with 10 nodes, with each node capable of spawning eight threads:

```
proc hpcountreg data=simulate dist=zip;
   performance nthreads=8 nodes=10 details
               host="&GRIDHOST" install="&GRIDINSTALLLOC";
   model y_p=x1-x7;
   zeromodel y_p ~ z1-z3;
run;
```

Because the two models being estimated are identical, it is reasonable to expect that Output 5.1.2 and Output 5.1.3 would show the same results. However, you can see a significant difference in performance between the two models. The second model, which was run on a grid that used 10 nodes with eight threads each, took only 3.15 seconds instead of 52.22 seconds to optimize.

In certain circumstances, you might observe slight numerical differences in the results, depending on the number of nodes and threads involved. This happens because the order in which partial results are accumulated can make a difference in the final result, owing to the limits of numerical precision and the propagation of error in numerical computations.

Output 5.1.3 Zero-Inflated Poisson Model Execution on 10 Nodes with Eight Threads Each

The HPCOUNTREG Procedure

Model Fit Summary	
Dependent Variable	y_p
Number of Observations	1000000
Data Set	WORK.SIMULATE
Model	ZIP
ZI Link Function	Logistic
Log Likelihood	-2215238
Maximum Absolute Gradient	2.10036E-8
Number of Iterations	7
Optimization Method	Newton-Raphson
AIC	4430500
SBC	4430642

Convergence criterion (FCONV=2.220446E-16) satisfied.

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	2.0005	0.000492	4069.80	<.0001
x1	1	0.2995	0.000352	850.17	<.0001
x2	1	0.3998	0.000353	1132.23	<.0001
x3	1	0.2008	0.000352	570.27	<.0001
x4	1	0.3994	0.000353	1132.85	<.0001
x5	1	-0.2995	0.000353	-848.95	<.0001
x6	1	-0.5000	0.000353	-1414.9	<.0001
x7	1	-0.3002	0.000352	-852.14	<.0001
Inf_Intercept	1	-0.9993	0.002521	-396.45	<.0001
Inf_z1	1	-0.6024	0.002585	-233.02	<.0001
Inf_z2	1	0.2976	0.002454	121.25	<.0001
Inf_z3	1	0.1974	0.002430	81.20	<.0001

Procedure Task Timing

Task	Seconds	Percent
Reading and Levelizing Data	0.03	0.78%
Communication to Client	0.01	0.35%
Optimization	3.15	98.27%
Post-Optimization	0.02	0.60%

As this example suggests, increasing the number of nodes and the number of threads per node improves performance significantly. When you use the parallelism afforded by a high-performance distributed environment, you can see an even more dramatic reduction in the time required for the optimization as the number of observations in the data set increases. When the data set is extremely large, the computations might not even be possible in some cases, given the typical memory resources and computational constraints of a desktop computer. Under such circumstances the high-performance distributed environment becomes a necessity.

References

Cameron, A. C., and Trivedi, P. K. (1986). "Econometric Models Based on Count Data: Comparisons and Applications of Some Estimators and Tests." *Journal of Applied Econometrics* 1:29–53.

Cameron, A. C., and Trivedi, P. K. (1998). *Regression Analysis of Count Data*. Cambridge: Cambridge University Press.

Conway, R. W., and Maxwell, W. L. (1962). "A Queuing Model with State Dependent Service Rates." *Journal of Industrial Engineering* 12:132–136.

Guikema, S. D., and Coffelt, J. P. (2008). "A Flexible Count Data Regression Model for Risk Analysis." *Risk Analysis* 28:213–223.

Lambert, D. (1992). "Zero-Inflated Poisson Regression with an Application to Defects in Manufacturing." *Technometrics* 34:1–14.

LaMotte, L. R. (1994). "A Note on the Role of Independence in *t* Statistics Constructed from Linear Statistics in Regression Models." *American Statistician* 48:238–240.

Long, J. S. (1997). *Regression Models for Categorical and Limited Dependent Variables*. Thousand Oaks, CA: Sage Publications.

Lord, D., Guikema, S. D., and Geedipally, S. R. (2008). "Application of the Conway-Maxwell-Poisson Generalized Linear Model for Analyzing Motor Vehicle Crashes." *Accident Analysis and Prevention* 40:1123–1134.

Shmueli, G., Minka, T. P., Kadane, J. B., Borle, S., and Boatwright, P. (2005). "A Useful Distribution for Fitting Discrete Data: Revival of the Conway-Maxwell-Poisson Distribution." *Journal of the Royal Statistical Society, Series C* 54:127–142.

Chapter 6
The HPPANEL Procedure

Contents

Overview: HPPANEL Procedure

The HPPANEL procedure is a high-performance version of the PANEL procedure in SAS/ETS software. Both procedures analyze a class of linear econometric models that commonly arise when time series and cross-sectional data are combined. This type of data on time series cross-sectional bases is often referred to as panel data. Typical examples of panel data include observations over time about households, countries, firms, trade, and so on. For example, in the case of survey data about household income, the panel is created by repeatedly surveying the same households in different time periods (years).

Unlike the PANEL procedure (which can be run only on an individual workstation), the HPPANEL procedure takes advantage of a computing environment that enables it to distribute the optimization task among one or more nodes. Running on one node is called single-machine, and running on more than one node is called distributed mode. In addition, each node (whether in single-machine mode or in distributed mode) can use one or more threads to carry out 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 provides a dramatic gain in performance.

NOTE: Distributed mode requires SAS High-Performance Econometrics.

You can use the HPPANEL procedure to read and write data in distributed form and perform analyses in distributed mode or in single-machine mode. For more information about how to affect the execution mode of SAS high-performance analytical procedures, see the section "Processing Modes" on page 6.

The HPPANEL procedure is specifically designed to operate in the high-performance distributed mode. By default, PROC HPPANEL performs computations in multiple threads.

The panel data models can be grouped into several categories that depend on the structure of the error term. The HPPANEL procedure uses the following error structures and the corresponding methods to analyze data:

- one-way and two-way models
- fixed-effects and random-effects 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.

Apart from the possible one-way or two-way nature of the effect, the other dimension of difference between the possible specifications is the nature of the cross-sectional or time-series effect. The models are referred to as fixed-effects models if the effects are nonrandom and as random-effects models otherwise.

If the effects are fixed, the models are essentially regression models that have dummy variables that correspond to the specified effects. For fixed-effects models, ordinary least squares (OLS) estimation is the best linear unbiased estimator. Random-effects models use a two-stage approach: In the first stage, variance components are calculated by using methods described by 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 ordinary least squares (OLS) regression is performed.

Getting Started: HPPANEL Procedure

The following statements use the cost function data from Greene (1990) to estimate the variance components model. 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. For more information, see Greene (1990).

```
data greene;
   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 1955   6.54535   1.35041  2 1960   6.69827   1.71109
2 1965   7.40245   2.09519  2 1970   7.82644   2.39480
3 1955   8.07153   2.94628  3 1960   8.47679   3.25967

   ... more lines ...
```

You decide to fit the following model to the data,

$$C_{it} = \text{Intercept} + \boldsymbol{\beta} P_{it} + v_i + e_t + \epsilon_{it} \quad \text{for } i = 1, \ldots, N \text{ and } t = 1, \ldots, T$$

where C_{it} and P_{it} represent the cost and production; and v_i, e_t, and ϵ_{it} are the cross-sectional, time series, and error variance components, respectively.

If you assume that the time and cross-sectional effects are random, four possible estimators are left for the variance components. The following statements choose the Fuller-Battese method to fit this model:

```
proc hppanel data=greene;
   model cost = production / rantwo vcomp = fb;
   id firm year;
   performance nodes=0 nthreads=2;
run;
```

The output of the HPPANEL procedure is shown in Output 6.1.

Figure 6.1 Two-Way Random Effects Results

The HPPANEL Procedure

Model Information	
Data Source	GREENE
Response Variable	cost
Model	RANTWO
Variance Component	FULLER

Fit Statistics	
Sum of Squared Error	0.348082
Degrees of Freedom	22
Mean Squared Error	0.015822
Root Mean Squared Error	0.125785
R-Square	0.813624

Variance Component Estimates	
Variance Component for Cross Sections	0.0469
Variance Component for Time Series	0.00906
Variance Component for Error	0.00875

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	-2.99992	0.64778	-4.63	<.0001
production	1	0.74660	0.07618	9.80	<.0001

Printed first is the model description, which reports the method used for estimation and the method used for estimating error components. Printed next is the fit statistics table, and then the variance components estimates. Finally, the table of regression parameter estimates shows the estimates, standard errors, and *t* tests.

Syntax: HPPANEL Procedure

The following statements are available in the HPPANEL procedure:

```
PROC HPPANEL options ;
  ID cross-section-id time-series-id ;
  MODEL response = regressors < /options > ;
  RESTRICT equation1< ,equation2. . . > ;
  TEST equation < ,equation2. . . >< / options > ;
  OUTPUT OUT=SAS-data-set < output-options > ;
  PERFORMANCE < performance-options > ;
```

The ID and MODEL statements are required.

The following sections provide a functional summary of statements and options, describe the PROC HPPANEL statement, and then describe the other statements in alphabetical order.

Functional Summary

Table 6.1 summarizes the statements and options that you can use in the HPPANEL procedure.

Table 6.1 Functional Summary

Description	Statement	Option
Data Set Options		
Includes correlations in the OUTEST= data set	PROC HPPANEL	CORROUT
Includes covariances in the OUTEST= data set	PROC HPPANEL	COVOUT
Specifies the input data set	PROC HPPANEL	DATA=
Specifies the name of an output SAS data set	OUTPUT	OUT=
Writes parameter estimates to an output data set	PROC HPPANEL	OUTEST=
Variable Role Options		
Specifies the cross-sectional and time ID variables	ID	
Printing Control Options		
Prints correlations of the estimates	PROC HPPANEL	CORRB
Prints covariances of the estimates	PROC HPPANEL	COVB
Suppresses printed output	PROC HPPANEL	NOPRINT
Prints fixed effects	MODEL	PRINTFIXED
Performs tests of linear hypotheses	TEST	
Model Estimation Options		
Estimates the between-groups model	MODEL	BTWNG
Estimates the between-time-periods model	MODEL	BTWNT
Estimates the one-way fixed-effects model	MODEL	FIXONE

Table 6.1 *continued*

Description	Statement	Option
Estimates the one-way fixed-effects model with respect to time	MODEL	FIXONETIME
Estimates the two-way fixed-effects model	MODEL	FIXTWO
Suppresses the intercept term	MODEL	NOINT
Estimates the pooled regression model	MODEL	POOLED
Estimates the one-way random-effects model	MODEL	RANONE
Estimates 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	
Specifies which tests to perform	TEST	WALD, LM, LR

PROC HPPANEL Statement

PROC HPPANEL *options* **;**

The HPPANEL statement invokes the HPPANEL procedure.

You can specify the following *options*:

DATA=*SAS-data-set*
names the input data set. Only one observation is allowed for each cross section and time period. If you omit the DATA= option, PROC HPPANEL uses the most recently created SAS data set.

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

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

NOPRINT
suppresses the normal printed output.

OUTEST=*SAS-data-set*
names an output data set to contain the parameter estimates. When the OUTEST= option is not specified, the OUTEST= data set is not created. For more information about the structure of the OUTEST= data set, see the section "OUTEST= Data Set" on page 190.

OUTCOV

COVOUT
writes the standard errors and covariance matrix of the parameter estimates to the OUTEST= data set. For more information, see the section "OUTEST= Data Set" on page 190.

OUTCORR

CORROUT

writes the correlation matrix of the parameter estimates to the OUTEST= data set. For more information, see the section "OUTEST= Data Set" on page 190.

In addition, you can specify any of the following MODEL statement options in the PROC HPPANEL statement: FIXONE, FIXONETIME, FIXTWO, RANONE, RANTWO, NOINT, PRINTFIXED, and VCOMP=. Specifying these options in the PROC HPPANEL statement is equivalent to specifying them in the MODEL statement. For a complete description of each of these options, see the section "MODEL Statement" on page 177.

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 the time period for each observation. The ID statement is required. Unlike the PANEL procedure, the HPPANEL procedure does not require the data set to be sorted.

MODEL Statement

MODEL *response* = *regressors* < / *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 is regressed on the independent variables (*regressors*). You can specify only one MODEL statement and only one *response*.

You specify the error structure and estimation technique by including one of the following *options* after a slash (/):

BTWNG

estimates the between-groups model.

BTWNT

estimates the between-time-periods model.

FIXONE

estimates a one-way fixed-effects model, which corresponds to cross-sectional effects.

FIXONETIME

estimates a one-way fixed-effects model, which corresponds to time effects.

FIXTWO

estimates a two-way fixed-effects model.

POOLED

estimates the pooled regression model.

RANONE
estimates a one-way random-effects model.

RANTWO
estimates a two-way random-effects model.

By default, a FIXONE estimation is performed.

You can also specify the following *options* after the slash:

NOINT
suppresses the intercept parameter from the model.

PRINTFIXED
prints the fixed effects.

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

FB	requests the Fuller-Battese estimator.
WK	requests the Wansbeek-Kapteyn estimator.
WH	requests the Wallace-Hussain estimator.
NERLOVE	requests the Nerlove estimator.

By default, VCOMP=WK for both balanced and unbalanced data.

OUTPUT Statement

OUTPUT OUT=*SAS-data-set* < *output-options* > **;**

The OUTPUT statement creates a new SAS data set to contain variables that are specified by the COPYVAR option, the cross-sectional ID (_CSID_), and the time period (_TSID_). This data set also contains the predicted value and the residual if they are specified by *output-options*. When the response values are missing for the observation, all output estimates except the residual are still computed as long as none of the explanatory variables are missing. You can specify only one OUTPUT statement.

You must specify the OUT= option:

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

You can specify one or more of the following *output-options*:

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

PREDICTED
outputs estimates of predicted dependent variables.

RESIDUAL
outputs estimates of residuals.

PERFORMANCE Statement

PERFORMANCE < *performance-options* > **;**

The PERFORMANCE statement specifies *performance-options* to control the multithreaded and distributed computing environment and requests detailed performance results of the HPPANEL procedure. You can also use the PERFORMANCE statement to control whether the HPPANEL procedure executes in single-machine or distributed mode. You can specify the following *performance-options*:

DETAILS
requests a table that shows a timing breakdown of the procedure steps.

NODES=*n*
specifies the number of nodes in the distributed computing environment, provided that the data are not processed alongside the database.

NTHREADS=*n*
specifies the number of threads for analytic computations and overrides the SAS system option THREADS | NOTHREADS. If you do not specify the NTHREADS= option, PROC HPPANEL creates one thread per CPU for the analytic computations.

The PERFORMANCE statement is documented further in the section "PERFORMANCE Statement" on page 31.

RESTRICT Statement

RESTRICT *equation1* < *,equation2. . .* > **;**

The RESTRICT statement specifies linear equality restrictions on the parameters in the MODEL statement. There can be as many unique restrictions as the number of parameters in the MODEL statement. Multiple RESTRICT statements are understood as joint restrictions on the model's parameters.

Currently, PROC HPPANEL only supports linear equality restrictions. Restriction expressions can be composed only of algebraic operations that involve the addition symbol (+), subtraction symbol (–), and multiplication symbol (*).

The following statements illustrate the use of the RESTRICT statement:

```
proc hppanel;
   id csid tsid;
   model y = x1 x2 x3;
   restrict x1 = 0, x2 * .5 + 2 * x3= 0;
   restrict  x2 = 0, intercept = 0;
run;
```

A RESTRICT statement cannot include a division sign in its formulation. As in the preceding example, you can obtain restrictions on the intercept by using the keyword INTERCEPT.

TEST Statement

TEST *equation1* <*,equation2...*> <*/ options*> **;**

The TEST statement performs Wald, Lagrange multiplier, and likelihood ratio tests of linear hypotheses about the regression parameters in the MODEL statement. Each *equation* specifies a linear hypothesis to be tested. Currently, only linear equality restrictions and tests are permitted in PROC HPPANEL. Test expressions can be composed only of algebraic operations that involve the addition symbol (+), subtraction symbol (–), and multiplication symbol (*). 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. In the equality restrictions, you can use the keyword INTERCEPT to refer to the coefficient of the intercept.

You can specify the following *options* after the 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.

By default, the Wald test is performed.

The following statements illustrate the use of the TEST statement:

```
proc hppanel;
   id csid tsid;
   model y = x1 x2 x3;
   test x1 = 0, x2 * .5 + 2 * x3 = 0;
   test intercept = 0, x3 = 0;
run;
```

The first test investigates the joint hypothesis that

$$\beta_1 = 0$$

and

$$0.5\beta_2 + 2\beta_3 = 0$$

Details: HPPANEL Procedure

Specifying the Input Data

The HPPANEL procedure is similar to other regression procedures in SAS. Suppose you want to regress the variable Y on regressors X1 and X2. Cross sections are identified by the variable State, and time periods are identified by the variable Date. Unlike the PANEL procedure, the HPPANEL procedure does not require the data set to be sorted. To invoke the HPPANEL procedure, you must specify the cross section and time series variables in an ID statement. The following statements show the correct syntax:

```
proc hppanel data=a;
   id state date;
   model y = x1 x2;
   performance nodes=2 nthreads=4;
run;
```

Specifying the Regression Model

The MODEL statement in PROC HPPANEL is specified like the MODEL statement in other SAS regression procedures: the dependent variable is listed first, followed by an equal sign, followed by the list of regressor variables, as shown in the following statements:

```
proc hppanel data=a;
   id state date;
   model y = x1 x2;
   performance nodes=2 nthreads=4;
run;
```

Specifying the Number of Nodes and Number of Threads

The PERFORMANCE statement in PROC HPPANEL is specified like the PERFORMANCE statement in other SAS high-performance procedures. The following statements execute the model in the distributed computing environment with two threads and four nodes:

```
proc hppanel data=a;
   id state date;
   model y = x1 x2;
   performance nodes=2 nthreads=4;
run;
```

The major advantage of using PROC HPPANEL is that you can incorporate a model for the structure of the random errors. It is important to consider what type of error structure model is appropriate for your data and to specify the corresponding option in the MODEL statement.

The error structure options supported by the HPPANEL procedure are FIXONE, FIXONETIME, FIXTWO, RANONE, and RANTWO. For more information about these methods and the error structures they assume,

see the following sections. The following statements fit a Fuller-Battese one-way random-effects model:

```
proc hppanel data=a;
   id state date;
   model y = x1 x2 / ranone vcomp=fb;
   performance nodes=0 nthreads=1;
run;
```

To aid in model specification within this class of models, PROC HPPANEL provides one specification test statistic, the Hausman *m* statistic, which provides information about the appropriateness of the random-effects specification. The *m* statistic is based on the idea that, under the null hypothesis of no correlation between the effects variables and the regressors, ordinary least squares (OLS) and generalized least squares (GLS) are consistent. However, OLS is inefficient. Hence, a test can be based on the result that the covariance between an efficient estimator and its difference from an inefficient estimator is 0. Rejection of the null hypothesis might suggest that the fixed-effects model is more appropriate.

The HPPANEL procedure also provides the Buse R-square measure. This number is interpreted as a measure of the proportion of the transformed sum of squares of the dependent variable that is attributable to the influence of the independent variables. For OLS estimation, the Buse R-square measure is equivalent to the usual R-square measure.

Unbalanced Data

The HPPANEL procedure can process data that have different numbers of time series observations across different cross sections. The missing time series observations are recognized by the absence of time series ID variable values in some of the cross sections in the input data set. Moreover, if an observation that has a particular time series ID value and cross-sectional ID value is present in the input data set but one or more of the model variables are missing, that time series point is treated as missing for that cross section.

One-Way Fixed-Effects Model

The specification for the one-way fixed-effects model is

$$u_{it} = \gamma_i + \epsilon_{it}$$

where the γ_i are nonrandom parameters to be estimated.

Let $\mathbf{Q_0} = \mathrm{diag}(\mathbf{E}_{T_i})$, with $\bar{\mathbf{J}}_{T_i} = \mathbf{J}_{T_i}/T_i$ and $\mathbf{E}_{T_i} = \mathbf{I}_{T_i} - \bar{\mathbf{J}}_{T_i}$, where $\mathbf{J}_{T_i}$ is a matrix of T_i ones.

The matrix $\mathbf{Q_0}$ represents the within transformation. In the one-way model, the within transformation is the conversion of the raw data to deviations from a cross section's mean. The vector $\tilde{\mathbf{x}}_{it}$ is a row of the general matrix $\mathbf{X}_s$, where the subscripted *s* implies that the constant (column of ones) is missing.

Let $\tilde{\mathbf{X}}_s = \mathbf{Q_0}\mathbf{X}_s$ and $\tilde{\mathbf{y}} = \mathbf{Q_0}\mathbf{y}$. The estimator of the slope coefficients is given by

$$\tilde{\beta}_s = (\tilde{\mathbf{X}}_s'\tilde{\mathbf{X}}_s)^{-1}\tilde{\mathbf{X}}_s'\tilde{\mathbf{y}}$$

After the slope estimates have been calculated, the estimation of an intercept or the cross-sectional fixed effects is handled as follows. First, you obtain the cross-sectional effects:

$$\gamma_i = \bar{y}_{i\cdot} - \tilde{\beta}_s\bar{x}_{i\cdot} \quad \text{for} \quad i = 1 \ldots N$$

If the NOINT option is specified, then the dummy variables' coefficients are set equal to the fixed effects. If you want an intercept, then the *i*th dummy variable is obtained from the following expression:

$$D_i = \gamma_i - \gamma_N \quad \text{for} \quad i = 1 \ldots N-1$$

The intercept is the *N*th fixed effect γ_N.

The within-model sum of squared errors is

$$\text{SSE} = \sum_{i=1}^{N} \sum_{t=1}^{T_i} (y_{it} - \gamma_i - \mathbf{X}_s \tilde{\beta}_s)^2$$

The estimated error variance can be written as

$$\hat{\sigma}_\epsilon^2 = \text{SSE}/(M - N - (K-1))$$

Alternatively, an equivalent way to express the error variance is

$$\hat{\sigma}_\epsilon^2 = \tilde{\mathbf{u}}' \mathbf{Q}_0 \tilde{\mathbf{u}}/(M - N - (K-1))$$

where the residuals $\tilde{\mathbf{u}}$ are given by $\tilde{\mathbf{u}} = (\mathbf{I}_M - \mathbf{j}_M \mathbf{j}'_M/M)(\mathbf{y} - \mathbf{X}_s \tilde{\beta}_s)$ if there is an intercept and by $\tilde{\mathbf{u}} = (\mathbf{y} - \mathbf{X}_s \tilde{\beta}_s)$ if there is not. The drawback is that the formula changes (but the results do not) with the inclusion of a constant.

The variance covariance matrix of $\tilde{\beta}_s$ is given by

$$\text{Var}\left[\tilde{\beta}_s\right] = \hat{\sigma}_\epsilon^2 (\tilde{\mathbf{X}}'_s \tilde{\mathbf{X}}_s)^{-1}$$

The covariance of the dummy variables and the dummy variables with the $\tilde{\beta}_s$ depends on whether the intercept is included in the model. For more information, see the section "One-Way Fixed-Effects Model (FIXONE and FIXONETIME Options)" (Chapter 25, *SAS/ETS User's Guide*).

Alternatively, the FIXONETIME model option estimates a one-way model in which the heterogeneity comes from time effects. This option is analogous to re-sorting the data by time and then by cross section, and then running a FIXONE model. The advantage of using the FIXONETIME option is that sorting is avoided and the model remains labeled correctly.

Two-Way Fixed-Effects Model

The specification for the two-way fixed-effects model is

$$u_{it} = \gamma_i + \alpha_t + \epsilon_{it}$$

where the γ_i and α_t are nonrandom parameters to be estimated.

If you do not specify the NOINT option (which suppresses the intercept) in the MODEL statement, the estimates for the fixed effects are reported under the restriction that $\gamma_N = 0$ and $\alpha_T = 0$. If you specify the NOINT option to suppress the intercept, only the restriction $\alpha_T = 0$ is imposed.

Balanced Panels

Assume that the data are balanced (for example, all cross sections have T observations). Then you can write

$$\tilde{y}_{it} = y_{it} - \bar{y}_{i\cdot} - \bar{y}_{\cdot t} + \bar{\bar{y}}$$

$$\tilde{\mathbf{x}}_{it} = \mathbf{x}_{it} - \bar{\mathbf{x}}_{i\cdot} - \bar{\mathbf{x}}_{\cdot t} + \bar{\bar{\mathbf{x}}}$$

where the symbols are as follows:

- y_{it} and $\mathbf{x}_{it}$ are the dependent variable (a scalar) and the explanatory variables (a vector whose columns are the explanatory variables, not including a constant), respectively
- $\bar{y}_{i\cdot}$ and $\bar{\mathbf{x}}_{i\cdot}$ are cross section means
- $\bar{y}_{\cdot t}$ and $\bar{\mathbf{x}}_{\cdot t}$ are time means
- $\bar{\bar{y}}$ and $\bar{\bar{\mathbf{x}}}$ are the overall means

The two-way fixed-effects model is simply a regression of $\tilde{y}_{it}$ on $\tilde{\mathbf{x}}_{it}$. Therefore, the two-way β is given by

$$\tilde{\beta}_s = \left(\tilde{\mathbf{X}}'\tilde{\mathbf{X}}\right)^{-1}\tilde{\mathbf{X}}'\tilde{\mathbf{y}}$$

The following calculations of cross-sectional dummy variables, time dummy variables, and intercepts are similar to how they are calculated in the one-way model:

First, you obtain the net cross-sectional and time effects. Denote the cross-sectional effects by γ and the time effects by α. These effects are calculated from the following relations:

$$\hat{\gamma}_i = \left(\bar{y}_{i\cdot} - \bar{\bar{y}}\right) - \tilde{\beta}_s\left(\bar{x}_{i\cdot} - \bar{\bar{x}}\right)$$

$$\hat{\alpha}_t = \left(\bar{y}_{\cdot t} - \bar{\bar{y}}\right) - \tilde{\beta}_s\left(\bar{x}_{\cdot t} - \bar{\bar{x}}\right)$$

Use the superscript C and T to denote the cross-sectional dummy variables and time dummy variables, respectively. Under the NOINT option, the following equations produce the dummy variables:

$$D_i^C = \hat{\gamma}_i + \hat{\alpha}_T$$

$$D_t^T = \hat{\alpha}_t - \hat{\alpha}_T$$

When an intercept is specified, the equations for dummy variables and intercept are

$$D_i^C = \hat{\gamma}_i - \hat{\gamma}_N$$

$$D_t^T = \hat{\alpha}_t - \hat{\alpha}_T$$

$$\text{Intercept} = \hat{\gamma}_N + \hat{\alpha}_T$$

The sum of squared errors is

$$\text{SSE} = \sum_{i=1}^{N}\sum_{t=1}^{T_i}(y_{it} - \gamma_i - \alpha_t - \mathbf{X}_s\tilde{\beta}_s)^2$$

The estimated error variance is

$$\hat{\sigma}_\epsilon^2 = \mathrm{SSE}/(M - N - T - (K-1))$$

With or without a constant, the covariance matrix of $\tilde{\beta}_s$ is given by

$$\mathrm{Var}\left[\tilde{\beta}_s\right] = \hat{\sigma}_\epsilon^2(\tilde{\mathbf{X}}_s'\tilde{\mathbf{X}}_s)^{-1}$$

For information about the covariance matrix that is related to dummy variables, see the section "Two-Way Random-Effects Model (RANTWO Option)" (Chapter 25, *SAS/ETS User's Guide*).

Unbalanced Panels

Let $\mathbf{X}_*$ and $\mathbf{y}_*$ be the independent and dependent variables, respectively, that are arranged by time and by cross section within each time period. (Note that the input data set that the PANEL procedure uses must be sorted by cross section and then by time within each cross section.) Let M_t be the number of cross sections that are observed in year *t*, and let $\sum_t M_t = M$. Let $\mathbf{D}_t$ be the $M_t \times N$ matrix that is obtained from the $N \times N$ identity matrix from which rows that correspond to cross sections that are not observed at time *t* have been omitted. Consider

$$\mathbf{Z} = (\mathbf{Z}_1, \mathbf{Z}_2)$$

where $\mathbf{Z}_1 = (\mathbf{D}_1', \mathbf{D}_2', \ldots, \mathbf{D}_T')'$ and $\mathbf{Z}_2 = \mathrm{diag}(\mathbf{D}_1\mathbf{j}_N, \mathbf{D}_2\mathbf{j}_N, \ldots, \mathbf{D}_T\mathbf{j}_N)$. The matrix $\mathbf{Z}$ contains the dummy variable structure for the two-way model.

Let

$$\begin{aligned}
\mathbf{\Delta}_N &= \mathbf{Z}_1'\mathbf{Z}_1 \\
\mathbf{\Delta}_T &= \mathbf{Z}_2'\mathbf{Z}_2 \\
\mathbf{A} &= \mathbf{Z}_2'\mathbf{Z}_1 \\
\bar{\mathbf{Z}} &= \mathbf{Z}_2 - \mathbf{Z}_1\Delta_N^{-1}\mathbf{A}' \\
\mathbf{Q} &= \Delta_T - \mathbf{A}\Delta_N^{-1}\mathbf{A}' \\
\mathbf{P} &= (\mathbf{I}_M - \mathbf{Z}_1\Delta_N^{-1}\mathbf{Z}_1') - \bar{\mathbf{Z}}\mathbf{Q}^{-1}\bar{\mathbf{Z}}'
\end{aligned}$$

The estimate of the regression slope coefficients is given by

$$\tilde{\beta}_s = (\mathbf{X}_{*s}'\mathbf{P}\mathbf{X}_{*s})^{-1}\mathbf{X}_{*s}'\mathbf{P}\mathbf{y}_*$$

where $\mathbf{X}_{*s}$ is the $\mathbf{X}_*$ matrix without the vector of 1s.

The estimator of the error variance is

$$\hat{\sigma}_\epsilon^2 = \tilde{\mathbf{u}}'\mathbf{P}\tilde{\mathbf{u}}/(M - T - N + 1 - (K-1))$$

where the residuals are given by $\tilde{\mathbf{u}} = (\mathbf{I}_M - \mathbf{j}_M\mathbf{j}_M'/M)(\mathbf{y}_* - \mathbf{X}_{*s}\tilde{\beta}_s)$ if there is an intercept in the model and by $\tilde{\mathbf{u}} = \mathbf{y}_* - \mathbf{X}_{*s}\tilde{\beta}_s$ if there is no intercept.

The actual implementation is quite different from the theory. For more information, see the section "Two-Way Fixed-Effects Model (FIXTWO Option)" (Chapter 25, *SAS/ETS User's Guide*).

One-Way Random-Effects Model

The specification for the one-way random-effects model is

$$u_{it} = v_i + \epsilon_{it}$$

Let $\mathbf{Z_0} = \mathrm{diag}(\mathbf{J}_{T_i})$, $\mathbf{P_0} = \mathrm{diag}(\bar{\mathbf{J}}_{T_i})$, and $\mathbf{Q_0} = \mathrm{diag}(\mathbf{E}_{T_i})$, with $\bar{\mathbf{J}}_{T_i} = \mathbf{J}_{T_i}/T_i$ and $\mathbf{E}_{T_i} = \mathbf{I}_{T_i} - \bar{\mathbf{J}}_{T_i}$. Define $\tilde{\mathbf{X}}_s = \mathbf{Q_0}\mathbf{X}_s$. Also define $\tilde{\mathbf{y}} = \mathbf{Q_0}\mathbf{y}$ and $\mathbf{J}$ as a vector of 1s whose length is T_i.

In the one-way model, estimation proceeds in a two-step fashion. First, you obtain estimates of the variance of the σ_ϵ^2 and σ_v^2. There are multiple ways to derive these estimates; PROC HPPANEL provides four options. For more information, see the section "One-Way Random-Effects Model (RANONE Option)" (Chapter 25, *SAS/ETS User's Guide*).

After the variance components are calculated from any method, the next task is to estimate the regression model of interest. For each individual, you form a weight (θ_i),

$$\theta_i = 1 - \sigma_\epsilon / w_i$$

$$w_i^2 = T_i \sigma_v^2 + \sigma_\epsilon^2$$

where T_i is the *i*th cross section's time observations.

Taking the θ_i, you form the partial deviations,

$$\tilde{y}_{it} = y_{it} - \theta_i \bar{y}_{i\cdot}$$

$$\tilde{x}_{it} = x_{it} - \theta_i \bar{x}_{i\cdot}$$

where $\bar{y}_{i\cdot}$ and $\bar{x}_{i\cdot}$ are cross section means of the dependent variable and independent variables (including the constant if any), respectively.

The random-effects β is then the result of simple OLS on the transformed data.

Two-Way Random-Effects Model

The specification for the two-way random-effects model is

$$u_{it} = v_i + e_t + \epsilon_{it}$$

As it does for the one-way random-effects model, the HPPANEL procedure provides four options for variance component estimators. However, unbalanced panels present some special concerns that do not occur for one-way random-effects models.

Let $\mathbf{X}_*$ and $\mathbf{y}_*$ be the independent and dependent variables that are arranged by time and by cross section within each time period. (Note that the input data set that the PANEL procedure uses must be sorted by cross section and then by time within each cross section.) Let M_t be the number of cross sections that are observed in time *t*, and let $\sum_t M_t = M$. Let $\mathbf{D}_t$ be the $M_t \times N$ matrix that is obtained from the $N \times N$ identity matrix from which rows that correspond to cross sections that are not observed at time *t* have been omitted. Consider

$$\mathbf{Z} = (\mathbf{Z}_1, \mathbf{Z}_2)$$

where $\mathbf{Z}_1 = (\mathbf{D}_1', \mathbf{D}_2', \ldots, \mathbf{D}_T')'$ and $\mathbf{Z}_2 = \mathrm{diag}(\mathbf{D}_1\mathbf{j}_N, \mathbf{D}_2\mathbf{j}_N, \ldots, \mathbf{D}_T\mathbf{j}_N)$.

The matrix $\mathbf{Z}$ contains the dummy variable structure for the two-way model.

For notational ease, let

$$\Delta_N = \mathbf{Z}_1'\mathbf{Z}_1$$

$$\Delta_T = \mathbf{Z}_2'\mathbf{Z}_2$$

$$\mathbf{A} = \mathbf{Z}_2'\mathbf{Z}_1$$

$$\bar{\mathbf{Z}} = \mathbf{Z}_2 - \mathbf{Z}_1\Delta_N^{-1}\mathbf{A}'$$

$$\bar{\Delta}_1 = \mathbf{I}_M - \mathbf{Z}_1\Delta_N^{-1}\mathbf{Z}_1'$$

$$\bar{\Delta}_2 = \mathbf{I}_M - \mathbf{Z}_2\Delta_T^{-1}\mathbf{Z}_2'$$

$$\mathbf{Q} = \Delta_T - \mathbf{A}\Delta_N^{-1}\mathbf{A}'$$

$$\mathbf{P} = (\mathbf{I}_M - \mathbf{Z}_1\Delta_N^{-1}\mathbf{Z}_1') - \bar{\mathbf{Z}}\mathbf{Q}^{-1}\bar{\mathbf{Z}}'$$

PROC HPPANEL provides four methods to estimate the variance components. For more information, see the section "Two-Way Random-Effects Model (RANTWO Option)" (Chapter 25, *SAS/ETS User's Guide*).

After the estimates of the variance components are calculated, you can proceed to the final estimation. If the panel is balanced, partial mean deviations are used as follows

$$\tilde{y}_{it} = y_{it} - \theta_1\bar{y}_{i\cdot} - \theta_2\bar{y}_{\cdot t} + \theta_3\bar{y}_{\cdot\cdot}$$

$$\tilde{x}_{it} = x_{it} - \theta_1\bar{x}_{i\cdot} - \theta_2\bar{x}_{\cdot t} + \theta_3\bar{x}_{\cdot\cdot}$$

The θ estimates are obtained from

$$\theta_1 = 1 - \frac{\sigma_\epsilon}{\sqrt{T\sigma_\nu^2 + \sigma_\epsilon^2}}$$

$$\theta_2 = 1 - \frac{\sigma_\epsilon}{\sqrt{N\sigma_e^2 + \sigma_\epsilon^2}}$$

$$\theta_3 = \theta_1 + \theta_2 + \frac{\sigma_\epsilon}{\sqrt{T\sigma_\nu^2 + N\sigma_e^2 + \sigma_\epsilon^2}} - 1$$

With these partial deviations, PROC HPPANEL uses OLS on the transformed series (including an intercept if you want).

The case of an unbalanced panel is somewhat more complicated. Wansbeek and Kapteyn show that the inverse of Ω can be written as

$$\sigma_\epsilon^2\Omega^{-1} = \mathbf{V} - \mathbf{V}\mathbf{Z}_2\tilde{\mathbf{P}}^{-1}\mathbf{Z}_2'\mathbf{V}$$

with the following:

$$\begin{aligned}
\mathbf{V} &= \mathbf{I}_M - \mathbf{Z}_1\tilde{\Delta}_N^{-1}\mathbf{Z}_1' \\
\tilde{\mathbf{P}} &= \tilde{\Delta}_T - \mathbf{A}\tilde{\Delta}_N^{-1}\mathbf{A}' \\
\tilde{\Delta}_N &= \Delta_N + \left(\frac{\sigma_\epsilon^2}{\sigma_\nu^2}\right)\mathbf{I}_N \\
\tilde{\Delta}_T &= \Delta_T + \left(\frac{\sigma_\epsilon^2}{\sigma_e^2}\right)\mathbf{I}_T
\end{aligned}$$

By using the inverse of the covariance matrix of the error, it becomes possible to complete GLS on the unbalanced panel.

Between Estimators

The between-groups estimator is the regression of the cross section means of $\mathbf{y}$ on the cross section means of $\tilde{\mathbf{X}}_s$. In other words, you fit the following regression:

$$\bar{y}_{i\cdot} = \bar{\mathbf{x}}_{i\cdot}\beta^{BG} + \eta_i$$

The between-time-periods estimator is the regression of the time means of $\mathbf{y}$ on the time means of $\tilde{\mathbf{X}}_s$. In other words, you fit the following regression:

$$\bar{y}_{\cdot t} = \bar{\mathbf{x}}_{\cdot t}\beta^{BT} + \zeta_t$$

In both cases, the error is assumed to be normally distributed with mean zero and a constant variance.

Pooled Estimator

The pooled estimator is simply linear regression that is run on all the data, without regard to cross section or time:

$$y_{it} = \mathbf{x}_{it}\beta^{P} + u_{it}$$

The error is assumed to be normally distributed with mean zero and a constant variance.

Linear Hypothesis Testing

For a linear hypothesis of the form $\mathbf{R}\,\beta = \mathbf{r}$, where $\mathbf{R}$ is $J \times K$ and $\mathbf{r}$ is $J \times 1$, the F-statistic with $J, M - K$ degrees of freedom is computed as

$$(\mathbf{R}\beta - \mathbf{r})'[\mathbf{R}\hat{\mathbf{V}}\mathbf{R}']^{-1}(\mathbf{R}\beta - \mathbf{r})$$

However, it is also possible to write the F statistic as

$$F = \frac{(\hat{\mathbf{u}}_*'\hat{\mathbf{u}}_* - \hat{\mathbf{u}}'\hat{\mathbf{u}})/J}{\hat{\mathbf{u}}'\hat{\mathbf{u}}/(M-K)}$$

where

- $\hat{\mathbf{u}}_*$is the residual vector from the restricted regression
- $\hat{\mathbf{u}}$ is the residual vector from the unrestricted regression
- J is the number of restrictions

- $M - K$ are the degrees of freedom, M is the number of observations, and K is the number of parameters in the model

The Wald, likelihood ratio (LR), and Lagrange multiplier (LM) tests are all related to the F test. You use this relationship of the F test to the likelihood ratio and Lagrange multiplier tests. The Wald test is calculated from its definition.

The Wald test statistic is

$$W = (\mathbf{R}\beta - \mathbf{r})'[\mathbf{R}\hat{\mathbf{V}}\mathbf{R}']^{-1}(\mathbf{R}\beta - \mathbf{r})$$

The likelihood ratio is

$$\mathrm{LR} = M \ln\left[1 + \frac{1}{M-K} JF\right]$$

The Lagrange multiplier test statistic is

$$\mathrm{LM} = M\left[\frac{JF}{M - K + JF}\right]$$

where JF represents the number of restrictions multiplied by the result of the F test.

The distribution of these test statistics is the χ^2 distribution whose degrees of freedom equal the number of restrictions imposed (J). The three tests are asymptotically equivalent, but they have differing small-sample properties. Greene (2000, p. 392) and Davidson and MacKinnon (1993, pp. 456–458) discuss the small-sample properties of these statistics.

Specification Tests

The HPPANEL procedure outputs one specification test for random effects: the Hausman (1978) specification test (m statistic) can be used to test hypotheses in terms of bias or inconsistency of an estimator. This test was also proposed by Wu (1973) and further extended in Hausman and Taylor (1982). Hausman's m statistic is as follows.

Consider two estimators, $\hat{\beta}_a$ and $\hat{\beta}_b$, which under the null hypothesis are both consistent, but only $\hat{\beta}_a$ is asymptotically efficient. Under the alternative hypothesis, only $\hat{\beta}_b$ is consistent. The m statistic is

$$m = (\hat{\beta}_b - \hat{\beta}_a)'(\hat{\mathbf{S}}_b - \hat{\mathbf{S}}_a)^{-1}(\hat{\beta}_b - \hat{\beta}_a)$$

where $\hat{\mathbf{S}}_b$ and $\hat{\mathbf{S}}_a$ are consistent estimates of the asymptotic covariance matrices of $\hat{\beta}_b$ and $\hat{\beta}_a$. Then m is distributed as χ^2 with k degrees of freedom, where k is the dimension of $\hat{\beta}_a$ and $\hat{\beta}_b$.

In the random-effects specification, the null hypothesis of no correlation between effects and regressors implies that the OLS estimates of the slope parameters are consistent and inefficient but the GLS estimates of the slope parameters are consistent and efficient. This facilitates a Hausman specification test. The reported degrees of freedom for the χ^2 statistic are equal to the number of slope parameters. If the null hypothesis holds, the random-effects specification should be used.

OUTPUT OUT= Data Set

PROC HPPANEL writes the initial data of the estimated model, predicted values, and residuals to an output data set when the OUT= option is specified in the OUTPUT statement. The OUT= data set contains the following variables:

CSID is the value of the cross section ID. The variable name is the one specified in the id statement.

TSID is the value of the time period in the dynamic model. The variable name is the one specified in the id statement.

Regressors are the values of regressor variables that are specified in the COPYVAR option.

Pred is the predicted value of dependent variable. This column is output only if the PRED option is specified.

Resid is the residual from the regression. This column is output only if the RESIDUAL option is specified.

OUTEST= Data Set

PROC HPPANEL writes the parameter estimates to an output data set when the OUTEST= option is specified in the PROC HPPANEL statement. The OUTEST= data set contains the following variables in the PROC statement:

METHOD is a character variable that identifies the estimation method.

TYPE is a character variable that identifies the type of observation. Values of the _TYPE_ variable are CORRB, COVB, CSPARMS, STD, and the type of model estimated. The CORRB observation contains correlations of the parameter estimates; the COVB observation contains covariances of the parameter estimates; the STD observation indicates the row of standard deviations of the corresponding coefficients; and the type of model estimated observation contains the parameter estimates.

NAME is a character variable that contains the name of a regressor variable for COVB and CORRB observations and is left blank for other observations. The _NAME_ variable is used in conjunction with the _TYPE_ values COVB and CORRB to identify rows of the correlation or covariance matrix.

DEPVAR is a character variable that contains the name of the response variable.

MSE is the mean square error of the transformed model.

VARCS is the variance component estimate due to cross sections. The _VARCS_ variable is included in the OUTEST= data set when the RANONE option is specified in the MODEL or PROC HPPANEL statement.

VARTS is the variance component estimate due to time series. The _VARTS_ variable is included in the OUTEST= data set when the RANTWO option is specified in the MODEL or PROC HPPANEL statement.

VARERR is the variance component estimate due to error. The _VARERR_ variable is included in the OUTEST= data set when the RANONE or RANTWO option is specified in the MODEL or PROC HPPANEL statement.

Intercept is the intercept parameter estimate. (The intercept is missing for models when the NOINT option is specified in the MODEL statement.)

Regressors are the regressor variables that are specified in the MODEL statement. The regressor variables in the OUTEST= data set contain the corresponding parameter estimates, and the corresponding covariance or correlation matrix elements for _TYPE_=COVB and _TYPE_=CORRB observations.

Printed Output

The printed output from PROC HPPANEL includes the following:

- the model information, which includes the data source, the dependent variable name, the estimation method used, and for random-effects model analysis, the variance component estimation method.
- the number of observations
- the fit statistics, which include the sum of squared error (SSE), the degree of freedom for error (DFE), the mean square error (MSE), the root mean square error (RMSE), and the R-square
- the error components estimates for random-effects model
- the Hausman test statistics, which include the degree of freedom (DF), the test statistics, and the *p*-value.
- the regression parameter estimates and analysis, which include for each regressor the name of the regressor, the degrees of freedom, the parameter estimate, the standard error of the estimate, a *t* statistic for testing whether the estimate is significantly different from 0, and the significance probability of the *t* statistic

Optionally, PROC HPPANEL prints the following:

- the covariance and correlation of the resulting regression parameter estimates
- the WALD, LR, and LM test statistics for linear equality restrictions that are specified in the TEST statements
- the timing breakdown of the procedure steps

ODS Table Names

PROC HPPANEL assigns a name to each table it creates. You can use these names to refer to the table when you use the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in Table 6.2.

Table 6.2 ODS Tables Produced in PROC HPPANEL

ODS Table Name	Description	Option
ODS Tables Created by the MODEL Statement		
ModelInfo	Model information	Default
PerformanceInfo	Performance information	Default
Nobs	Number of observations	Default
FitStatistics	Fit statistics	Default
ParameterEstimates	Parameter estimates	Default
CovB	Covariance of parameter estimates	COVB
CorrB	Correlations of parameter estimates	CORRB
RandomEffectsTest	Hausman test for random effects	RANONE, RANTWO
ODS Tables Created by the TEST Statement		
TestResults	Test results	
ODS Tables Created by the PERFORMANCE Statement		
Timing	Timing Table	

Example: HPPANEL Procedure

Example 6.1: One-Way Random-Effects High-Performance Model

This example shows the use of the one-way random-effects model that is available in the HPPANEL procedure; the example emphasizes processing a large data set and the performance improvements that are achieved by executing in a high-performance distributed environment.

The following DATA step generates five million observations from one-way panel data that includes 50,000 cross sections and 100 time periods:

```
data hppan_ex01 (keep = cs ts y x1-x10);
   retain seed 55371;
   array x[10];
   label y  = 'Dependent Variable';
   do cs = 1 to 50000;
      dummy = 10 * rannor(seed);
      do ts = 1 to 100;
      /*- generate regressors and compute the structural */
      /*- part of the dependent variable                 */
         y = 5;
         do k = 1 to 10;
            x[k] = -1 + 2 * ranuni(seed);
            y = y + x[k] * k;
         end;

         /*- add an error term, such that e - N(0,100)   -------*/
         y = y + 10 * rannor(seed);
         /*- add a random effect, such that v - N(0,100) -------*/
         y = y + dummy;
         output;
      end;
   end;
run;
```

The estimation is executed in distributed mode on a grid with ten nodes, with one thread per node. To run the following statements successfully, you need to set the macro variables GRIDHOST and GRIDINSTALLLOC to resolve to appropriate values, or you can replace the references to the macro variables in the example with the appropriate values.

```
%let GRIDHOST      = <<your grid host>>;
%let GRIDINSTALLOC = <<your grid install location>>;

option set = GRIDHOST       = "&GRIDHOST";
option set = GRIDINSTALLLOC = "&GRIDINSTALLLOC";

proc hppanel data=hppan_ex01;
   id  cs ts;
   model y = x1-x10 / ranone;
   performance nodes = 10 threads = 1 details
               host="&GRIDHOST" install="&GRIDINSTALLLOC";
run;
```

In Output 6.1.1, the "Performance Information" table shows that the model was estimated on the grid that is defined in the macro variable named GRIDHOST in a distributed environment with ten nodes, and one thread per node. The grid installation location is defined in the macro variable named GRIDINSTALLLOC.

Output 6.1.1 Grid Information with Ten Nodes and One Thread per Node

Performance Information	
Host Node	<< your grid host >>
Install Location	<< your grid install location >>
Execution Mode	Distributed
Number of Compute Nodes	10
Number of Threads per Node	1

Output 6.1.2 shows the results for the one-way random-effects model. The "Model Information" table shows detailed information about the model. The "Number of Observations" table indicates that all five million observations were used to fit the model. All parameter estimates in the "Parameter Estimates" table are highly significant and correspond to the theoretical values that were set for them during the data generating process. In the "Procedure Task Timing" table, you can see that for five million observations, computing the moments took 2.28 seconds, and the time taken for cross-product accumulation was 0.57 seconds.

Output 6.1.2 One-Way Random-Effects Model

Model Information	
Data Source	HPPAN_EX01
Response Variable	y
Model	RANONE
Variance Component	WANSBEEK

Number of Observations	
Number of Observations Read	5000000
Number of Observations Used	5000000
Number of Cross Sections	50000
Number of Time Series	100

Fit Statistics	
Sum of Squared Error	4.9976E8
Degrees of Freedom	4999989
Mean Squared Error	99.952
Root Mean Squared Error	9.9976
R-Square	0.559771

Variance Component Estimates	
Variance Component for Cross Sections	99.9117
Variance Component for Error	99.9520

Hausman Test for Random Effects			
Coefficients	**DF**	**m Value**	**Pr > m**
10	10	14.04	0.1713

Output 6.1.2 *continued*

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	4.96955	0.04492	110.62	<.0001
x1	1	1.00902	0.00778	129.69	<.0001
x2	1	1.99743	0.00778	256.66	<.0001
x3	1	3.00116	0.00778	385.64	<.0001
x4	1	3.99847	0.00778	513.68	<.0001
x5	1	4.99497	0.00778	641.81	<.0001
x6	1	6.01034	0.00778	772.12	<.0001
x7	1	6.99770	0.00778	899.39	<.0001
x8	1	7.98897	0.00778	1026.61	<.0001
x9	1	9.00692	0.00778	1157.12	<.0001
x10	1	10.00563	0.00778	1285.47	<.0001

Procedure Task Timing

Task	Seconds	Percent
Data Read and Variable Levelization	0.38	11.86%
Communication to Client	0.00	0.00%
Computing Moments	2.28	70.46%
Cross-Product Accumulation	0.57	17.67%

For comparison, you now fit a pooled regression estimation on the same data, again using a grid of 10 nodes with one thread each. The following SAS statements perform the estimation on the grid:

```
proc hppanel data=hppan_ex01;
   id  cs ts;
   model y = x1-x10 / pooled;
   performance nodes = 10 threads = 1 details
               host="&GRIDHOST" install="&GRIDINSTALLLOC";
run;
```

Based on Output 6.1.3, you find that the parameter estimates are similar to those from the random-effects estimator. You also find that the timings are similar, indicating that the bulk of the computational effort is due to tasks common to both random-effects estimation and standard OLS regression. In both cases, estimation is dominated by the calculation of sums of squares and other moment terms, over the whole data set.

Output 6.1.3 Pooled Regression Model

The HPPANEL Procedure

Model Information

Data Source	HPPAN_EX01
Response Variable	y
Model	POOLED

Output 6.1.3 *continued*

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	4.96957	0.00632	786.03	<.0001
x1	1	1.01251	0.01095	92.49	<.0001
x2	1	1.98374	0.01095	181.17	<.0001
x3	1	3.00294	0.01095	274.23	<.0001
x4	1	3.99649	0.01095	364.90	<.0001
x5	1	5.00187	0.01095	456.77	<.0001
x6	1	5.99952	0.01095	547.77	<.0001
x7	1	7.00478	0.01095	639.88	<.0001
x8	1	7.97232	0.01095	728.13	<.0001
x9	1	9.01244	0.01095	822.90	<.0001
x10	1	10.01578	0.01095	914.52	<.0001

Procedure Task Timing

Task	Seconds	Percent
Data Read and Variable Levelization	0.38	13.46%
Communication to Client	0.00	0.00%
Computing Moments	2.26	78.90%
Cross-Product Accumulation	0.22	7.63%

References

Davidson, R., and MacKinnon, J. G. (1993). *Estimation and Inference in Econometrics*. New York: Oxford University Press.

Fuller, W. A., and Battese, G. E. (1974). "Estimation of Linear Models with Crossed-Error Structure." *Journal of Econometrics* 2:67–78.

Greene, W. H. (1990). *Econometric Analysis*. New York: Macmillan.

Greene, W. H. (2000). *Econometric Analysis*. 4th ed. Upper Saddle River, NJ: Prentice-Hall.

Hausman, J. A. (1978). "Specification Tests in Econometrics." *Econometrica* 46:1251–1271.

Hausman, J. A., and Taylor, W. E. (1982). "A Generalized Specification Test." *Economics Letters* 8:239–245.

Nerlove, M. (1971). "Further Evidence on the Estimation of Dynamic Relations from a Time Series of Cross Sections." *Econometrica* 39:359–382.

Wallace, T., and Hussain, A. (1969). "The Use of Error Components Model in Combining Cross Section with Time Series Data." *Econometrica* 37:55–72.

Wansbeek, T., and Kapteyn, A. (1989). "Estimation of the Error-Components Model with Incomplete Panels." *Journal of Econometrics* 41:341–361.

Wu, D. M. (1973). "Alternative Tests of Independence between Stochastic Regressors and Disturbances." *Econometrica* 41:733–750.

Chapter 7
The HPQLIM Procedure

Contents

Overview: HPQLIM Procedure

The HPQLIM (high-performance qualitative and limited dependent variable model) procedure is a high-performance version of the QLIM procedure in SAS/ETS software, which analyzes univariate limited dependent variable models in which dependent variables are observed only in a limited range of values. Unlike the QLIM procedure, which can be run only on an individual workstation, the HPQLIM procedure takes advantage of a 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 provides a dramatic gain in performance.

With the HPQLIM procedure you can read and write data in distributed form and perform analyses in distributed mode and single-machine mode. For more information about how to affect the execution mode of SAS high-performance analytical procedures, see the section "Processing Modes" on page 6.

The HPQLIM procedure is specifically designed to operate in the high-performance distributed environment. It can use maximum likelihood or Bayesian methods. In both cases, the likelihood evaluation is performed in a distributed environment. By default, PROC HPQLIM uses multiple threads to perform computations.

The HPQLIM procedure is similar in use to the other SAS procedures that support regression or simultaneous equations models. 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 with the CENSORED or TRUNCATED option in the ENDOGENOUS or MODEL statement. For example, the two-limit censored model requires two variables: one that contains the lower (bottom) bound and one that contains the upper (top) bound. The following statements execute the model in the distributed computing environment with two threads and four nodes:

```
proc hpqlim data=a;
   model y = x1 x2 x3;
   endogenous y ~ censored(lb=bottom ub=top);
   performance nthreads=2 nodes=4 details;
run;
```

The bounds can be numbers if they are fixed for all observations in the data set. For example, the standard Tobit model can be specified as follows:

```
proc hpqlim data=a;
   model y = x1 x2 x3;
   endogenous y ~ censored(lb=0);
   performance nthreads=2 nodes=4 details;
run;
```

PROC HPQLIM Features

The HPQLIM procedure supports the following models:

- linear regression 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 HPQLIM procedure allows for a number of different linear and nonlinear variance specifications.

The HPQLIM procedure also offers a class of models in which the dependent variable is censored or truncated from below or above or both. When a continuous dependent variable is observed only within a certain range, and values outside this range are not available, the HPQLIM procedure offers a class of models that adjust for truncation. In some cases, the dependent variable is continuous only in 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 it cannot be guaranteed that the predicted values from the model will fall in the appropriate region.

Stochastic frontier production and cost models 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.

The HPQLIM procedure can use maximum likelihood or Bayesian methods. Initial starting values for the nonlinear optimizations are typically calculated by OLS. Initial values for the Bayesian sampling are typically calculated by maximum likelihood.

Getting Started: HPQLIM Procedure

This example illustrates the use of the HPQLIM procedure. The data were originally published by Mroz (1987), and the following statements show a subset of that data set:

```
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
;
```

In these data, Hours is the number of hours that the wife worked outside the household in a given year, Yrs_Ed is the years of education, and Yrs_Exp is the years of work experience.

By 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 home at all ($y_i = 0$ is observed). This yields the following model:

$$y_i^* = \mathbf{x}_i'\boldsymbol{\beta} + \epsilon_i$$

$$y_i = \begin{cases} y_i^* & \text{if } y_i^* > 0 \\ 0 & \text{if } y_i^* \leq 0 \end{cases}$$

where $\epsilon_i \sim \text{iid} N(0, \sigma^2)$ and the set of explanatory variables is denoted by $\mathbf{x}_i$. The following statements fit a Tobit model to the hours worked with years of education and years of work experience as covariates:

```
/*-- Tobit Model --*/
proc hpqlim data=subset;
   model hours = yrs_ed yrs_exp;
   endogenous hours ~ censored(lb=0);
   performance nthreads=2 nodes=4 details;
run;
```

The output of the HPQLIM procedure is shown in Output 7.1.

Figure 7.1 Tobit Analysis Results

Estimating a Tobit Model

The HPQLIM Procedure

Model Fit Summary	
Number of Endogenous Variables	1
Endogenous Variable	Hours
Number of Observations	17
Log Likelihood	-74.93700
Maximum Absolute Gradient	1.18953E-6
Number of Iterations	23
Optimization Method	Quasi-Newton
AIC	157.87400
Schwarz Criterion	161.20685

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	-5598.295130	27.692220	-202.16	<.0001
Yrs_Ed	1	373.123254	53.988877	6.91	<.0001
Yrs_Exp	1	63.336247	36.551299	1.73	0.0831
_Sigma	1	1582.859635	390.076480	4.06	<.0001

The "Parameter Estimates" table contains four rows. The first three rows correspond to the vector estimate of the regression coefficients $\boldsymbol{\beta}$. The last row is called _Sigma, which corresponds to the estimate of the error variance σ.

Syntax: HPQLIM Procedure

The following statements are available in the HPQLIM procedure:

PROC HPQLIM *options* ;
BAYES < *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* > ;
PRIOR **_REGRESSORS** | *parameter-list* ∼ *distribution* ;
RESTRICT *restriction1* < , *restriction2* . . . > ;
TEST *options* ;
WEIGHT *variable* < / *option* > ;
PERFORMANCE < *performance-options* > ;

One MODEL statement is required. If a FREQ or WEIGHT statement is specified more than once, the variable that is specified in the first instance is used.

Functional Summary

Table 7.1 summarizes the statements and options used with the HPQLIM procedure.

Table 7.1 Functional Summary

Description	Statement	Option
Data Set Options		
Specifies the input data set	PROC HPQLIM	DATA=
Writes parameter estimates to an output data set	PROC HPQLIM	OUTEST=
Writes predictions to an output data set	OUTPUT	OUT=
Declaring the Role of Variables		
Specifies BY-group processing	BY	
Specifies a frequency variable	FREQ	
Specifies a weight variable	WEIGHT	NONORMALIZE
Printing Control Options		
Requests all printing options	PROC HPQLIM	PRINTALL
Prints the correlation matrix of the estimates	PROC HPQLIM	CORRB
Prints the covariance matrix of the estimates	PROC HPQLIM	COVB
Suppresses the normal printed output	PROC HPQLIM	NOPRINT

Table 7.1 *continued*

Description	Statement	Option
Plotting Options		
Displays plots	PROC HPQLIM	PLOTS=
Optimization Process Control Options		
Selects the iterative minimization method to use	PROC HPQLIM	METHOD=
Specifies the maximum number of iterations allowed	PROC HPQLIM	MAXITER=
Specifies the maximum number of function calls	PROC HPQLIM	MAXFUNC=
Specifies the upper limit of CPU time in seconds	PROC HPQLIM	MAXTIME=
Specifies an absolute convergence criterion	PROC HPQLIM	ABSCONV=
Specifies an absolute function convergence criterion	PROC HPQLIM	ABSFCONV=
Specifies an absolute gradient convergence criterion	PROC HPQLIM	ABSGCONV=
Specifies a relative function convergence criterion	PROC HPQLIM	FCONV=
Specifies a relative gradient convergence criterion	PROC HPQLIM	GCONV=
Specifies an absolute parameter convergence criterion	PROC HPQLIM	ABSXCONV=
Specifies a matrix singularity criterion	PROC HPQLIM	SINGULAR=
Sets boundary restrictions on parameters	BOUNDS	
Sets initial values for parameters	INIT	
Sets linear restrictions on parameters	RESTRICT	
Model Estimation Options		
Suppresses the intercept parameter	MODEL	NOINT
Specifies the method to calculate parameter covariance	PROC HPQLIM	COVEST=
Bayesian MCMC Options		
Specifies the initial values of the MCMC	INIT	
Specifies the maximum number of tuning phases	BAYES	MAXTUNE=
Specifies the minimum number of tuning phases	BAYES	MINTUNE=
Specifies the number of burn-in iterations	BAYES	NBI=
Specifies the number of iterations during the sampling phase	BAYES	NMC=
Specifies the number of iterations during the tuning phase	BAYES	NTU=
Controls options for constructing the initial proposal covariance matrix	BAYES	PROPCOV
Specifies the sampling scheme	BAYES	SAMPLING=
Specifies the random number generator seed	BAYES	SEED=
Controls the thinning of the Markov chain	BAYES	THIN=
Bayesian Summary Statistics and Convergence Diagnostic Options		
Displays convergence diagnostics	BAYES	DIAGNOSTICS=

Table 7.1 *continued*

Description	Statement	Option
Displays summary statistics of the posterior samples	BAYES	STATISTICS=
Bayesian Prior and Posterior Sample Options		
Specifies a SAS data set for the posterior samples	BAYES	OUTPOST=
Bayesian Analysis Options		
Specifies the normal prior distribution	PRIOR	NORMAL(MEAN=, VAR=)
Specifies the gamma prior distribution	PRIOR	GAMMA(SHAPE=, SCALE=)
Specifies the inverse gamma prior distribution	PRIOR	IGAMMA(SHAPE=, SCALE=)
Specifies the uniform prior distribution	PRIOR	UNIFORM(MIN=, MAX=)
Specifies the beta prior distribution	PRIOR	BETA(SHAPE1=, SHAPE2=, MIN=, MAX=)
Specifies the *t* prior distribution	PRIOR	T(LOCATION=, DF=)
Endogenous Variable Options		
Specifies a discrete variable	ENDOGENOUS	DISCRETE()
Specifies a censored variable	ENDOGENOUS	CENSORED()
Specifies a truncated variable	ENDOGENOUS	TRUNCATED()
Specifies a stochastic frontier variable	ENDOGENOUS	FRONTIER()
Heteroscedasticity Model Options		
Specifies the function for heteroscedasticity models	HETERO	LINK=
Squares the function for heteroscedasticity models	HETERO	SQUARE
Specifies no constant for heteroscedasticity models	HETERO	NOCONST
Output Control Options		
Outputs predicted values	OUTPUT	PREDICTED
Outputs the structured part	OUTPUT	XBETA
Outputs residuals	OUTPUT	RESIDUAL
Outputs the error standard deviation	OUTPUT	ERRSTD
Outputs marginal effects	OUTPUT	MARGINAL
Outputs probability for the current response	OUTPUT	PROB
Outputs probability for all responses	OUTPUT	PROBALL
Outputs the expected value	OUTPUT	EXPECTED
Outputs the conditional expected value	OUTPUT	CONDITIONAL
Outputs inverse Mills ratio	OUTPUT	MILLS
Outputs technical efficiency measures	OUTPUT	TE1
	OUTPUT	TE2

Table 7.1 *continued*

Description	Statement	Option
Includes covariances in the OUTEST= data set	PROC HPQLIM	COVOUT
Includes correlations in the OUTEST= data set	PROC HPQLIM	CORROUT
Test Request Options		
Requests Wald, Lagrange multiplier, and likelihood ratio tests	TEST	ALL
Requests the Wald test	TEST	WALD
Requests the Lagrange multiplier test	TEST	LM
Requests the likelihood ratio test	TEST	LR

PROC HPQLIM Statement

PROC HPQLIM *options* **;**

The PROC HPQLIM statement invokes the HPQLIM procedure. You can specify the following *options*.

Data Set Options

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

Output Data Set Options

OUTEST=*SAS-data-set*
writes the parameter estimates to an output data set.

COVOUT
writes the covariance matrix for the parameter estimates to the OUTEST= data set. This option is valid only if the OUTEST= option is specified.

CORROUT
writes the correlation matrix for the parameter estimates to the OUTEST= data set. This option is valid only if the OUTEST= option is specified.

Printing Options

NOPRINT
suppresses the normal printed output but does not suppress error listings. If this option is specified, then any other print option is turned off.

PRINTALL
turns on all the printing options. The options that are set by PRINTALL are COVB and CORRB.

CORRB
prints the correlation matrix of the parameter estimates.

COVB
prints the covariance matrix of the parameter estimates.

Model Estimation Options

COVEST=*covariance-option*
specifies the method for calculating the covariance matrix of parameter estimates. You can specify the following *covariance-options*:

OP	specifies the covariance from the outer product matrix.
HESSIAN	specifies the covariance from the inverse Hessian matrix.
QML	specifies the covariance from the outer product and Hessian matrices (the quasi-maximum likelihood estimates).

The default is COVEST=HESSIAN.

Optimization Control Options

PROC HPQLIM uses the nonlinear optimization (NLO) subsystem to perform nonlinear optimization tasks. You can specify the following *options*:

ABSCONV=*r*

ABSTOL=*r*
specifies an absolute function value convergence criterion by which minimization stops when $f(\theta^{(k)}) \leq r$. 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 by which minimization stops when the function value has a small change in successive iterations:

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

The default value is $r = 0$.

ABSGCONV=*r*

ABSGTOL=*r*
specifies an absolute gradient convergence criterion. Optimization stops when the maximum absolute gradient element is small:

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

The default value is r=1E–5.

ABSXCONV=*r*

ABSXTOL=*r*

specifies an absolute parameter convergence criterion. Optimization stops when the Euclidean distance between successive parameter vectors is small:

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

The default is 0.

FCONV=*r*

FTOL=*r*

specifies a relative function convergence criterion. Optimization stops when a relative change of the function value in successive iterations is small:

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

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

GCONV=*r*

GTOL=*r*

specifies a relative gradient convergence criterion. For all techniques except CONGRA, optimization stops when the normalized predicted function reduction is small:

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

For the CONGRA technique (where a reliable Hessian estimate H is not available), the following criterion is used:

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

The default value is r =1E–8.

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.

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 running time can be much longer than r. The actual running 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.
NONE	specifies that no optimization be performed beyond using the ordinary least squares method to compute the parameter estimates.
NEWRAP	specifies the Newton-Raphson method (the default).
NRRIDG	specifies the Newton-Raphson ridge method.
QUANEW	specifies the quasi-Newton method.
TRUREG	specifies the trust region method.

SINGULAR=*r*

specifies the general singularity criterion that is applied by the HPQLIM procedure in sweeps and inversions. The default for the optimization is 1E–8.

Plotting Options

PLOTS< (*global-plot-options***) > = ***plot-request* | *(plot-requests)*

controls the display of plots. By default, the plots are displayed in panels unless the UNPACK *global-plot-option* is specified. When you specify only one *plot-request*, you can omit the parentheses around it.

Global Plot Options

You can specify the following *global-plot-options*:

ONLY

displays only the requested plot.

UNPACKPANEL

UNPACK

specifies that all paneled plots be unpacked, meaning that each plot in a panel is displayed separately.

Plot Requests

You can specify the following *plot-requests*:

ALL

specifies all types of available plots.

AUTOCORR< (LAGS=*n***) >**

displays the autocorrelation function plots for the parameters. The optional LAGS= suboption specifies the number (up to lag *n*) of autocorrelations to be plotted in the autocorrelation function plot. If this suboption is not specified, autocorrelations are plotted up to lag 50. This *plot-request* is available only for Bayesian analysis.

BAYESDIAG
is equivalent to specifying the TRACE, AUTOCORR, and DENSITY *plot-requests*.

DENSITY< (FRINGE) >
displays the kernel density plots for the parameters. If you specify the FRINGE suboption, a fringe plot is created on the X axis of the kernel density plot. This *plot-request* is available only for Bayesian analysis.

NONE
suppresses all diagnostic plots.

TRACE< (SMOOTH) >
displays the trace plots for the parameters. The SMOOTH suboption displays a fitted penalized B-spline curve for each plot. This *plot-request* is available only for Bayesian analysis.

BAYES Statement

BAYES < *options* > **;**

The BAYES statement controls the Metropolis sampling scheme that is used to obtain samples from the posterior distribution of the underlying model and data.

DIAGNOSTICS=ALL | NONE | *(keyword-list)*
DIAG=ALL | NONE | *(keyword-list)*
controls which diagnostics are produced. All the following diagnostics are produced when you specify DIAGNOSTICS=ALL. If you do not want any of these diagnostics, specify DIAGNOSTICS=NONE. If you want some but not all of the diagnostics, or if you want to change certain settings of these diagnostics, specify one or more of the following keywords. The default is DIAGNOSTICS=NONE.

AUTOCORR < (LAGS=*numeric-list***) >**
computes the autocorrelations at lags that are specified in the *numeric-list*. Elements in the *numeric-list* are truncated to integers, and repeated values are removed. If the LAGS= option is not specified, autocorrelations of lags 1, 5, and 10 are computed.

ESS
computes Carlin's estimate of the effective sample size, the correlation time, and the efficiency of the chain for each parameter.

GEWEKE < (*geweke-options***) >**
computes the Geweke spectral density diagnostics, which are essentially a two-sample t test between the first f_1 portion and the last f_2 portion of the chain. The defaults are $f_1 = 0.1$ and $f_2 = 0.5$, but you can choose other fractions by using the following *geweke-options*:

FRAC1=*value*
specifies the fraction f_1 for the first window.

FRAC2=*value*
specifies the fraction f_2 for the second window.

HEIDELBERGER < (*heidel-options*) >
computes for each variable the Heidelberger and Welch diagnostic, which consists of a stationarity test of the null hypothesis that the sample values form a stationary process. If the stationarity test is not rejected, a halfwidth test is then carried out. Optionally, you can specify one or more of the following *heidel-options*:

EPS=*value*
specifies a positive number ϵ such that if the halfwidth is less than ϵ times the sample mean of the retained iterates, the halfwidth test is passed.

HALPHA=*value*
specifies the α level ($0 < \alpha < 1$) for the halfwidth test.

SALPHA=*value*
specifies the α level ($0 < \alpha < 1$) for the stationarity test.

MCSE

MCERROR
computes the Monte Carlo standard error for each parameter. The Monte Carlo standard error, which measures the simulation accuracy, is the standard error of the posterior mean estimate and is calculated as the posterior standard deviation divided by the square root of the effective sample size.

RAFTERY< (*raftery-options*) >
computes the Raftery and Lewis diagnostics, which evaluate the accuracy of the estimated quantile ($\hat{\theta}_Q$ for a given $Q \in (0,1)$) of a chain. $\hat{\theta}_Q$ can achieve any degree of accuracy when the chain is allowed to run for a long time. The computation stops when the estimated probability $\hat{P}_Q = \Pr(\theta \leq \hat{\theta}_Q)$ reaches within $\pm R$ of the value Q with probability S; that is, $\Pr(Q - R \leq \hat{P}_Q \leq Q + R) = S$. The following *raftery-options* enable you to specify Q, R, S, and a precision level ϵ for the test:

QUANTILE | Q=*value*
specifies the order (a value between 0 and 1) of the quantile of interest. The default is 0.025.

ACCURACY | R=*value*
specifies a small positive number as the margin of error for measuring the accuracy of the estimation of the quantile. The default is 0.005.

PROBABILITY | S=*value*
specifies the probability of attaining the accuracy of the estimation of the quantile. The default is 0.95.

EPSILON | EPS=*value*
specifies the tolerance level (a small positive number) for the stationary test. The default is 0.001.

MINTUNE=*number*
specifies the minimum number of tuning phases. The default is 2.

MAXTUNE=*number*
specifies the maximum number of tuning phases. The default is 24.

NBI=*number*
specifies the number of burn-in iterations before the chains are saved. The default is 1,000.

NMC=*number*
specifies the number of iterations after the burn-in. The default is 1,000.

NTU=*number*
specifies the number of samples for each tuning phase. The default is 500.

OUTPOST=*SAS-data-set*
names the SAS data set to contain the posterior samples. Alternatively, you can create the output data set by specifying an ODS OUTPUT statement as follows:

ODS OUTPUT POSTERIORSAMPLE = < *SAS-data-set* > **;**

PROPCOV=*value*
specifies the method that is used in constructing the initial covariance matrix for the Metropolis-Hastings algorithm. The QUANEW and NMSIMP methods find numerically approximated covariance matrices at the optimum of the posterior density function with respect to all continuous parameters. The tuning phase starts at the optimized values; in some problems, this can greatly increase convergence performance. If the approximated covariance matrix is not positive definite, then an identity matrix is used instead. You can specify the following *values*:

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 quasi-Newton optimization.

TRUREG performs a trust-region optimization.

SAMPLING=MULTIMETROPOLIS | UNIMETROPOLIS
specifies how to sample from the posterior distribution. SAMPLING=MULTIMETROPOLIS implements a Metropolis sampling scheme on a single block that contains all the parameters of the model. SAMPLING=UNIMETROPOLIS implements a Metropolis sampling scheme on multiple blocks, one for each parameter of the model. The default is SAMPLING=MULTIMETROPOLIS.

SEED=*number*
specifies an integer seed in the range 1 to $2^{31} - 1$ for the random number generator in the simulation. Specifying a seed enables you to reproduce identical Markov chains for the same specification. If you do not specify the SEED= option, or if you specify a nonpositive seed, a random seed is derived from the time of day.

STATISTICS < (*global-options*) > = **ALL** | **NONE** | *keyword* | *(keyword-list)*

STATS < (*global-options*) > = **ALL** | **NONE** | *keyword* | *(keyword-list)*

controls the number of posterior statistics that are produced. Specifying STATISTICS=ALL is equivalent to specifying STATISTICS=(CORR COV INTERVAL PRIOR SUMMARY). If you do not want any posterior statistics, specify STATISTICS=NONE. The default is STATISTICS=(SUMMARY INTERVAL). You can specify the following *global-options*:

ALPHA=*value* < *,value* >**...** < *,value* >

controls the probabilities of the credible intervals. The *value*, which must be between 0 and 1, produces a pair of 100(1–*value*)% equal-tail and highest posterior density (HPD) intervals for each parameter. The default is ALPHA=0.05, which yields the 95% credible intervals for each parameter.

PERCENT=*value* < *,value* >**...** < *,value* >

requests the percentile points of the posterior samples. The *value* must be between 0 and 100. The default is PERCENT=25, 50, 75, which yields the 25th, 50th, and 75th percentile points, respectively, for each parameter.

You can specify the following *keywords*:

CORR produces the posterior correlation matrix.

COV produces the posterior covariance matrix.

INTERVAL produces equal-tail credible intervals and HPD intervals. The default is to produce the 95% equal-tail credible intervals and 95% HPD intervals, but you can use the ALPHA= *global-option* to request intervals of any probabilities.

NONE suppresses printing of all summary statistics.

PRIOR produces a summary table of the prior distributions that are used in the Bayesian analysis.

SUMMARY produces the means, standard deviations, and percentile points (25th, 50th, and 75th) for the posterior samples. You can use the PERCENT= *global-option* to request specific percentile points.

THIN=*number*

THINNING=*number*

controls the thinning of the Markov chain. Only one in every k samples is used when THIN=k. If NBI=n_0 and NMC=n, the number of samples that are retained is

$$\left[\frac{n_0 + n}{k}\right] - \left[\frac{n_0}{k}\right]$$

where $[a]$ represents the integer part of the number a. The default is THIN=1.

BOUNDS Statement

BOUNDS *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 HPQLIM 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 HPQLIM procedure, see the section "Naming of Parameters" on page 238. 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 223.

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 with PROC HPQLIM to obtain separate analyses on observations in groups defined by the BY variables.

BY statement processing is not supported when the HPQLIM 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.

ENDOGENOUS Statement

ENDOGENOUS *variables* ~ *options* **;**

The ENDOGENOUS statement specifies the type of dependent variables that appear on the left-hand side of the equation. The listed endogenous variables refer to the dependent variables that appear on the left-hand side of the equation. Currently, no right-hand-side endogeneity is handled in PROC HPQLIM. All variables that appear on the right-hand side of the equation are treated as exogenous.

Discrete Variable Options

DISCRETE < (*discrete-options* **) >**
specifies that the endogenous variables in this statement be discrete. You can specify the following *discrete-options*:

DISTRIBUTION=*distribution-type*

DIST=*distribution-type*

D=*distribution-type*
specifies the cumulative distribution function that is used 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.

ORDER=DATA | FORMATTED | FREQ | INTERNAL
specifies the sort order for the levels of the discrete variables that are specified in the ENDOGENOUS statement. This ordering determines which parameters in the model correspond to each level in the data. You can specify the following sort orders:

DATA sorts levels by order of appearance in the input data set.

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*.

Censored Variable Options

CENSORED (*censored-options* **)**
specifies that the endogenous variables in this statement be censored. 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 in *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 in *variable* is missing, no upper bound is set. By default, no upper bound is set.

Truncated Variable Options

TRUNCATED (*truncated-options* **)**
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 in *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 in *variable* is missing, no upper bound is set. By default, no upper bound is set.

Stochastic Frontier Variable Options

FRONTIER < (*frontier-options* **) >**
You can specify the following *frontier-options*:

TYPE=HALF | EXPONENTIAL | TRUNCATED
specifies the model type.

HALF	specifies half-normal model.
EXPONENTIAL	specifies exponential model.
TRUNCATED	specifies truncated normal model.

PRODUCTION
specifies that the estimated model be a production function.

COST
specifies that the estimated model be a cost function.

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 HPQLIM treats each observation as if it appeared *n* times, where 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 the first FREQ statement is used.

HETERO Statement

HETERO *dependent-variables* $\sim$ *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 HPQLIM supports the following heteroscedastic regression model:

$$y_i = \mathbf{x}_i'\boldsymbol{\beta} + \epsilon_i$$

$$\epsilon_i \sim \mathrm{N}(0, \sigma_i^2)$$

For more information about the specification of functional forms, see the section "Heteroscedasticity" on page 228. The following *options* specify the functional forms of heteroscedasticity:

LINK=EXP | LINEAR
specifies the functional form.

EXP specifies the exponential link function:

$$\sigma_i^2 = \sigma^2(1 + \exp(\mathbf{z}_i'\boldsymbol{\gamma}))$$

LINEAR specifies the linear link function:

$$\sigma_i^2 = \sigma^2(1 + \mathbf{z}_i'\boldsymbol{\gamma})$$

The default is LINK=EXP.

NOCONST
specifies that there be no constant in the linear or exponential heteroscedasticity model:

$$\sigma_i^2 = \sigma^2(\mathbf{z}_i'\boldsymbol{\gamma})$$
$$\sigma_i^2 = \sigma^2\exp(\mathbf{z}_i'\boldsymbol{\gamma})$$

This option is ignored if you do not specify the LINK= option.

SQUARE
estimates the model by using the square of the linear heteroscedasticity function. For example, you can specify the following heteroscedasticity function:

$$\sigma_i^2 = \sigma^2(1 + (\mathbf{z}_i'\boldsymbol{\gamma})^2)$$

```
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 $\mathbf{z}_i'\boldsymbol{\gamma}$ is the same as the exponential of $2\mathbf{z}_i'\boldsymbol{\gamma}$. Hence, the only difference is that all $\boldsymbol{\gamma}$ estimates are divided by two.

This option is ignored if you do not specify the LINK= option. You cannot use the HETERO statement within a Bayesian framework.

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* **=** *regressors* < */ options* > **;**

The MODEL statement specifies the dependent variable and independent regressor variables for the regression model.

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 that are specified in the ENDOGENOUS statement. If an endogenous variable option is specified in both the MODEL statement and the ENDOGENOUS statement, the option in the ENDOGENOUS statement is used.

Discrete Variable Options

DISCRETE <(*discrete-options* **)>**
specifies that the endogenous variables in this statement be discrete. You can specify the following *discrete-options*:

DISTRIBUTION=*distribution-type*

DIST=*distribution-type*

D=*distribution-type*
specifies the cumulative distribution function that is used 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.

ORDER=DATA | FORMATTED | FREQ | INTERNAL
specifies the sort order for the levels of the discrete variables that are specified in the ENDOGENOUS statement. This ordering determines which parameters in the model correspond to each level in the data. You can specify the following sort orders:

DATA sorts levels by order of appearance in the input data set.

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*.

Censored Variable Options

CENSORED <(*censored-options* **)>**
specifies that the endogenous variables in this statement be censored. 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 in *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 in *variable* is missing, no upper bound is set. By default, no upper bound is set.

Truncated Variable Options

TRUNCATED < (*truncated-options* **) >**
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 in *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 in *variable* is missing, no upper bound is set. By default, no upper bound is set.

Stochastic Frontier Variable Options

FRONTIER < (*frontier-options* **) >**
You can specify the following *frontier-options*:

TYPE=HALF | EXPONENTIAL | TRUNCATED
specifies the model type.

HALF	specifies a half-normal model.
EXPONENTIAL	specifies an exponential model.
TRUNCATED	specifies a truncated normal model.

PRODUCTION
specifies that the estimated model be a production function.

COST
specifies that the estimated model be a cost function.

If neither PRODUCTION nor COST is specified, a production function is estimated by default.

OUTPUT Statement

OUTPUT OUT=*SAS-data-set* < *output-options* > **;**

The OUTPUT statement creates a new SAS data set to contain variables that are specified with the COPYVAR option and the following data if they are specified by *output-options*: estimates of $\mathbf{x}'\boldsymbol{\beta}$, 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 the observation, all output estimates except the residual 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*
names the output data set.

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 SAS variables to the output data set.

ERRSTD
outputs estimates of σ_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, binary discrete, and selection endogenous variables.

PREDICTED
outputs estimates of predicted endogenous variables.

PROB
outputs estimates of probability of discrete endogenous variables taking 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 $\mathbf{x}'\boldsymbol{\beta}$.

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).

PERFORMANCE Statement

PERFORMANCE < *performance-options* > **;**

The PERFORMANCE statement specifies *performance-options* to control the multithreaded and distributed computing environment and requests detailed performance results of the HPQLIM procedure. You can also use the PERFORMANCE statement to control whether the HPQLIM procedure executes in single-machine or distributed mode. You can specify the following *performance-options*:

DETAILS
requests a table that shows a timing breakdown of the procedure steps.

NODES=*n*
specifies the number of nodes in the distributed computing environment, provided that the data are not processed alongside the database.

NTHREADS=*n*
specifies the number of threads for analytic computations and overrides the SAS system option THREADS | NOTHREADS. If you do not specify the NTHREADS= option, PROC HPQLIM creates one thread per CPU for the analytic computations.

The PERFORMANCE statement is documented further in the section "PERFORMANCE Statement" on page 31.

PRIOR Statement

PRIOR _REGRESSORS | *parameter-list* $\sim$ *distribution* **;**

The PRIOR statement specifies the prior distribution of the model parameters. You must specify one parameter or a list of parameters, a tilde $\sim$, and then a distribution with its parameters. Multiple PRIOR statements are allowed.

You can specify the following *distributions*:

NORMAL(MEAN=μ**, VAR=**σ^2**)**
specifies a normal distribution with the parameters MEAN and VAR.

GAMMA(SHAPE=*a***, SCALE=***b***)**
specifies a gamma distribution with the parameters SHAPE and SCALE.

IGAMMA(SHAPE=*a***, SCALE=***b***)**
specifies an inverse gamma distribution with the parameters SHAPE and SCALE.

UNIFORM(MIN=*m***, MAX=***M***)**
specifies a uniform distribution that is defined between MIN and MAX.

BETA(SHAPE1=*a***, SHAPE2=***b***, MIN=***m***, MAX=***M***)**
specifies a beta distribution with the parameters SHAPE1 and SHAPE2 and defined between MIN and MAX.

T(LOCATION=μ, DF=ν)
specifies a noncentral t distribution with DF degrees of freedom and a location parameter equal to LOCATION.

For more information about how to specify *distributions*, see the section "Standard Distributions" on page 232.

You can specify the special keyword REGRESSORS to select all the parameters that are used in the linear regression component of the model.

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 substitution ($-$) 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 hpqlim data=one;
   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):

```
proc hpqlim;
   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

$$\beta_1 = 0$$

and

$$0.5\beta_2 + 2\beta_3 = 0$$

Only linear equality restrictions and tests are permitted in PROC HPQLIM. 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 quoted string after the TEST keyword. If you specify both a label before the TEST keyword and a quoted string after the keyword, PROC HPQLIM uses the label that precedes the colon. If no label or quoted string is specified, PROC HPQLIM 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 (/):

NONORMALIZE

specifies that the weights must be used as is. When this option is not specified, 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: HPQLIM Procedure

Ordinal Discrete Choice Modeling

Binary Probit and Logit Model

The binary choice model is

$$y_i^* = \mathbf{x}_i'\boldsymbol{\beta} + \epsilon_i$$

where the value of the latent dependent variable, y_i^*, is observed only as follows:

$$\begin{aligned} y_i &= 1 \quad \text{if } y_i^* > 0 \\ &= 0 \quad \text{otherwise} \end{aligned}$$

The disturbance, ϵ_i, of the probit model has a standard normal distribution with the 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 distribution function (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:

$$P(y_i = 1) = F(\mathbf{x}_i'\boldsymbol{\beta}) = \begin{cases} \Phi(\mathbf{x}_i'\boldsymbol{\beta}) & \text{(probit)} \\ \Lambda(\mathbf{x}_i'\boldsymbol{\beta}) & \text{(logit)} \end{cases}$$

For more information, see the section "Ordinal Discrete Choice Modeling" (Chapter 27, *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

$$y_i^* = \mathbf{x}_i'\boldsymbol{\beta} + \epsilon_i$$

where error disturbances, ϵ_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

$$R_{i,j} = \mu_j - \mathbf{x}_i'\boldsymbol{\beta}$$

The probability that the unobserved dependent variable is contained in the jth 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 the section "Ordinal Discrete Choice Modeling" (Chapter 27, *SAS/ETS User's Guide*).

Limited Dependent Variable Models

Censored Regression Models

When the dependent variable is censored, values in a certain range are all transformed to a single value. For example, the standard Tobit model can be defined as

$$y_i^* = \mathbf{x}_i'\boldsymbol{\beta} + \epsilon_i$$

$$y_i = \begin{cases} y_i^* & \text{if} y_i^* > 0 \\ 0 & \text{if} y_i^* \leq 0 \end{cases}$$

where $\epsilon_i \sim \text{iid} N(0, \sigma^2)$.

The Tobit model can be generalized to handle observation-by-observation censoring. The censored model 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 Chapter 27.7, "Censored Regression Models" (*SAS/ETS User's Guide*).

Truncated Regression Models

In a truncated model, the observed sample is a subset of the population where 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

$$\ell = \sum_{i \in \{y_i > 0\}} \left\{ -\ln \Phi(\mathbf{x}_i'\boldsymbol{\beta}/\sigma) + \ln \left[\frac{\phi((y_i - \mathbf{x}_i'\boldsymbol{\beta})/\sigma)}{\sigma} \right] \right\}$$

For more information, see the section "Truncated Regression Models" (Chapter 27, *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 allow for random shocks of the production or cost but also include 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

$$\ln(y_i) = \beta_0 + \sum_n \boldsymbol{\beta_n} \ln(x_{ni}) + \epsilon_i$$

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 ϵ_i is negatively skewed and represents technical inefficiency. If $u_i < 0$, the error term ϵ_i is positively skewed and represents cost inefficiency. PROC HPQLIM 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

$$f(\epsilon) = \frac{2}{\sigma}\phi\left(\frac{\epsilon}{\sigma}\right)\Phi\left(-\frac{\epsilon\lambda}{\sigma}\right)$$

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 the section "Stochastic Frontier Production and Cost Models" (Chapter 27, *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 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 ϵ for the production function is

$$f(\epsilon) = \left(\frac{1}{\sigma_u}\right)\Phi\left(-\frac{\epsilon}{\sigma_v} - \frac{\sigma_v}{\sigma_u}\right)\exp\left\{\frac{\epsilon}{\sigma_u} + \frac{\sigma_v^2}{2\sigma_u^2}\right\}$$

The marginal density function for the cost function is equal to

$$f(\epsilon) = \left(\frac{1}{\sigma_u}\right) \Phi\left(\frac{\epsilon}{\sigma_v} - \frac{\sigma_v}{\sigma_u}\right) \exp\left\{-\frac{\epsilon}{\sigma_u} + \frac{\sigma_v^2}{2\sigma_u^2}\right\}$$

For more information, see the section "Stochastic Frontier Production and Cost Models" (Chapter 27, *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 zero. Under the normal–truncated normal model, the error term component v_i is iid $N^+(0, \sigma_v^2)$ and u_i is iid $N(\mu, \sigma_u^2)$. The joint density of v_i and u_i can be written as

$$f(u, v) = \frac{1}{\sqrt{2\pi}\sigma_u\sigma_v\Phi\left(\mu/\sigma_u\right)} \exp\left\{-\frac{(u-\mu)^2}{2\sigma_u^2} - \frac{v^2}{2\sigma_v^2}\right\}$$

The marginal density function of ϵ 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 the section "Stochastic Frontier Production and Cost Models" (Chapter 27, *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, (ϵ_i), is heteroscedastic, the variance can be specified as a function of variables

$$E(\epsilon_i^2) = \sigma_i^2 = f(\mathbf{z}_i'\boldsymbol{\gamma})$$

Table 7.2 shows various functional forms of heteroscedasticity and the corresponding options to request each model.

Table 7.2 Specification Summary for Modeling Heteroscedasticity

Number	Model	Options
1	$f(\mathbf{z}_i'\boldsymbol{\gamma}) = \sigma^2(1 + \exp(\mathbf{z}_i'\gamma))$	LINK=EXP (default)
2	$f(\mathbf{z}_i'\boldsymbol{\gamma}) = \sigma^2 \exp(\mathbf{z}_i'\gamma)$	LINK=EXP NOCONST
3	$f(\mathbf{z}_i'\boldsymbol{\gamma}) = \sigma^2(1 + \sum_{l=1}^{L} \gamma_l z_{li})$	LINK=LINEAR
4	$f(\mathbf{z}_i'\boldsymbol{\gamma}) = \sigma^2(1 + (\sum_{l=1}^{L} \gamma_l z_{li})^2)$	LINK=LINEAR SQUARE
5	$f(\mathbf{z}_i'\boldsymbol{\gamma}) = \sigma^2(\sum_{l=1}^{L} \gamma_l z_{li})$	LINK=LINEAR NOCONST
6	$f(\mathbf{z}_i'\boldsymbol{\gamma}) = \sigma^2((\sum_{l=1}^{L} \gamma_l z_{li})^2)$	LINK=LINEAR SQUARE NOCONST

In models 3 and 5, variances of some observations might be negative. Although the HPQLIM 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 zero, 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.

For more information, see the section "Heteroscedasticity and Box-Cox Transformation" (Chapter 27, *SAS/ETS User's Guide*).

Tests on Parameters

In general, the tested hypothesis can be written as

$$H_0 : \mathbf{h}(\theta) = 0$$

where $\mathbf{h}(\theta)$ is an $r \times 1$ vector-valued function of the parameters θ 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 θ and $\tilde{\theta}$ be the constrained estimate of θ such that $h(\tilde{\theta}) = 0$. Let

$$A(\theta) = \partial h(\theta)/\partial\theta \mid_{\hat{\theta}}$$

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

$$LM = \lambda' A(\tilde{\theta}) \tilde{V} A'(\tilde{\theta}) \lambda$$

  where λ 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 θ and L is the concentrated log-likelihood value.

The following statements use the TEST statement to perform a likelihood ratio test:

```
proc hpqlim;
   model y = x1 x2 x3;
   test x1 = 0, x2 * .5 + 2 * x3 = 0 /lr;
run;
```

For more information, see the section "Tests on Parameters" (Chapter 27, *SAS/ETS User's Guide*).

Bayesian Analysis

To perform Bayesian analysis, you must specify a BAYES statement. Unless otherwise stated, all options that are described in this section are options in the BAYES statement.

By default, PROC HPQLIM uses the random walk Metropolis algorithm to obtain posterior samples. For the implementation details of the Metropolis algorithm in PROC HPQLIM, such as the blocking of the parameters and tuning of the covariance matrices, see the sections "Blocking of Parameters" on page 230 and "Tuning the Proposal Distribution" on page 230.

The Bayes theorem states that

$$p(\theta|\mathbf{y}) \propto \pi(\theta)L(y|\theta)$$

where θ is a parameter or a vector of parameters and $\pi(\theta)$ is the product of the prior densities that are specified in the PRIOR statement. The term $L(y|\theta)$ is the likelihood that is associated with the MODEL statement.

Blocking of Parameters

In a multivariate parameter model, all the parameters are updated in one single block (by default or when you specify the SAMPLING=MULTIMETROPOLIS option). This can be inefficient, especially when parameters have vastly different scales. As an alternative, you can update the parameters one at a time (by specifying SAMPLING=UNIMETROPOLIS).

Tuning the Proposal Distribution

One key factor in achieving high efficiency of a Metropolis-based Markov chain is finding a good proposal distribution for each block of parameters. This process is called tuning. The tuning phase consists of a number of loops that are controlled by the options MINTUNE= and MAXTUNE=. The MINTUNE= option controls the minimum number of tuning loops and has a default value of 2. The MAXTUNE= option controls the maximum number of tuning loops and has a default value of 24. Each loop repeats the number of times specified by the NTU= option, which has a default of 500. At the end of every loop, PROC HPQLIM examines the acceptance probability for each block. The acceptance probability is the percentage of NTU

proposed values that have been accepted. If this probability does not fall within the acceptance tolerance range (see the following section), the proposal distribution is modified before the next tuning loop.

A good proposal distribution should resemble the actual posterior distribution of the parameters. Large sample theory states that the posterior distribution of the parameters approaches a multivariate normal distribution (Gelman et al. 2004, Appendix B; Schervish 1995, Section 7.4). That is why a normal proposal distribution often works well in practice. The default proposal distribution in PROC HPQLIM is the normal distribution.

For more information, see Chapter 27.7, "Bayesian Analysis" (*SAS/ETS User's Guide*).

Initial Values of the Markov Chains

You can assign initial values to any parameters. For more information, see the INIT statement. If you use the optimization PROPCOV= option, PROC HPQLIM starts the tuning at the optimized values. This option overwrites the provided initial values.

Prior Distributions

The PRIOR statement specifies the prior distribution of the model parameters. You must specify one parameter or a list of parameters, a tilde ~, and then a distribution with its parameters. You can specify multiple PRIOR statements to define independent priors. Parameters that are associated with a regressor variable are referred to by the name of the corresponding regressor variable.

You can specify the special keyword _REGRESSORS to consider all the regressors of a model. If multiple PRIOR statements affect the same parameter, the last PRIOR statement prevails. For example, in a regression with two regressors (X1, X2), the following statements imply that the prior on X1 is NORMAL(MEAN=0, VAR=1) and the prior on X2 is GAMMA(SHAPE=3, SCALE=4):

```
...
prior _Regressors ~ uniform(min=0, max=1);
prior X1 X2 ~ gamma(shape=3, scale=4);
prior X1 ~ normal(mean=0, var=1);
...
```

If a parameter is not associated with a PRIOR statement or if some of the prior hyperparameters are missing, then the default choices in Table 7.3 are considered.

Table 7.3 Default Values for Prior Distributions

PRIOR Distribution	Hyperparameter$_1$	Hyperparameter$_2$	Min	Max	Parameters Default Choice
NORMAL	MEAN=0	VAR=1E6	$-\infty$	∞	Regression-Location-Threshold
IGAMMA	SHAPE=2.000001	SCALE=1	> 0	∞	Scale
GAMMA	SHAPE=1	SCALE=1	0	∞	
UNIFORM			$-\infty$	∞	
BETA	SHAPE1=1	SHAPE2=1	$-\infty$	∞	
T	LOCATION=0	DF=3	$-\infty$	∞	

For density specification, see the section "Standard Distributions" on page 232.

Standard Distributions

Table 7.4 through Table 7.9 show all the distribution density functions that PROC HPQLIM recognizes. You specify these distribution densities in the PRIOR statement.

Table 7.4 Beta Distribution

PRIOR statement	BETA(SHAPE1=a, SHAPE2=b, MIN=m, MAX=M) Note: Commonly $m = 0$ and $M = 1$.
Density	$\frac{(\theta-m)^{a-1}(M-\theta)^{b-1}}{B(a,b)(M-m)^{a+b-1}}$
Parameter restriction	$a > 0, \quad b > 0, \quad -\infty < m < M < \infty$
Range	$\begin{cases} [m, M] & \text{when } a = 1, b = 1 \\ [m, M) & \text{when } a = 1, b \neq 1 \\ (m, M] & \text{when } a \neq 1, b = 1 \\ (m, M) & \text{otherwise} \end{cases}$
Mean	$\frac{a}{a+b} \times (M - m) + m$
Variance	$\frac{ab}{(a+b)^2(a+b+1)} \times (M - m)^2$
Mode	$\begin{cases} \frac{a-1}{a+b-2} \times M + \frac{b-1}{a+b-2} \times m & a > 1, b > 1 \\ m \text{ and } M & a < 1, b < 1 \\ m & \begin{cases} a < 1, b \geq 1 \\ a = 1, b > 1 \end{cases} \\ M & \begin{cases} a \geq 1, b < 1 \\ a > 1, b = 1 \end{cases} \\ \text{not unique} & a = b = 1 \end{cases}$
Defaults	SHAPE1=SHAPE2=1, MIN $\rightarrow -\infty$, MAX $\rightarrow \infty$

Table 7.5 Gamma Distribution

PRIOR statement	GAMMA(SHAPE=a, SCALE=b)
Density	$\frac{1}{b^a \Gamma(a)} \theta^{a-1} e^{-\theta/b}$
Parameter restriction	$a > 0, b > 0$
Range	$[0, \infty)$
Mean	ab
Variance	ab^2
Mode	$(a - 1)b$
Defaults	SHAPE=SCALE=1

Table 7.6 Inverse Gamma Distribution

PRIOR statement	IGAMMA(SHAPE=a, SCALE=b)
Density	$\frac{b^a}{\Gamma(a)}\theta^{-(a+1)}e^{-b/\theta}$
Parameter restriction	$a > 0, b > 0$
Range	$0 < \theta < \infty$
Mean	$\frac{b}{a-1}, \quad a > 1$
Variance	$\frac{b^2}{(a-1)^2(a-2)}, \quad a > 2$
Mode	$\frac{b}{a+1}$
Defaults	SHAPE=2.000001, SCALE=1

Table 7.7 Normal Distribution

PRIOR statement	NORMAL(MEAN=μ, VAR=σ^2)
Density	$\frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(\theta-\mu)^2}{2\sigma^2}\right)$
Parameter restriction	$\sigma^2 > 0$
Range	$-\infty < \theta < \infty$
Mean	μ
Variance	σ^2
Mode	μ
Defaults	MEAN=0, VAR=1000000

Table 7.8 *t* Distribution

PRIOR statement	T(LOCATION=μ, DF=ν)
Density	$\frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)\sqrt{\pi\nu}}\left[1+\frac{(\theta-\mu)^2}{\nu}\right]^{-\frac{\nu+1}{2}}$
Parameter restriction	$\nu > 0$
Range	$-\infty < \theta < \infty$
Mean	μ, for $\nu > 1$
Variance	$\frac{\nu}{\nu-2}$, for $\nu > 2$
Mode	μ
Defaults	LOCATION=0, DF=3

Table 7.9 Uniform Distribution

PRIOR statement	UNIFORM(MIN=m, MAX=M)
Density	$\frac{1}{M-m}$
Parameter restriction	$-\infty < m < M < \infty$
Range	$\theta \in [m, M]$
Mean	$\frac{m+M}{2}$
Variance	$\frac{(M-m)^2}{12}$
Mode	Not unique
Defaults	MIN$\rightarrow -\infty$, MAX$\rightarrow \infty$

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. 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 σ_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, $\mu_0 = -\infty$ and $\mu_1 = 0$, $\mu_2 = \infty$. For an ordinal response model with M response categories $(\mu_0, \ldots, \mu_M)$, define

$$R_{i,j} = \mu_j - \mathbf{x}_i'\boldsymbol{\beta}$$

The probability that the unobserved dependent variable is contained in the jth category can be written as

$$P[\mu_{j-1} < y_i^* \le \mu_j] = F(R_{i,j}) - F(R_{i,j-1})$$

The marginal effect of changes in the regressors on the probability of $y_i = j$ is then

$$\frac{\partial \text{Prob}[y_i = j]}{\partial \mathbf{x}} = [f(\mu_{j-1} - \mathbf{x}_i'\boldsymbol{\beta}) - f(\mu_j - \mathbf{x}_i'\boldsymbol{\beta})]\boldsymbol{\beta}$$

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

$$\frac{\partial E[y_i|L_i < y_i^* < R_i]}{\partial \mathbf{x}} = \boldsymbol{\beta}\left[1 - \frac{(\phi(a_i) - \phi(b_i))^2}{(\Phi(b_i) - \Phi(a_i))^2} + \frac{a_i\phi(a_i) - b_i\phi(b_i)}{\Phi(b_i) - \Phi(a_i)}\right]$$

where $a_i = \frac{L_i - \mathbf{x}_i'\boldsymbol{\beta}}{\sigma_i}$ and $b_i = \frac{R_i - \mathbf{x}_i'\boldsymbol{\beta}}{\sigma_i}$.

The marginal effects in the censored regression model are

$$\frac{\partial E[y|\mathbf{x}_i]}{\partial \mathbf{x}} = \boldsymbol{\beta} \times \text{Prob}[L_i < y_i^* < R_i]$$

Expected and Conditionally Expected Values

The expected value is the unconditional expectation of the dependent variable. For a censored variable, it is

$$E[y_i] = \Phi(a_i)L_i + (\mathbf{x}_i'\boldsymbol{\beta} + \lambda\sigma_i)(\Phi(b_i) - \Phi(a_i)) + (1 - \Phi(b_i))R_i$$

For a left-censored variable ($R_i = \infty$), this formula is

$$E[y_i] = \Phi(a_i)L_i + (\mathbf{x}_i'\boldsymbol{\beta} + \lambda\sigma_i)(1 - \Phi(a_i))$$

where $\lambda = \frac{\phi(a_i)}{1-\Phi(a_i)}$.

For a right-censored variable ($L_i = -\infty$), this formula is

$$E[y_i] = (\mathbf{x}_i'\boldsymbol{\beta} + \lambda\sigma_i)\Phi(b_i) + (1 - \Phi(b_i))R_i$$

where $\lambda = -\frac{\phi(b_i)}{\Phi(b_i)}$.

For a noncensored variable, this formula is

$$E[y_i] = \mathbf{x}_i'\boldsymbol{\beta}$$

The conditional expected value is the expectation when the variable is inside the boundaries:

$$E[y_i|L_i < y_i < R_i] = \mathbf{x}_i'\boldsymbol{\beta} + \lambda\sigma_i$$

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 227.

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 227. 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 $\sigma_*^2 = \sigma_u^2\sigma_v^2/\sigma^2$. Then, as shown by Jondrow et al. (1982), conditional density is as follows:

$$f(u|\epsilon) = \frac{f(u,\epsilon)}{f(\epsilon)} = \frac{1}{\sqrt{2\pi}\sigma_*} \exp\left\{-\frac{(u-\mu_*)^2}{2\sigma_*^2}\right\} \Big/ \left[1 - \Phi\left(-\frac{\mu_*}{\sigma_*}\right)\right]$$

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

$$TE1_i = E(\exp\{-u_i\}|\epsilon_i) = \left[\frac{1 - \Phi(\sigma_* - \mu_{*i}/\sigma_*)}{1 - \Phi(-\mu_{*i}/\sigma_*)}\right] \exp\left\{-\mu_{*i} + \frac{1}{2}\sigma_*^2\right\}$$

The second version of the estimate (Jondrow et al. 1982) is

$$TE2_i = \exp\{-E(u_i|\epsilon_i)\}$$

where

$$E(u_i|\epsilon_i) = \mu_{*i} + \sigma_*\left[\frac{\phi(-\mu_{*i}/\sigma_*)}{1 - \Phi(-\mu_{*i}/\sigma_*)}\right] = \sigma_*\left[\frac{\phi(\epsilon_i\lambda/\sigma)}{1 - \Phi(\epsilon_i\lambda/\sigma)} - \left(\frac{\epsilon_i\lambda}{\sigma}\right)\right]$$

Normal-Exponential Model

Define $A = -\tilde{\mu}/\sigma_v$ and $\tilde{\mu} = -\epsilon - \sigma_v^2/\sigma_u$. 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

$$TE1_i = E(\exp\{-u_i\}|\epsilon_i) = \left[\frac{1 - \Phi(\sigma_v - \tilde{\mu}_i/\sigma_v)}{1 - \Phi(-\tilde{\mu}_i/\sigma_v)}\right] \exp\left\{-\tilde{\mu}_i + \frac{1}{2}\sigma_v^2\right\}$$

The second version of the estimate is

$$TE2_i = \exp\{-E(u_i|\epsilon_i)\}$$

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 $\sigma_*^2 = \sigma_u^2\sigma_v^2/\sigma^2$. Then, as shown by Kumbhakar and Lovell (2000), conditional density is as follows:

$$f(u|\epsilon) = \frac{1}{\sqrt{2\pi}\sigma_*[1 - \Phi(-\tilde{\mu}/\sigma_*)]} \exp\left\{ -\frac{(u - \tilde{\mu})^2}{2\sigma_*^2} \right\}$$

Hence, $f(u|\epsilon)$ is the density for $N^+(\tilde{\mu}, \sigma_*^2)$.

From this result, it follows that the estimate of technical efficiency is

$$TE1_i = E(\exp\{-u_i\}|\epsilon_i) = \frac{1 - \Phi(\sigma_* - \tilde{\mu}_i/\sigma_*)}{1 - \Phi(-\tilde{\mu}_i/\sigma_*)} \exp\left\{ -\tilde{\mu}_i + \frac{1}{2}\sigma_*^2 \right\}$$

The second version of the estimate is

$$TE2_i = \exp\{-E(u_i|\epsilon_i)\}$$

where

$$E(u_i|\epsilon_i) = \tilde{\mu}_i + \sigma_* \left[\frac{\phi(\tilde{\mu}_i/\sigma_*)}{1 - \Phi(-\tilde{\mu}_i/\sigma_*)} \right]$$

OUTEST= Data Set

The OUTEST= data set contains all the parameters that are estimated by a MODEL statement. Each parameter contains the estimate for the corresponding parameter in the corresponding model. In addition, the OUTEST= data set contains the following variables:

NAME	indicates the name of the independent variable.
TYPE	indicates the type of observation. PARM indicates the row of coefficients; STD indicates the row of standard deviations of the corresponding coefficients.
STATUS	indicates the convergence status for optimization.

The rest of the columns correspond to the explanatory variables.

The OUTEST= data set contains one observation for the MODEL statement, which shows the parameter estimates for that model. If you specify the COVOUT option in the PROC HPQLIM statement, the OUTEST= data set includes additional observations for the MODEL statement, which show the rows of the covariance matrix of parameter estimates. For covariance observations, the value of the _TYPE_ variable is COV, and the _NAME_ variable identifies the parameter that is associated with that row of the covariance matrix. If you specify the CORROUT option in the PROC HPQLIM statement, the OUTEST= data set includes additional observations for the MODEL statement, which show the rows of the correlation matrix of parameter estimates. For correlation observations, the value of the _TYPE_ variable is CORR, and the _NAME_ variable identifies the parameter that is associated with that row of the correlation matrix.

Naming

Naming of Parameters

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. If the HETERO statement is included, 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 7.10 shows the *options* in the OUTPUT statement, with the corresponding variable names and their explanations.

Table 7.10 OUTPUT Statement Options

output-option	**Variable Name**	**Explanation**
CONDITIONAL	CEXPCT_y	Conditional expected value of *y*, conditioned on the truncation
ERRSTD	ERRSTD_y	Standard deviation of error term
EXPECTED	EXPCT_y	Unconditional expected value of *y*
MARGINAL	MEFF_x	Marginal effect of *x* on *y* ($\frac{\partial y}{\partial x}$) with single equation
PREDICTED	P_y	Predicted value of *y*
RESIDUAL	RESID_y	Residual of *y*, (y – PredictedY)
PROB	PROB_y	Probability that *y* is taking the observed value in this observation (discrete *y* only)
PROBALL	PROB*i*_y	Probability that *y* is taking the *i*th value (discrete *y* only)
MILLS	MILLS_y	Inverse Mills ratio for *y*
TE1	TE1	Technical efficiency estimate for each producer proposed by Battese and Coelli (1988)
TE2	TE2	Technical efficiency estimate for each producer proposed by Jondrow et al. (1982)
XBETA	XBETA_y	Structure part ($\mathbf{x}'\boldsymbol{\beta}$) of *y* equation

If you prefer to name the output variables differently, you can use the RENAME option in the data set. For example, the following statements rename the residual of *y* as Resid:

```
proc hpqlim data=one;
   model y = x1-x10 / censored;
   output out=outds(rename=(resid_y=resid)) residual;
run;
```

ODS Table Names

PROC HPQLIM assigns a name to each table that it creates. You can use these names to refer to the table when you use the Output Delivery System (ODS) to select tables and create output data sets. These names are listed in Table 7.11.

Table 7.11 ODS Tables Produced in PROC HPQLIM

ODS Table Name	Description	Option
ODS Tables Created by the MODEL Statement and TEST Statement		
ResponseProfile	Response profile	Default
FitSummary	Summary of nonlinear estimation	Default
ParameterEstimates	Parameter estimates	Default
SummaryContResponse	Summary of continuous response	Default
CovB	Covariance of parameter estimates	COVB
CorrB	Correlation of parameter estimates	CORRB
ODS Tables Created by the BAYES Statement		
AutoCorr	Autocorrelation statistics for each parameter	Default
Corr	Correlation matrix of the posterior samples	STATS=COR
Cov	Covariance matrix of the posterior samples	STATS=COV
ESS	Effective sample size for each parameter	Default
MCSE	Monte Carlo standard error for each parameter	Default
Geweke	Geweke diagnostics for each parameter	Default
Heidelberger	Heidelberger-Welch diagnostics for each parameter	DIAGNOSTICS=HEIDEL
PostIntervals	Equal-tail and HPD intervals for each parameter	Default
PosteriorSample	Posterior samples	(ODS output data set only)
PostSummaries	Posterior summaries	Default
PriorSummaries	Prior summaries	STATS=PRIOR
Raftery	Raftery-Lewis diagnostics for each parameter	DIAGNOSTICS=RAFTERY
ODS Tables Created by the TEST Statement		
TestResults	Test results	Default

ODS Graphics

You can use a name to reference every graph that is produced through ODS Graphics. The names of the graphs that PROC HPQLIM generates are listed in Table 7.12.

Table 7.12 Graphs Produced by PROC HPQLIM When a BAYES Statement Is Included

ODS Graph Name	Plot Description	Statement and Option
Bayesian Diagnostic Plots		
ADPanel	Autocorrelation function and density panel	PLOTS=(AUTOCORR DENSITY)
AutocorrPanel	Autocorrelation function panel	PLOTS=AUTOCORR
AutocorrPlot	Autocorrelation function plot	PLOTS(UNPACK)=AUTOCORR
DensityPanel	Density panel	PLOTS=DENSITY
DensityPlot	Density plot	PLOTS(UNPACK)=DENSITY
TAPanel	Trace and autocorrelation function panel	PLOTS=(TRACE AUTOCORR)
TADPanel	Trace, density, and autocorrelation function panel	PLOTS=(TRACE AUTOCORR DENSITY)
TDPanel	Trace and density panel	PLOTS=(TRACE DENSITY)
TracePanel	Trace panel	PLOTS=TRACE
TracePlot	Trace plot	PLOTS(UNPACK)=TRACE

Examples: The HPQLIM Procedure

Example 7.1: High-Performance Model with Censoring

This example shows the use of the HPQLIM procedure with an emphasis on processing a large data set and on the performance improvements that are achieved by executing in the high-performance distributed environment.

The following DATA step generates 5 million replicates from a censored model. The model contains seven variables:

```
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. The model is executed in the distributed computing environment with two threads and only one node. These settings are used to obtain a hypothetical environment that might resemble running the HPQLIM procedure on a desktop workstation with a dual-core CPU. To run these statements successfully, you need to set the macro variables GRIDHOST and GRIDINSTALLLOC to resolve to appropriate values, or you can replace the references to the macro variables in the example with the appropriate values.

```
option set=GRIDHOST="&GRIDHOST";
option set=GRIDINSTALLLOC="&GRIDINSTALLLOC";

title1 'Estimating a Censored Model';

proc hpqlim data=simulate ;
   performance nthreads=2 nodes=1 details
               host="&GRIDHOST" install="&GRIDINSTALLLOC";
   model y=x1-x7 /censored(lb=0 ub=400);
run;
```

Output 7.1.1 shows that the censored model was estimated on the grid, defined in a macro variable named GRIDHOST, in a distributed environment on only one node with two threads.

Output 7.1.1 Censored Model with One Node and Two Threads: Performance Table

Estimating a Censored Model

Performance Information	
Host Node	<< your grid host >>
Install Location	<< your grid install location >>
Execution Mode	Distributed
Number of Compute Nodes	1
Number of Threads per Node	2

Output 7.1.2 shows the estimation results for the censored model. The "Model Fit Summary" table shows detailed information about the model and indicates that all 5 million observations were used to fit 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. The optimization of the model with 5 million observations took 58.03 seconds.

Output 7.1.2 Censored Model with One Node and Two Threads: Summary

Model Information	
Data Source	SIMULATE
Response Variable	y
Optimization Technique	Quasi-Newton

Number of Observations	
Number of Observations Read	5000000
Number of Observations Used	5000000

Summary Statistics of Continuous Responses

Variable	Mean	Standard Error	Type	Lower Bound	Upper Bound	N Obs Lower Bound	N Obs Upper Bound
y	127.0	159.491090	Censored	0	400.0	249E4	8E5

Convergence criterion (FCONV=2.220446E-16) satisfied.

Model Fit Summary	
Number of Endogenous Variables	1
Endogenous Variable	y
Number of Observations	5000000
Log Likelihood	-15268972
Maximum Absolute Gradient	0.0008332
Number of Iterations	10
Optimization Method	Quasi-Newton
AIC	30537962
Schwarz Criterion	30538083

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	2.220358	0.222201	9.99	<.0001
x1	1	3.055491	0.201620	15.15	<.0001
x2	1	4.000196	0.201570	19.85	<.0001
x3	1	1.852735	0.201555	9.19	<.0001
x4	1	4.170323	0.201533	20.69	<.0001
x5	1	-3.010670	0.201458	-14.94	<.0001
x6	1	-5.176019	0.201541	-25.68	<.0001
x7	1	-2.695886	0.201671	-13.37	<.0001
_Sigma	1	399.997846	0.261930	1527.12	<.0001

Procedure Task Timing

Task	Seconds	Percent
Reading and Levelizing Data	1.75	2.92%
Communication to Client	0.19	0.32%
Optimization	58.03	96.76%
Post-optimization	0.00	0.00%

In the following statements, the PERFORMANCE statement is modified to use a grid with 10 nodes, with each node capable of spawning eight threads:

```
proc hpqlim data=simulate ;
   performance nthreads=8 nodes=10 details
               host="&GRIDHOST" install="&GRIDINSTALLLOC";
   model y=x1-x7 /censored(lb=0 ub=400);
run;
```

The second model which was run on a grid with 10 nodes and eight threads each (Output 7.1.3) took only 1.5 seconds instead of 57.6 seconds to optimize.

Output 7.1.3 Censored Model on Ten Nodes with Eight Threads Each: Performance Table

Estimating a Censored Model

Performance Information	
Host Node	<< your grid host >>
Install Location	<< your grid install location >>
Execution Mode	Distributed
Number of Compute Nodes	10
Number of Threads per Node	8

Because the two models being estimated are identical, it is reasonable to expect that Output 7.1.2 and Output 7.1.4 would show the same results except for the performance. However, in certain circumstances, you might observe slight numerical differences in the results (depending on the number of nodes and threads) because the order in which partial results are accumulated, the limits of numerical precision, and the propagation of error in numerical computations can make a difference in the final result.

Output 7.1.4 Censored Model on Ten Nodes with Eight Threads Each: Summary

Model Information	
Data Source	SIMULATE
Response Variable	y
Optimization Technique	Quasi-Newton

Number of Observations	
Number of Observations Read	5000000
Number of Observations Used	5000000

Summary Statistics of Continuous Responses

Variable	Mean	Standard Error	Type	Lower Bound	Upper Bound	N Obs Lower Bound	N Obs Upper Bound
y	127.0	159.491090	Censored	0	400.0	249E4	8E5

Convergence criterion (FCONV=2.220446E-16) satisfied.

Output 7.1.4 *continued*

Model Fit Summary	
Number of Endogenous Variables	1
Endogenous Variable	y
Number of Observations	5000000
Log Likelihood	-15268972
Maximum Absolute Gradient	0.0008332
Number of Iterations	10
Optimization Method	Quasi-Newton
AIC	30537962
Schwarz Criterion	30538083

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	2.220358	0.222201	9.99	<.0001
x1	1	3.055491	0.201620	15.15	<.0001
x2	1	4.000196	0.201570	19.85	<.0001
x3	1	1.852735	0.201555	9.19	<.0001
x4	1	4.170323	0.201533	20.69	<.0001
x5	1	-3.010670	0.201458	-14.94	<.0001
x6	1	-5.176019	0.201541	-25.68	<.0001
x7	1	-2.695886	0.201671	-13.37	<.0001
_Sigma	1	399.997846	0.261930	1527.12	<.0001

Procedure Task Timing

Task	Seconds	Percent
Reading and Levelizing Data	0.10	5.57%
Communication to Client	0.11	6.27%
Optimization	1.59	88.16%
Post-optimization	0.00	0.00%

As this example suggests, increasing the number of nodes and the number of threads per node improves performance significantly. When you use the parallelism that a high-performance distributed environment affords, you can see an even more dramatic reduction in the time required for the optimization as the number of observations in the data set increases. When the data set is extremely large, the computations might not even be possible with the typical memory resources and computational constraints of a desktop computer. Under such circumstances the high-performance distributed environment becomes a necessity.

Example 7.2: Bayesian High-Performance Model with Censoring

This example shows the use of the Bayesian analysis available in the HPQLIM procedure with an emphasis on processing a large data set and on the performance improvements that are achieved by executing in a high-performance distributed environment.

The model and the data set are the same as in Example 7.1, and the priors are set to the defaults.

The model is executed in the distributed computing environment with 10 nodes, where each node spawns eight threads. To run the following statements successfully, you need to set the macro variables GRIDHOST and GRIDINSTALLLOC to resolve to appropriate values, or you can replace the references to the macro variables in the example with the appropriate values:

```
option set=GRIDHOST="&GRIDHOST";
option set=GRIDINSTALLLOC="&GRIDINSTALLLOC";

title1 'Bayesian Estimation of a Censored Model';

proc hpqlim data=simulate ;
bayes nbi=10000 nmc=30000;
   performance nthreads=8 nodes=10 details
               host="&GRIDHOST" install="&GRIDINSTALLLOC";
   model y=x1-x7 /censored(lb=0 ub=400);
   %*;     ods output PerformanceInfo=perfInfo;
   %*;     ods output Timing=time;
run;
```

Output 7.2.1 shows a summary of the posterior distribution that is associated with the censored model when you use diffuse prior distributions.

Output 7.2.1 Posterior Summary for Bayesian Censored Model

Bayesian Estimation of a Censored Model

The HPQLIM Procedure

Posterior Summaries

Parameter	N	Mean	Standard Deviation	Percentiles 25%	50%	75%
Intercept	30000	2.2235	0.2240	2.0763	2.2181	2.3842
x1	30000	3.0638	0.2050	2.9276	3.0630	3.1993
x2	30000	3.9996	0.2038	3.8594	4.0050	4.1365
x3	30000	1.8587	0.2031	1.7246	1.8596	1.9912
x4	30000	4.1705	0.1982	4.0348	4.1656	4.3093
x5	30000	-3.0118	0.1976	-3.1483	-3.0087	-2.8781
x6	30000	-5.1700	0.2023	-5.3096	-5.1695	-5.0342
x7	30000	-2.6907	0.2066	-2.8308	-2.6875	-2.5465
_Sigma	30000	400.0	0.2668	399.8	400.0	400.2

Output 7.2.2 show a summary of the performance when you use a distributed computing environment with 10 nodes, where each node spawns eight threads.

Output 7.2.2 Performance Analysis for Bayesian Censored Model on Ten Nodes with Eight Threads Each

Bayesian Estimation of a Censored Model

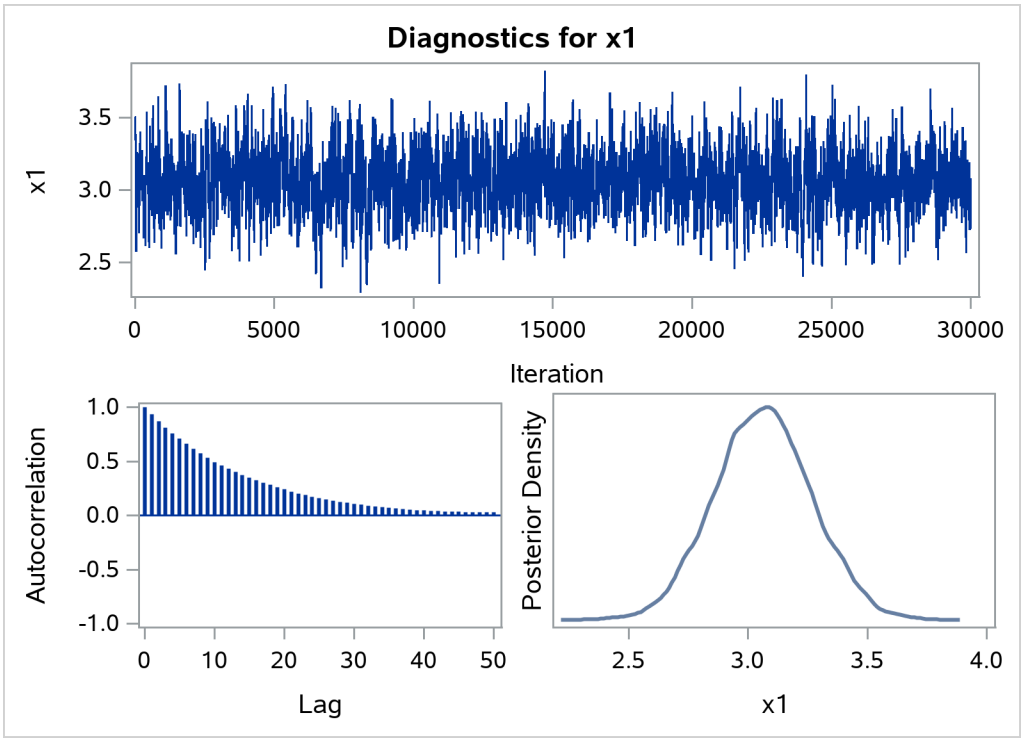
Performance Information	
Host Node	<< your grid host >>
Install Location	<< your grid install location >>
Execution Mode	Distributed
Number of Compute Nodes	10
Number of Threads per Node	8

Bayesian Estimation of a Censored Model

Procedure Task Timing		
Task	**Seconds**	**Percent**
Reading and Levelizing Data	0.09	0.01%
Communication to Client	0.16	0.01%
Bayesian Analysis: Likelihood for MCMC	1282.67	99.84%
Bayesian Analysis: MCMC	0.32	0.03%
Optimization	1.52	0.12%
Post-optimization	0.00	0.00%

Finally, Output 7.2.3 shows the diagnostic and summary plots that are associated with X1.

Output 7.2.3 Bayesian Diagnostic and Summary Plots for x1



The implementation took only 21 minutes to sample from the posterior distribution. The same implementation in a hypothetical environment resembling a desktop workstation with a dual-core CPU would have taken approximately 12 hours.

References

Aigner, C., Lovell, C. A. K., and Schmidt, P. (1977). "Formulation and Estimation of Stochastic Frontier Production Function Models." *Journal of Econometrics* 6:21–37.

Battese, G. E., and Coelli, T. J. (1988). "Prediction of Firm-Level Technical Efficiencies with a Generalized Frontier Production Function and Panel Data." *Journal of Econometrics* 38:387–399.

Christensen, L. R., and Greene, W. H. (1976). "Economies of Scale in U.S. Electric Power Generation." *Journal of Political Economy* 84:655–676.

Coelli, T. J., Prasada Rao, D. S., and Battese, G. E. (1998). *An Introduction to Efficiency and Productivity Analysis*. London: Kluwer Academic.

Gelman, A., Carlin, J. B., Stern, H. S., and Rubin, D. B. (2004). *Bayesian Data Analysis*. 2nd ed. London: Chapman & Hall.

Jondrow, J., Lovell, C. A. K., Materov, I. S., and Schmidt, P. (1982). "On the Estimation of Technical Efficiency in the Stochastic Frontier Production Function Model." *Journal of Econometrics* 19:233–238.

Kumbhakar, S. C., and Lovell, C. A. K. (2000). *Stochastic Frontier Analysis*. New York: Cambridge University Press.

McKelvey, R. D., and Zavoina, W. (1975). "A Statistical Model for the Analysis of Ordinal Level Dependent Variables." *Journal of Mathematical Sociology* 4:103–120.

Meeusen, W., and van den Broeck, J. (1977). "Efficiency Estimation from Cobb-Douglas Production Functions with Composed Error." *International Economic Review* 18:435–444.

Mroz, T. A. (1987). "The Sensitivity of an Empirical Model of Married Women's Work to Economic and Statistical Assumptions." *Econometrica* 55:765–799.

Schervish, M. J. (1995). *Theory of Statistics*. New York: Springer-Verlag.

Wooldridge, J. M. (2002). *Econometric Analysis of Cross Section and Panel Data*. Cambridge, MA: MIT Press.

Chapter 8
The HPSEVERITY Procedure

Contents

Overview: HPSEVERITY Procedure

The HPSEVERITY procedure estimates parameters of any arbitrary continuous probability distribution that is used to model the magnitude (severity) of a continuous-valued event of interest. Some examples of such events are loss amounts paid by an insurance company and demand of a product as depicted by its sales. PROC HPSEVERITY 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 HPSEVERITY runs in either single-machine mode or distributed mode. **NOTE:** Distributed mode requires SAS High-Performance Econometrics.

PROC HPSEVERITY provides a default set of probability distribution models that includes the Burr, exponential, gamma, generalized Pareto, inverse Gaussian (Wald), lognormal, Pareto (Type II), 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 SAS data set. You can optionally group the values by a set of BY variables. PROC HPSEVERITY 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 HPSEVERITY 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 the procedure to output the status of the estimation process, the parameter estimates and their standard errors, the estimated covariance structure of the parameters, the statistics of fit, estimated cumulative distribution function (CDF) for each of the specified distributions, and the empirical distribution function (EDF) estimate (which is used to compute the KS, AD, and CvM statistics of fit).

The following key features make PROC HPSEVERITY unique among SAS procedures that can estimate continuous probability distributions:

- It enables you to fit a distribution model when the severity values are truncated or 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 HPSEVERITY 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), whereas the latter is used by default when you specify both left-censoring and right-censoring effects. The procedure computes the standard errors for all EDF estimators.

- 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 HPSEVERITY 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.

  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 construct 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.

- It enables you to specify your own objective function to be optimized for estimating the parameters of a model. You can write SAS programming statements to specify the contribution of each observation to the objective function. You can use keyword functions such as _PDF_ and _CDF_ to generalize the objective function to any distribution. If you do not specify your own objective function, then the parameters of a model are estimated by maximizing the likelihood function of the data.

- It enables you to create scoring functions that offer a convenient way to evaluate any distribution function, such as PDF, CDF, QUANTILE, or your custom distribution function, for a fitted model on new observations.

Because the HPSEVERITY procedure is a high-performance analytical procedure, it also does the following:

- enables you to run in distributed mode on a cluster of machines that distribute the data and the computations
- enables you to run in single-machine mode on the server where SAS is installed
- exploits all the available cores and concurrent threads, regardless of execution mode

For more information, see the section "Processing Modes" on page 6.

Getting Started: HPSEVERITY Procedure

This section outlines the use of the HPSEVERITY 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 HPSEVERITY 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 Φ 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;
```

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 hpseverity data=test_sev1 crit=aicc;
   loss y;
   dist _predefined_;
run;
```

The PROC HPSEVERITY statement specifies the input data set 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 8.1 through Figure 8.3. First, information about the input data set is displayed followed by the "Model Selection" table, as shown in Figure 8.1. 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 given distribution model has converged, might have converged, or failed. The Selected column indicates whether a given 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 8.1 Data Set Information and Model Selection Table

The HPSEVERITY Procedure

Input Data Set	
Name	WORK.TEST_SEV1
Label	Simple Lognormal Sample

Model Selection			
Distribution	**Converged**	**AICC**	**Selected**
Burr	Yes	322.50845	No
Exp	Yes	508.12287	No
Gamma	Yes	320.50264	No
Igauss	Yes	319.61652	No
Logn	Yes	319.56579	Yes
Pareto	Yes	510.28172	No
Gpd	Yes	510.20576	No
Weibull	Yes	334.82373	No

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 8.2. 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 HPSEVERITY 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 8.2 Estimation Details for the Lognormal Model

The HPSEVERITY Procedure
Logn Distribution

Distribution Information	
Name	Logn
Description	Lognormal Distribution
Distribution Parameters	2

Figure 8.2 *continued*

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

Optimization Summary	
Optimization Technique	Trust Region
Iterations	2
Function Calls	8
Log Likelihood	-157.72104

Fit Statistics	
-2 Log Likelihood	315.44208
AIC	319.44208
AICC	319.56579
BIC	324.65242
Kolmogorov-Smirnov	0.50641
Anderson-Darling	0.31240
Cramer-von Mises	0.04353

Parameter Estimates

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

The parameter estimates of the Burr distribution are shown in Figure 8.3. These estimates are used in the next example.

Figure 8.3 Parameter Estimates for the Burr Model

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
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 HPSEVERITY enables you to specify that the response variable values are left-truncated or right-censored. The following DATA step expands the data set 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 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, THRESHOLD specifies the left-truncation of Y. 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.

```
/*----- 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);

   /* make about 20% of the observations left-truncated */
   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 statements use the AICC criterion to analyze which of the four predefined distributions (lognormal, Burr, gamma, and Weibull) has the best fit for the data:

```
proc hpseverity data=test_sev2 crit=aicc print=all ;
   loss y / lt=threshold rc=limit;

   dist logn burr gamma weibull;
   performance nthreads=2;
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 HPSEVERITY statement requests that all the displayed output be prepared. The NTHREADS option in the PERFORMANCE statement specifies that two threads of computation be used. The option is shown here just for illustration. You should use it only when you want to restrict the procedure to use a different number of threads than the value of the CPUCOUNT= system option, which usually defaults to the number of physical CPU cores available on your machine, thereby allowing the procedure to fully utilize the computational power of your machine.

Some of the key results prepared by PROC HPSEVERITY are shown in Figure 8.4 through Figure 8.7. In addition to the estimates of the range, mean, and standard deviation of Y, the "Descriptive Statistics for y" table shown in Figure 8.4 also indicates the number of observations that are left-truncated or right-censored. The "Model Selection" table in Figure 8.4 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 8.4 Summary Results for the Truncated and Censored Data

The HPSEVERITY Procedure

Input Data Set	
Name	WORK.TEST_SEV2
Label	A Lognormal Sample With Censoring and Truncation

Figure 8.4 *continued*

Descriptive Statistics for y	
Observations	100
Observations Used for Estimation	100
Minimum	2.30264
Maximum	8.34116
Mean	4.62007
Standard Deviation	1.23627
Left Truncated Observations	23
Right Censored Observations	14

Model Selection			
Distribution	**Converged**	**AICC**	**Selected**
Logn	Yes	298.92672	Yes
Burr	Yes	302.66229	No
Gamma	Yes	299.45293	No
Weibull	Yes	309.26779	No

PROC HPSEVERITY 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 8.5. It indicates that the lognormal model is chosen by all the criteria.

Figure 8.5 Comparing All Statistics of Fit for the Truncated and Censored Data

All Fit Statistics													
Distribution	**-2 Log Likelihood**		**AIC**		**AICC**		**BIC**		**KS**		**AD**		**CvM**	
Logn	294.80301	*	298.80301	*	298.92672	*	304.01335	*	0.51824	*	0.34736	*	0.05159	*
Burr	296.41229		302.41229		302.66229		310.22780		0.66984		0.36712		0.05726	
Gamma	295.32921		299.32921		299.45293		304.53955		0.62511		0.42921		0.05526	
Weibull	305.14408		309.14408		309.26779		314.35442		0.93307		1.40699		0.17465	

Note: The asterisk (*) marks the best model according to each column's criterion.

Specifying Initial Values for Parameters

All the predefined distributions have parameter initialization functions built into them. For the current example, Figure 8.6 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 8.6 Burr Model Summary for the Truncated and Censored Data

Initial Parameter Values and Bounds			
Parameter	**Initial Value**	**Lower Bound**	**Upper Bound**
Theta	4.78102	1.05367E-8	Infty
Alpha	2.00000	1.05367E-8	Infty
Gamma	2.00000	1.05367E-8	Infty

Figure 8.6 *continued*

Optimization Summary	
Optimization Technique	Trust Region
Iterations	8
Function Calls	23
Log Likelihood	-148.20614

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Theta	1	4.76980	0.62492	7.63	<.0001
Alpha	1	1.16363	0.58859	1.98	0.0509
Gamma	1	5.94081	1.05004	5.66	<.0001

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 8.3). One of the ways in which you can specify the initial values is as follows:

```
/*------ Specifying initial values using INIT= option -------*/
proc hpseverity data=test_sev2 crit=aicc print=all;
   loss y / lt=threshold rc=limit;

   dist burr(init=(theta=4.62348 alpha=1.15706 gamma=6.41227));
   performance nthreads=2;
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 8.7. 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.

Figure 8.7 Burr Model Optimization Summary for the Truncated and Censored Data

The HPSEVERITY Procedure
Burr Distribution

Optimization Summary	
Optimization Technique	Trust Region
Iterations	5
Function Calls	16
Log Likelihood	-148.20614

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Theta	1	4.76980	0.62492	7.63	<.0001
Alpha	1	1.16363	0.58859	1.98	0.0509
Gamma	1	5.94081	1.05004	5.66	<.0001

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 HPSEVERITY 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

$$Y \sim \exp(\sum_{j=1}^{k} \beta_j x_j) \cdot \mathcal{F}(\Theta)$$

where $\mathcal{F}$ denotes the distribution of Y with parameters Θ 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 θ denotes the scale parameter and Ω denotes the set of nonscale parameters. The scale θ is affected by the regressors as

$$\theta = \theta_0 \cdot \exp(\sum_{j=1}^{k} \beta_j x_j)$$

where θ_0 denotes a *base* value of the scale parameter.

Given this form of the model, PROC HPSEVERITY 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 μ is a log-transformed scale parameter. This can be verified by replacing μ with a parameter $\theta = e^{\mu}$, which results in the following expressions for the PDF f and the CDF F in terms of θ and σ, respectively, where Φ denotes the CDF of the standard normal distribution:

$$f(x; \theta, \sigma) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\log(x)-\log(\theta)}{\sigma}\right)^2} \quad \text{and} \quad F(x; \theta, \sigma) = \Phi\left(\frac{\log(x) - \log(\theta)}{\sigma}\right)$$

With this parameterization, the PDF satisfies the $f(x; \theta, \sigma) = \frac{1}{\theta} f(\frac{x}{\theta}; 1, \sigma)$ condition and the CDF satisfies the $F(x; \theta, \sigma) = F(\frac{x}{\theta}; 1, \sigma)$ condition. This makes θ 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:

$$\mu = \log(\theta) = 1 + 0.75\ \text{X1} - \text{X2} + 0.25\ \text{X3}$$

```
/*----------- 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 PROC HPSEVERITY step fits the lognormal, Burr, and gamma distribution models to these data. The regressors are specified in the SCALEMODEL statement.

```
proc hpseverity data=test_sev3 crit=aicc print=all;
   loss y;
   scalemodel x1-x3;

   dist logn burr gamma;
run;
```

Some of the key results prepared by PROC HPSEVERITY are shown in Figure 8.8 through Figure 8.12. The descriptive statistics of all the variables are shown in Figure 8.8.

Figure 8.8 Summary Results for the Regression Example

The HPSEVERITY Procedure

Input Data Set	
Name	WORK.TEST_SEV3
Label	A Lognormal Sample Affected by Regressors

Descriptive Statistics for y	
Observations	100
Observations Used for Estimation	100
Minimum	1.17863
Maximum	6.65269
Mean	2.99859
Standard Deviation	1.12845

Figure 8.8 *continued*

Descriptive Statistics for Regressors

Variable	N	Minimum	Maximum	Mean	Standard Deviation
x1	100	0.0005115	0.97971	0.51689	0.28206
x2	100	0.01883	0.99937	0.47345	0.28885
x3	100	0.00255	0.97558	0.48301	0.29709

The comparison of the fit statistics of all the models is shown in Figure 8.9. 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.

Figure 8.9 Comparison of Statistics of Fit for the Regression Example

All Fit Statistics

Distribution	-2 Log Likelihood		AIC		AICC		BIC		KS		AD		CvM	
Logn	187.49609	*	197.49609	*	198.13439	*	210.52194	*	1.97544		17.24618		1.21665	
Burr	190.69154		202.69154		203.59476		218.32256		2.09334		13.93436	*	1.28529	
Gamma	188.91483		198.91483		199.55313		211.94069		1.94472	*	15.84787		1.17617	*

Note: The asterisk (*) marks the best model according to each column's criterion.

The distribution information and the convergence results of the lognormal model are shown in Figure 8.10. 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 8.10 Convergence Results for the Lognormal Model with Regressors

The HPSEVERITY Procedure
Logn Distribution

Distribution Information

Name	Logn
Description	Lognormal Distribution
Distribution Parameters	2
Regression Parameters	3

Convergence Status

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

Optimization Iteration History

Iter	Function Calls	-Log Likelihood	Change	Maximum Gradient
0	2	93.75285		6.16002
1	4	93.74805	-0.0048055	0.11031
2	6	93.74805	-1.5017E-6	0.00003376
3	10	93.74805	-1.705E-13	3.1122E-12

Figure 8.10 *continued*

Optimization Summary	
Optimization Technique	Trust Region
Iterations	3
Function Calls	10
Log Likelihood	-93.74805

The final parameter estimates of the lognormal model are shown in Figure 8.11. 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 μ. 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 μ 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 8.11 Parameter Estimates for the Lognormal Model with Regressors

Parameter Estimates					
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 8.12. The estimate that is reported for the parameter *Theta* is the base value for the scale parameter θ. If the gamma distribution is chosen to model Y, then the effective value of the scale parameter is $\theta = 0.14293 \exp(0.64562\, x_1 - 0.89831\, x_2 + 0.14901\, x_3)$.

Figure 8.12 Parameter Estimates for the Gamma Model with Regressors

Parameter Estimates					
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: HPSEVERITY Procedure

The following statements are available in the HPSEVERITY procedure:

PROC HPSEVERITY *options* ;
BY *variable-list* ;
LOSS < *response-variable* > < / *censoring-truncation-options* > ;
WEIGHT *weight-variable* ;
CLASS *variable* < *(options)* > . . . < *variable* < *(options)* > > < / *global-options* > ;
SCALEMODEL *regression-effect-list* < / *scalemodel-options* > ;
DIST *distribution-name-or-keyword* < *(distribution-option)* < *distribution-name-or-keyword* < *(distribution-option)* > > . . . > < / *preprocess-options* > ;
OUTPUT < **OUT**=*SAS-data-set* > *output-options* ;
OUTSCORELIB < **OUTLIB=** > *fcmp-library-name options* ;
NLOPTIONS *options* ;
PERFORMANCE *options* ;
Programming statements ;

Functional Summary

Table 8.1 summarizes the statements and options that control the HPSEVERITY procedure.

Table 8.1 Functional Summary

Description	Statement	Option
Statements		
Specifies BY-group processing	BY	
Specifies the response variable to model along with censoring and truncation effects	LOSS	
Specifies the weight variable	WEIGHT	
Specifies the classification variables	CLASS	
Specifies the regression effects to model	SCALEMODEL	
Specifies distributions to fit	DIST	
Specifies the scoring functions and quantiles to write	OUTPUT	
Specifies the library to write scoring functions to	OUTSCORELIB	
Specifies optimization options	NLOPTIONS	
Specifies performance options	PERFORMANCE	
Specifies programming statements that define an objective function	Programming statements	
Input and Output Options		
Specifies that the OUTEST= data set contain covariance estimates	PROC HPSEVERITY	COVOUT
Specifies the input data set	PROC HPSEVERITY	DATA=
Specifies the input data set for parameter estimates	PROC HPSEVERITY	INEST=

Table 8.1 *continued*

Description	Statement	Option
Specifies the input item store for parameter initialization	PROC HPSEVERITY	INSTORE=
Limits the length of effect names	PROC HPSEVERITY	NAMELEN=
Specifies the output data set for estimates of scoring functions and quantiles	OUTPUT	OUT=
Specifies the output data set for CDF estimates	PROC HPSEVERITY	OUTCDF=
Specifies the output data set for parameter estimates	PROC HPSEVERITY	OUTEST=
Specifies the output data set for model information	PROC HPSEVERITY	OUTMODELINFO=
Specifies the output data set for statistics of fit	PROC HPSEVERITY	OUTSTAT=
Specifies the output item store for context and estimation results	PROC HPSEVERITY	OUTSTORE=
Data Interpretation Options		
Specifies left-censoring	LOSS	LEFTCENSORED=
Specifies left-truncation	LOSS	LEFTTRUNCATED=
Specifies the probability of observability	LOSS	PROBOBSERVED=
Specifies right-censoring	LOSS	RIGHTCENSORED=
Specifies right-truncation	LOSS	RIGHTTRUNCATED=
Model Estimation Options		
Specifies the model selection criterion	PROC HPSEVERITY	CRITERION=
Specifies the method for computing mixture distribution	SCALEMODEL	DFMIXTURE=
Specifies initial values for model parameters	DIST	INIT=
Specifies the objective function symbol	PROC HPSEVERITY	OBJECTIVE=
Specifies the offset variable in the scale regression model	SCALEMODEL	OFFSET=
Specifies the denominator for computing covariance estimates	PROC HPSEVERITY	VARDEF=
Empirical Distribution Function (EDF) Estimation Options		
Specifies the confidence level for reporting the confidence interval for EDF estimates	PROC HPSEVERITY	EDFALPHA=
Specifies the nonparametric method of CDF estimation	PROC HPSEVERITY	EMPIRICALCDF=
Specifies the sample to be used for computing the EDF estimates	PROC HPSEVERITY	INITSAMPLE
EMPIRICALCDF=MODIFIEDKM Options		
Specifies the α value for the lower bound on risk set size	PROC HPSEVERITY	ALPHA=
Specifies the c value for the lower bound on risk set size	PROC HPSEVERITY	C=

Table 8.1 *continued*

Description	Statement	Option
Specifies the absolute lower bound on risk set size	PROC HPSEVERITY	RSLB=
EMPIRICALCDF=TURNBULL Options		
Specifies that the final EDF estimates be maximum likelihood estimates	PROC HPSEVERITY	ENSUREMLE
Specifies the relative convergence criterion	PROC HPSEVERITY	EPS=
Specifies the maximum number of iterations	PROC HPSEVERITY	MAXITER=
Specifies the threshold below which an EDF estimate is deemed to be 0	PROC HPSEVERITY	ZEROPROB=
OUT= Data Set Generation Options		
Specifies the variables to copy from the DATA= data set to the OUT= data set	OUTPUT	COPYVARS=
Specifies the scoring functions to estimate	OUTPUT	FUNCTIONS=
Specifies the quantiles to estimate	OUTPUT	QUANTILES=
Scoring Function Generation Options		
Specifies that scoring functions of all models be written to one package	OUTSCORELIB	COMMONPACKAGE
Specifies the output data set for BY-group identifiers	OUTSCORELIB	OUTBYID=
Specifies the output library for scoring functions	OUTSCORELIB	OUTLIB=
Displayed Output and Plotting Options		
Specifies that distributions be listed to the log without estimating any models that use them	DIST	LISTONLY
Limits or suppresses the display of class levels	PROC HPSEVERITY	NOCLPRINT
Suppresses all displayed and graphical output	PROC HPSEVERITY	NOPRINT
Specifies which graphical output to prepare	PROC HPSEVERITY	PLOTS=
Specifies which output to display	PROC HPSEVERITY	PRINT=
Specifies that distributions be validated without estimating any models that use them	DIST	VALIDATEONLY

PROC HPSEVERITY Statement

PROC HPSEVERITY *options* **;**

The PROC HPSEVERITY statement invokes the procedure. You can specify two types of *options* in the PROC HPSEVERITY 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 sets used by PROC HPSEVERITY and various forms of output generated by PROC HPSEVERITY. The *options* are listed in alphabetical order.

COVOUT
specifies that the OUTEST= data set 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 set, see the section "OUTEST= Data Set" on page 365.

DATA=*SAS-data-set*
names the input data set. If you do not specify the DATA= option, then the most recently created SAS data set is used.

EDFALPHA=*confidence-level*
specifies the confidence level in the (0,1) range that is used for computing the confidence intervals for the EDF estimates. The lower and upper confidence limits that correspond to this level are reported in the OUTCDF= data set, if specified, and are displayed in the plot that is created when you specify the PLOTS=CDFPERDIST option.

If you do not specify the EDFALPHA= option, then PROC HPSEVERITY uses a default value of 0.05.

INEST=*SAS-data-set*
names the input data set that contains the initial values of the parameter estimates to start the optimization process. 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 set. For more information about the variables in this data set, see the section "INEST= Data Set" on page 362.

If you specify the SCALEMODEL statement, then PROC HPSEVERITY reads the INEST= data set only if the SCALEMODEL statement contains singleton continuous effects. For more generic regression effects, you should save the estimates by specifying the OUTSTORE= item store in a step and then use the INSTORE= option to read those estimates. The INSTORE= option is the newer and more flexible method of specifying initial values for distribution and regression parameters.

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 HPSEVERITY 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 HPSEVERITY 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 HPSEVERITY, which can be useful for replicating the results across different runs. If you do not specify the seed value, PROC HPSEVERITY 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 set 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.

INSTORE=*store-name*
names the item store that contains the context and results of the severity model estimation process. An item store has a binary file format that cannot be modified. You must specify an item store that you have created in another PROC HPSEVERITY step by using the OUTSTORE= option.

The *store-name* is a usual one- or two-level SAS name, as for SAS data sets. If you specify a one-level name, then PROC HPSEVERITY reads the item store from the WORK library. If you specify a two-level name of the form *libname.membername*, then PROC HPSEVERITY reads the item store from the *libname* library.

This option is more flexible than the INEST= option, because it can read estimates of any type of scale regression model; the INEST= option can read only scale regression models that contain singleton continuous effects.

For more information about how the input item store is used for parameter initialization, see the sections "Parameter Initialization" on page 304 and "Parameter Initialization for Regression Models" on page 307.

NAMELEN=*number*
specifies the length to which long regression effect names are shortened. The default and minimum value is 20.

This option does not apply to the names of singleton continuous effects if you have not specified any CLASS variables.

NOCLPRINT< =*number* **>**
suppresses the display of the "Class Level Information" table if you do not specify *number*. If you specify *number*, the values of the classification variables are displayed for only those variables whose number of levels is less than *number*. Specifying a *number* helps to 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= and PLOTS= options is ignored.

OUTCDF=*SAS-data-set*
names the output data set to contain estimates of the cumulative distribution function (CDF) value at each of the observations. This data set is created only when you run PROC HPSEVERITY in single-machine mode.

The information is output for each specified model whose parameter estimation process converges. The data set also contains the estimates of the empirical distribution function (EDF). For more information about the variables in this data set, see the section "OUTCDF= Data Set" on page 364.

OUTEST=*SAS-data-set*
names the output data set to contain estimates of the parameter values and their standard errors for each model whose parameter estimation process converges. For more information about the variables in this data set, see the section "OUTEST= Data Set" on page 365.

If you specify the SCALEMODEL statement such that it contains at least one effect that is not a singleton continuous effect, then the OUTEST= data set that this option creates cannot be used as an INEST= data set in a subsequent PROC HPSEVERITY step. In such cases, it is recommended that you use the newer OUTSTORE= option to save the estimates and specify those estimates in a subsequent PROC HPSEVERITY step by using the INSTORE= option.

OUTMODELINFO=*SAS-data-set*
names the output data set to contain the information about each candidate distribution. For more information about the variables in this data set, see the section "OUTMODELINFO= Data Set" on page 366.

OUTSTAT=*SAS-data-set*
names the output data set to contain the values of statistics of fit for each model whose parameter estimation process converges. For more information about the variables in this data set, see the section "OUTSTAT= Data Set" on page 367.

OUTSTORE=*store-name*
names the item store to contain the context and results of the severity model estimation process. The resulting item store has a binary file format that cannot be modified. You can specify this item store in a subsequent PROC HPSEVERITY step by using the INSTORE= option.

The *store-name* is a usual one- or two-level SAS name, as for SAS data sets. If you specify a one-level name, then the item store resides in the WORK library and is deleted at the end of the SAS session. Because item stores are meant to be consumed by a subsequent PROC HPSEVERITY step for parameter initialization, typical usage specifies a two-level name of the form *libname.membername*.

This option is more useful than the OUTEST= option, especially when you specify a scale regression model that contains interaction effects or effects that have CLASS variables. You can initialize such scale regression models in a subsequent PROC HPSEVERITY step only by specifying the item store that this option creates as an INSTORE= item store in that step.

PLOTS <(*global-plot-options***)> <=***plot-request-option***>**

PLOTS <(*global-plot-options***)> <=(***plot-request-option . . . plot-request-option***)>**
specifies the desired graphical output. The graphical output is created only when you run PROC HPSEVERITY in single-machine mode. If you specify more than one *global-plot-option*, then separate them with spaces and enclose them in parentheses. If you specify more than one *plot-request-option*, then separate them with spaces and enclose them in parentheses.

You can specify the following *global-plot-options*:

HISTOGRAM
plots the histogram of the response variable on the PDF plots.

KERNEL
plots the kernel estimate of the probability density of the response variable on the PDF plots.

ONLY
turns off the default graphical output and creates only the requested plots.

You can specify the following *plot-request-options*:

ALL
creates all the graphical output.

CDF
creates a plot that compares the cumulative distribution function (CDF) estimates of all the candidate distribution models to the empirical distribution function (EDF) estimate. The plot does not contain CDF estimates for models whose parameter estimation process does not converge.

CDFPERDIST
creates a plot of the CDF estimates of each candidate distribution model. A plot is not created for models whose parameter estimation process does not converge.

CONDITIONALPDF < (*cpdf-options*) >
CONDPDF < (*cpdf-options*) >
creates a plot that compares the conditional PDF estimates of all the candidate distribution models. The plot does not contain conditional PDF estimates for models whose parameter estimation process does not converge.

A conditional PDF of a loss random variable Y in an interval $(Y_l, Y_r]$ is the probability that a specific loss value is observed, given that the loss values belong to that interval. Formally, the conditional PDF of y, denoted by $f^c(y)$, for the $(Y_l, Y_r]$ interval is defined as $f^c(y) = \Pr[Y = y | Y_l < Y \leq Y_r]$. If $f(y)$ and $F(y)$ denote the PDF and CDF at loss value y, respectively, then $f^c(y)$ for the $(Y_l, Y_r]$ interval is computed as $f^c(y) = f(y)/(F(Y_r) - F(Y_l))$. The scaling factor of $1/(F(Y_r) - F(Y_l))$ ensures that the conditional PDF is a true PDF that integrates to 1 in the $(Y_l, Y_r]$ interval.

PROC HPSEVERITY prepares a conditional PDF comparison plot that contains at most three regions (intervals) of mutually exclusive ranges of the loss variable's value:

- Left-tail: $(y_{\min} - \epsilon, L]$,
- Center: $(L, R]$, and
- Right-tail: $(R, y_{\max}]$,

where $y_{\min}$ and $y_{\max}$ denote the smallest and largest values of the loss variable in the DATA= data set, respectively, and ϵ denotes a small machine-precision constant for a double-precision value.

You can specify the following *cpdf-options* to control how the values of L and R are computed and which regions are displayed:

LEFTQ | LEFT | L=*number*

specifies the CDF value, between 0 and 1, to mark the end of the left-tail region. The left-tail region always starts at the minimum loss variable value in the DATA= data set. The value of L, the end of the left-tail region, is determined by the *number* that you specify. Let the *number* be p_l. If you do not specify the QUANTILEBOUNDS option, then PROC HPSEVERITY sets L equal to the $100p_l$th percentile. If you specify the QUANTILEBOUNDS option, then for a distribution D with an estimated quantile function $\hat{Q}_D$, $L_D = \hat{Q}_D(p_l)$ marks the end of the left-tail region. L_D can be different for each distribution, so the left-tail region ends at different values for different distributions.

RIGHTQ | RIGHT | R=*number*

specifies the CDF value, between 0 and 1, to mark the start of the right-tail region. The right-tail region always ends at the maximum loss variable value in the DATA= data set. The value of R, the start of the right-tail region, is determined by the *number* that you specify. Let the *number* be p_r. If you do not specify the QUANTILEBOUNDS option, then PROC HPSEVERITY sets R equal to the $100p_r$th percentile. If you specify the QUANTILEBOUNDS option, then for a distribution D with an estimated quantile function $\hat{Q}_D$, $R_D = \hat{Q}_D(p_r)$ marks the start of the right-tail region. R_D can be different for each distribution, so the right-tail region starts at different values for different distributions.

QUANTILEBOUNDS

specifies that the region boundaries be computed by using the estimated quantile functions of individual distributions. If you do not specify this option, then the boundaries are computed by using the percentiles, which are quantiles from the empirical distribution.

When you specify this option, the left-tail region of different distributions can end at different values and the right-tail region of different distributions can start at different values, because the quantile function of different distributions can produce different values for the same CDF value.

SHOWREGION | SHOW=*region-option*

SHOWREGION | SHOW=(*region-options***)**

specifies the regions to display in the plot. You can specify any combination of the following *region-options*:

CENTER | C

specifies that the center region of the plot, which is the region between the end of the left-tail region and the beginning of the right-tail region, be shown. If you specify this option, you must also specify valid values for both the LEFTQ= and RIGHTQ= options.

LEFT | L

specifies that the left-tail region of the plot be shown. If you specify this option, you must also specify a valid value for the LEFTQ= option.

RIGHT | R

specifies that the right-tail region of the plot be shown. If you specify this option, you must also specify a valid value for the RIGHTQ= option.

If you do not specify the SHOWREGION option, then PROC HPSEVERITY determines the default displayed regions as follows:

- If you do not specify either the LEFTQ= or RIGHTQ= option, then this is equivalent to specifying (LEFTQ=0.25 RIGHTQ=0.75), and PROC HPSEVERITY displays all three regions (left-tail, center, and right-tail).
- If you specify valid values for both the LEFTQ= and RIGHTQ= options, then PROC HPSEVERITY displays all three regions (left-tail, center, and right-tail).
- If you specify a valid value for the LEFTQ= option but do not specify the RIGHTQ= option, then PROC HPSEVERITY displays two regions: left-tail and the remaining region that combines the center and right-tail regions.
- If you specify a valid value for the RIGHTQ= option but do not specify the LEFTQ= option, then PROC HPSEVERITY displays two regions: right-tail and the remaining region that combines the center and left-tail regions.

Whether you specify the SHOWREGION option or not, PROC HPSEVERITY does not display a region if the region contains fewer than five observations, and it issues a corresponding warning in the SAS log.

For an illustration of the CONDITIONALPDF option, see "Example 8.3: Defining a Model for Mixed-Tail Distributions" on page 382.

CONDITIONALPDFPERDIST < (*cpdf-options*) >

CONDPDFDIST < (*cpdf-options*) >

creates a plot of the conditional PDF estimates of each candidate distribution model. A plot is not created for models whose parameter estimation process does not converge.

The *cpdf-options* are identical to those listed for the CONDITIONALPDF plot option, except that they are interpreted in the context of each candidate distribution individually. You can specify a different set of values for the *cpdf-options* in the CONDITIONALPDFPERDIST option than you specify in the CONDITIONALPDF option.

For an illustration of the CONDITIONALPDFPERDIST option, see "Example 8.4: Fitting a Scaled Tweedie Model with Regressors" on page 389.

NONE

creates none of the graphical output. If you specify this option, then it overrides all the other *plot-request-options*. The default graphical output is also suppressed.

PDF

creates a plot that compares the probability density function (PDF) estimates of all the candidate distribution models. The plot does not contain PDF estimates for models whose parameter estimation process does not converge.

PDFPERDIST

creates a plot of the PDF estimates of each candidate distribution model. A plot is not created for models whose parameter estimation process does not converge.

PP

creates the probability-probability plot (known as the P-P plot), which compares the CDF estimate of each candidate distribution model to the empirical distribution function (EDF). The data that are shown in this plot are used for computing the EDF-based statistics of fit.

QQ
creates the quantile-quantile plot (known as the Q-Q plot), which compares the empirical quantiles to the quantiles of each candidate distribution model.

If you do not specify the PLOTS= option or if you do not specify the ONLY *global-plot-option*, then the default graphical output is equivalent to specifying PLOTS(HISTOGRAM KERNEL)=(CDF PDF).

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).

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 304.

The following *options* control the model estimation and selection process:

CRITERION | CRITERIA | CRIT=*criterion-option*
specifies the model selection criterion.

If you specify two or more candidate models for estimation, then the one with the best value for the selection criterion is chosen as the best model. If you specify the OUTSTAT= data set, then the best model's observation has a value of 1 for the _SELECTED_ variable.

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.

CUSTOM
specifies the custom objective function as the selection criterion. You can specify this only if you also specify the OBJECTIVE= option. 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 $-2 * \log(L)$ 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 327.

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^{α} proposed by Lai and Ying (1991). You can specify the values of c and α 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 α when the lower bound on the risk set size is defined as cn^{α}. 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^{α}. 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 HPSEVERITY 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 HPSEVERITY 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 HPSEVERITY 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 320.

OBJECTIVE=*symbol-name*
names the symbol that represents the objective function in the SAS programming statements that you specify. For each model to be estimated, PROC HPSEVERITY executes the programming statements to compute the value of this symbol for each observation. The values are added across all observations to obtain the value of the objective function. The optimization algorithm estimates the model parameters such that the objective function value is *minimized*. A separate optimization problem is solved for each candidate distribution. If you specify a BY statement, then a separate optimization problem is solved for each candidate distribution within each BY group.

For more information about writing SAS programming statements to define your own objective function, see the section "Custom Objective Functions" on page 359.

BY Statement

BY *variable-list* **;**

A BY statement can be used in the HPSEVERITY procedure to process the input data set in groups of observations defined by the BY variables.

If you specify the BY statement, then PROC HPSEVERITY expects the input data set to be sorted in the order of the BY variables unless you specify the NOTSORTED option.

The BY statement is always supported in the single-machine mode of execution. For the distributed mode, it is supported only when the DATA= data set resides on the client machine. In other words, the BY statement is supported only in the client-data (or local-data) mode of the distributed computing model and not for any of the alongside modes, such as the alongside-the-database or alongside-HDFS mode.

CLASS Statement

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

The CLASS statement names the classification variables to be used in the scale regression model. 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 311.

If you specify a CLASS statement, then it must precede the SCALEMODEL statement.

You can specify options either as individual variable *options* or as *global-options*. You can specify *options* for each variable by enclosing the options in parentheses after the variable name. You can also specify *global-options* for the CLASS statement by placing them after a slash (/). *Global-options* are applied to all the variables that you specify 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*.

You can specify the following values for either an *option* or a *global-option*:

DESCENDING
DESC
reverses the sort order of the classification variable. If you specify both the DESCENDING and ORDER= options, the HPSEVERITY procedure orders the levels of classification variables according to the ORDER= option and then reverses that order.

ORDER=DATA | FORMATTED | INTERNAL
ORDER=FREQ | FREQDATA | FREQFORMATTED | FREQINTERNAL
specifies the sort order for the levels of classification variables. This order is used by the parameterization method to create the parameters in the model. By default, ORDER=FORMATTED. For ORDER=FORMATTED and ORDER=INTERNAL, the sort order is machine-dependent. When ORDER=FORMATTED is in effect for numeric variables for which you have supplied no explicit format, the levels are ordered by their internal values.

The following table shows how the HPSEVERITY procedure interprets values of the ORDER= option:

Value of ORDER=	Levels Sorted By
DATA	Order of appearance in the input data set
FORMATTED	External formatted values, except for numeric variables that have no explicit format, which are sorted by their unformatted (internal) values
FREQ	Descending frequency count (levels that have more observations come earlier in the order)
FREQDATA	Order of descending frequency count, and within counts by order of appearance in the input data set when counts are tied
FREQFORMATTED	Order of descending frequency count, and within counts by formatted value when counts are tied
FREQINTERNAL	Order of descending frequency count, and within counts by unformatted (internal) value when counts are tied
INTERNAL	Unformatted value

For more information about sort order, see the chapter about the SORT procedure in *Base SAS Procedures Guide* and the discussion of BY-group processing in *SAS Language Reference: Concepts*.

REF=*'level'* | *keyword*
REFERENCE=*'level'* | *keyword*
specifies the reference level that is used when you specify PARAM=REFERENCE. For an individual (but not a global) variable REF= *option*, you can specify the *level* of the variable to use as the reference level. Specify the formatted value of the variable if a format is assigned. For a REF= *option* or *global-option*, you can use one of the following *keywords*:

FIRST designates the first-ordered level as reference.

LAST designates the last-ordered level as reference.

By default, REF=LAST.

If you choose a reference level for any CLASS variable, all variables are parameterized in the reference parameterization for computational efficiency. In other words, the HPSEVERITY procedure applies a single parameterization method to all classification variables.

Suppose that the variable temp has three levels (`'hot'`, `'warm'`, and `'cold'`) and that the variable gender has two levels (`'M'` and `'F'`). The following statements fit a scale regression model:

```
proc hpseverity;
   loss y;
   class gender(ref='F') temp;
   scalemodel gender*temp gender;
run;
```

Both CLASS variables are in reference parameterization in this model. The reference levels are `'F'` for the variable gender and `'warm'` for the variable temp, because the statements are equivalent to the following statements:

```
proc hpseverity;
   loss y;
   class gender(ref='F') temp(ref=last);
   scalemodel gender*temp gender;
run;
```

You can specify the following *global-options*:

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, even if the CLASS variables are not used in the model formulation.

PARAM=*keyword*
specifies the parameterization method for the classification variable or variables. You can specify the following *keywords*:

GLM specifies a less-than-full-rank reference cell coding.

REFERENCE specifies a reference cell encoding. You can choose the reference value by specifying an option for a specific *variable* or set of *variables* in the CLASS statement, or you can designate the first- or last-ordered value by specifying a *global-option*. By default, REFERENCE=LAST.

The GLM parameterization is the default. 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 313.

TRUNCATE< =*n* >
specifies the truncation width of formatted values of CLASS variables when the optional *n* is specified.

If *n* is not specified, the TRUNCATE option requests that classification levels be determined by using no more than the first 16 characters of the formatted values of CLASS variables.

DIST Statement

DIST *distribution-name-or-keyword < (distribution-option) < distribution-name-or-keyword < (distribution-option) > > . . . > < / preprocess-options >* **;**

The DIST statement specifies candidate distributions to be estimated by the HPSEVERITY procedure. You can specify multiple DIST statements, and each statement can contain one or more distribution specifications.

For your convenience, PROC HPSEVERITY 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 290.

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 HPSEVERITY 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 334.

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:

ALL
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.

If you specify the OUTCDF= option or request a CDF plot and you do not specify any DIST statement, then PROC HPSEVERITY does not fit any distributions and produces the empirical estimates of the cumulative distribution function.

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 HPSEVERITY 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 305.

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 304. If none of the initialization methods is used, then PROC HPSEVERITY initializes all parameters to 0.001.

You can specify the following *preprocess-options* in the DIST statement:

LISTONLY
specifies that the list of all candidate distributions be printed 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.

VALIDATEONLY
specifies that all candidate distributions be checked for validity without doing any further processing on them. If a distribution is invalid, the reason for invalidity is written to the SAS log. If all distributions are valid, then the distribution information is written to the SAS log. The information includes name, description, validity status (valid or invalid), and number of distribution parameters. The information is not written to the SAS log if you specify an OUTMODELINFO= data set or the PRINT=DISTINFO or PRINT=ALL option in the PROC HPSEVERITY statement. This option is especially useful when you specify your own distributions or when you specify the _USER_ or _ALL_ keywords in the DIST statement. It enables you to check whether your custom distribution definitions satisfy PROC HPSEVERITY'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 HPSEVERITY to perform additional checks to validate whether regression effects can be modeled on each candidate distribution.

LOSS Statement

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 set that you specify by using the DATA= option in the PROC HPSEVERITY 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:

LEFTCENSORED | LC=*variable-name*

LEFTCENSORED | LC=*number*

specifies the left-censoring variable or a global left-censoring limit.

You can use the *variable-name* argument to specify a data set variable that contains the left-censoring limit. If the value of this variable is missing, then PROC HPSEVERITY assumes that such observations are not left-censored.

Alternatively, you can use the *number* argument to specify a left-censoring limit value that applies to all the observations in the data set. 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 HPSEVERITY 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 HPSEVERITY 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 300.

LEFTTRUNCATED | LT=*variable-name* < *(left-truncation-option)* >

LEFTTRUNCATED | LT=*number* < *(left-truncation-option)* >

specifies the left-truncation variable or a global left-truncation threshold.

You can use the *variable-name* argument to specify a data set variable that contains the left-truncation threshold. If the value of this variable is missing or 0 for some observations, then PROC HPSEVERITY assumes that such observations are not left-truncated.

Alternatively, you can use the *number* argument to specify a left-truncation threshold that applies to all the observations in the data set. 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 300.

You can specify the following *left-truncation-option* for an alternative interpretation of the left-truncation threshold:

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 HPSEVERITY 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 301.

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 set variable that contains the right-censoring limit. If the value of this variable is missing, then PROC HPSEVERITY 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 set. 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 HPSEVERITY 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 HPSEVERITY 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 300.

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 set variable that contains the right-truncation threshold. If the value of this variable is missing for some observations, then PROC HPSEVERITY 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 set. 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 300.

NLOPTIONS Statement

NLOPTIONS *options* **;**

The HPSEVERITY procedure uses the nonlinear optimization (NLO) subsystem to perform the nonlinear optimization of the likelihood function to obtain the estimates of distribution and regression parameters. You can use the NLOPTIONS statement to control different aspects of this optimization process. 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), that is used in the HPSEVERITY procedure:

```
nloptions tech=dbldog maxiter=200;
```

A discussion of the full range of *options* that can be used in the NLOPTIONS statement is given in Chapter 6, "Nonlinear Optimization Methods" (*SAS/ETS User's Guide*). The HPSEVERITY procedure supports all those options except the options that are related to displaying the optimization information. You can use the PRINT= option in the PROC HPSEVERITY statement to request the optimization summary and iteration history. If you specify more than one NLOPTIONS statement, then the first statement is used.

OUTPUT Statement

OUTPUT < **OUT**=*SAS-data-set*> *output-options* **;**

The OUTPUT statement specifies the data set to write the estimates of scoring functions and quantiles to. To specify the name of the output data set, use the following option:

OUT=*SAS-data-set*
specifies the name of the output data set. If you do not specify this option, then PROC HPSEVERITY names the output data set by using the DATA*n* convention.

In alongside-the-database mode, the data in the DATA= data set are read in distributed form, minimizing data movement for best performance. Similarly, when PROC HPSEVERITY executes in distributed mode and when the libref of the OUT= data set points to the database appliance, PROC HPSEVERITY

writes the OUT= data in parallel to the database. For more information, see the section "Output Data Sets" on page 29.

To control the contents of the OUT= data set, specify the following *output-options*:

COPYVARS=*variable-list*
specifies the names of the variables that you want to copy from the input DATA= data set to the OUT= data set. 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 set, 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 defined in the search path that you specify by using the CMPLIB= system option.

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 set, PROC HPSEVERITY 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:

Right-Censored	Left-Censored	v
No	No	y
No	Yes	l
Yes	No	r
Yes	Yes	$(l + r)/2$

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 HPSEVERITY 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:

```
dist exp logn;
output out=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 HPSEVERITY creates the following three scoring function variables for the exponential (EXP) distribution in the Work.Score data set:

EXP_CDF contains the CDF estimate $F_{\text{exp}}(v, \hat{\theta})$, where F_{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_{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 HPSEVERITY creates the following three scoring function variables for the lognormal (LOGN) distribution in the Work.Score data set:

LOGN_CDF contains the CDF estimate $F_{\text{logn}}(v, \hat{\mu}, \hat{\sigma})$, where F_{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_{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 *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.

For more information about scoring functions, see the section "Scoring Functions" on page 352.

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_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. *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 HPSEVERITY 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 HPSEVERITY creates a variable name P*t*, where *t* is the text representation of $100p_i$ that 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 HPSEVERITY 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 HPSEVERITY creates three quantile estimate variables, BURR_NINETY, BURR_VAR, and BURR_P99_5, in the Work.Score data set 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_{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 set 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 set, see the section "OUT= Data Set" on page 363.

OUTSCORELIB Statement

OUTSCORELIB < **OUTLIB**=> *fcmp-library-name options* **;**

The OUTSCORELIB statement specifies the library to write scoring functions to. Scoring functions enable you to easily compute a distribution function on the fitted parameters of the distribution without going through a potentially complex process of extracting the fitted parameter estimates from other output such as the OUTEST= data set that is created by PROC HPSEVERITY.

If you specify the SCALEMODEL statement and if you specify interaction or classification effects, then PROC HPSEVERITY ignores the OUTSCORELIB statement and does not generate scoring functions. In other words, if you specify the SCALEMODEL statement, then PROC HPSEVERITY generates scoring functions if you specify only singleton continuous effects in the SCALEMODEL statement.

You must specify the following option as the first option in the statement:

OUTLIB=*fcmp-library-name*
names the FCMP library to contain the scoring functions. PROC HPSEVERITY writes the scoring functions to the FCMP library named *fcmp-library-name*. If a library or data set named *fcmp-library-name* already exists, PROC HPSEVERITY deletes it before proceeding.

This option is similar to the OUTLIB= option that you would specify in a PROC FCMP statement, except that *fcmp-library-name* must be a two-level name whereas the OUTLIB= option in the PROC FCMP statement requires a three-level name. The third level of a three-level name specifies the package to which the functions belong. You do not need to specify the package name in the *fcmp-library-name*, because PROC HPSEVERITY automatically creates the package for you. By default, a separate package is created for each distribution that has not failed to converge. Each package is named for a distribution. For example, if you define and fit a distribution named *mydist*, and if *mydist* does not fail to converge, then PROC HPSEVERITY creates a package named *mydist* in the OUTLIB= library that you specify. Further, let the definition of the *mydist* distribution contain three distribution functions, *mydist_PDF(x,Parm1,Parm2)*, *mydist_LOGCDF(x,Parm1,Parm2)*, and *mydist_XYZ(x,Parm1,Parm2)*. If you specify the OUTSCORELIB statement

```
outscorelib outlib=sasuser.scorefunc;
```

then the Sasuser.Scorefunc library contains the following three functions in a package named *mydist*: *SEV_PDF(x)*, *SEV_LOGCDF(x)*, and *SEV_XYZ(x)*.

The key feature of scoring functions is that they do not require the parameter arguments (*Parm1* and *Parm2* in this example). The fitted parameter estimates are encoded inside the scoring function so that you can compute or score the value of each function for a given value of the loss variable without having to know or extract the parameter estimates through some other means.

For convenience, you can omit the OUTLIB= portion of the specification and just specify the name, as in the following example:

```
outscorelib sasuser.scorefunc;
```

When the HPSEVERITY procedure runs successfully, the *fcmp-library-name* is appended to the CMPLIB system option, so you can immediately start using the scoring functions in a DATA step or PROC FCMP step.

You can specify the following *options* in the OUTSCORELIB statement:

COMMONPACKAGE
ONEPACKAGE
requests that only one common package be created to contain all the scoring functions.

If you specify this option, then all the scoring functions are created in a package called *sevfit*. For each distribution function that has the name *distribution_suffix*, the name of the corresponding scoring function is formed as *SEV_suffix_distribution*. For example, the scoring function of the distribution function 'MYDIST_BAR' is named 'SEV_BAR_MYDIST'.

If you do not specify this option, then all scoring functions for a distribution are created in a package that has the same name as the distribution, and for each distribution function that has the name *distribution_suffix*, the name of the corresponding scoring function is formed as *SEV_suffix*. For example, the scoring function of the distribution function 'MYDIST_BAR' is named 'SEV_BAR'.

OUTBYID=*SAS-data-set*

names the output data set to contain the unique identifier for each BY group. This unique identifier is used as part of the name of the package or scoring function for each distribution. This is a required option when you specify a BY statement in PROC HPSEVERITY.

The OUTBYID= data set contains one observation per BY group and a variable named _ID_ in addition to the BY variables that you specify in the BY statement. The _ID_ variable contains the unique identifier for each BY group. The identifier of the BY group is the decimal representation of the sequence number of the BY group. The first BY group has an identifier of 1, the second BY group has an identifier of 2, the tenth BY group has an identifier of 10, and so on.

If you do not specify the COMMONPACKAGE option in the OUTSCORELIB statement, then for each distribution, PROC HPSEVERITY creates as many packages as the number of BY groups. The unique BY-group identifier is used as a suffix for the package name. For example, if your DATA= data set has three BY groups and if you specify the OUTSCORELIB statement

```
outscorelib outlib=sasuser.byscorefunc outbyid=sasuser.byid;
```

then for the distribution 'MYDIST', the Sasuser.Byscorefunc library contains the three packages 'MYDIST1', 'MYDIST2', and 'MYDIST3', and each package contains one scoring function named 'SEV_BAR' for each distribution function named 'MYDIST_BAR'.

If you specify the COMMONPACKAGE option in the OUTSCORELIB statement, PROC HPSEVERITY creates as many versions of the distribution function as the number of BY groups. The unique BY-group identifier is used as a suffix for the function name. Extending the previous example, if you specify the OUTSCORELIB statement with the COMMONPACKAGE option,

```
outscorelib outlib=sasuser.byscorefunc outbyid=sasuser.byid commonpackage;
```

then for the distribution function 'MYDIST_BAR' of the distribution 'MYDIST', the Sasuser.Byscorefunc library contains the following three scoring functions: 'SEV_BAR_MYDIST1', 'SEV_BAR_MYDIST2', and 'SEV_BAR_MYDIST3'. All the scoring functions are created in one common package named *sevfit*.

For both the preceding examples, the Sasuser.Byid data set contains three observations, one for each BY group. The value of the _ID_ variable is 1 for the first BY group, 2 for the second BY group, and 3 for the third BY group.

For more information about scoring functions, see the section "Scoring Functions" on page 352.

PERFORMANCE Statement

PERFORMANCE *options* **;**

The PERFORMANCE statement defines performance parameters for distributed and multithreaded computing, passes variables that describe the distributed computing environment, and requests detailed results about the performance characteristics of PROC HPSEVERITY.

You can also use the PERFORMANCE statement to control whether a high-performance analytical procedure runs in single-machine or distributed mode.

The PERFORMANCE statement is documented further in the section "PERFORMANCE Statement" on page 31.

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 $\mathbf{X}\boldsymbol{\beta}$ 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 more information about effect construction rules, see the section "Specification and Parameterization of Model Effects" on page 313.

All the regressor variables must be present in the input data set that you specify by using the DATA= option in the PROC HPSEVERITY statement. The scale parameter of each candidate distribution is linked to the linear predictor $\mathbf{X}\boldsymbol{\beta}$ 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 305.

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 given distribution family, each observation in the input data set implies a different scaled version of the distribution. To compute estimates of CDF and PDF that are comparable across different distribution families, PROC HPSEVERITY 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 309.

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 HPSEVERITY 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 HPSEVERITY uses the default of 15.

SEED=*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 HPSEVERITY generates a seed internally that is based on the current value of the system clock.

OFFSET=*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 306.

WEIGHT Statement

WEIGHT *variable-name* **;**

The WEIGHT statement specifies the name of a variable whose values represent the weight of each observation. PROC HPSEVERITY 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.

Programming Statements

You can use a series of programming statements that use variables in the input data set that you specify in the DATA= option in the PROC HPSEVERITY statement to assign a value to an objective function symbol. You must specify the objective function symbol by using the OBJECTIVE= option in the PROC HPSEVERITY statement. If you do not specify the OBJECTIVE= option in the PROC HPSEVERITY statement, then the programming statements are ignored and models are estimated using the maximum likelihood method.

You can use most DATA step statements and functions in your program. Any additional functions, restrictions, and differences are listed in the section "Custom Objective Functions" on page 359.

Details: HPSEVERITY Procedure

Predefined Distributions

For the response variable Y, PROC HPSEVERITY assumes the model

$$Y \sim \mathcal{F}(\Theta)$$

where $\mathcal{F}$ is a continuous probability distribution with parameters Θ. 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 HPSEVERITY 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 HPSEVERITY procedure. A summary of the distributions is provided in Table 8.2. 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 μ 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 334.

Table 8.2 Predefined PROC HPSEVERITY Distributions

Name	Distribution	Parameters	PDF (f) and CDF (F)
BURR	Burr (Type XII)	$\theta > 0, \alpha > 0,$ $\gamma > 0$	$f(x) = \frac{\alpha\gamma z^{\gamma}}{x(1+z^{\gamma})^{(\alpha+1)}}$ $F(x) = 1 - \left(\frac{1}{1+z^{\gamma}}\right)^{\alpha}$
EXP	Exponential	$\theta > 0$	$f(x) = \frac{1}{\theta}e^{-z}$ $F(x) = 1 - e^{-z}$
GAMMA	Gamma	$\theta > 0, \alpha > 0$	$f(x) = \frac{z^{\alpha}e^{-z}}{x\Gamma(\alpha)}$ $F(x) = \frac{\gamma(\alpha,z)}{\Gamma(\alpha)}$
GPD	Generalized Pareto	$\theta > 0, \xi > 0$	$f(x) = \frac{1}{\theta}(1+\xi z)^{-1-1/\xi}$ $F(x) = 1 - (1+\xi z)^{-1/\xi}$
IGAUSS	Inverse Gaussian (Wald)	$\theta > 0, \alpha > 0$	$f(x) = \frac{1}{\theta}\sqrt{\frac{\alpha}{2\pi z^3}}\, e^{\frac{-\alpha(z-1)^2}{2z}}$ $F(x) = \Phi\left((z-1)\sqrt{\frac{\alpha}{z}}\right) +$ $\Phi\left(-(z+1)\sqrt{\frac{\alpha}{z}}\right) e^{2\alpha}$
LOGN	Lognormal	μ (no bounds), $\sigma > 0$	$f(x) = \frac{1}{x\sigma\sqrt{2\pi}}e^{-\frac{1}{2}\left(\frac{\log(x)-\mu}{\sigma}\right)^2}$ $F(x) = \Phi\left(\frac{\log(x)-\mu}{\sigma}\right)$
PARETO	Pareto (Type II)	$\theta > 0, \alpha > 0$	$f(x) = \frac{\alpha\theta^{\alpha}}{(x+\theta)^{\alpha+1}}$ $F(x) = 1 - \left(\frac{\theta}{x+\theta}\right)^{\alpha}$
TWEEDIE	Tweedie**	$p > 1, \mu > 0,$ $\phi > 0$	$f(x) = a(x,\phi)\exp\left[\frac{1}{\phi}\left(\frac{x\mu^{1-p}}{1-p} - \kappa(\mu,p)\right)\right]$ $F(x) = \int_0^x f(t)dt$
STWEEDIE	Scaled Tweedie**	$\theta > 0, \lambda > 0,$ $1 < p < 2$	$f(x) = a(x,\theta,\lambda,p)\exp\left(-\frac{x}{\theta} - \lambda\right)$ $F(x) = \int_0^x f(t)dt$
WEIBULL	Weibull	$\theta > 0, \tau > 0$	$f(x) = \frac{1}{x}\tau z^{\tau}e^{-z^{\tau}}$ $F(x) = 1 - e^{-z^{\tau}}$

**For more information, see the section "Tweedie Distributions" on page 292.

Notes:

1. $z = x/\theta$, wherever z is used.
2. θ 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 μ is the mean of the distribution, ϕ 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),

$$f(x; \mu, \phi, p) = a(x, \phi) \exp\left[\frac{1}{\phi}\left(\frac{x\mu^{1-p}}{1-p} - \kappa(\mu, p)\right)\right]$$

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 α and scale parameter θ. 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 λ, α, and θ are related to the natural parameters μ, ϕ, 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 HPSEVERITY procedure. The first version, named TWEEDIE and defined for $p > 1$, has the natural parameterization with parameters μ, ϕ, 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 θ, Poisson mean parameter λ, and the index parameter p. The index parameter decides the shape parameter α of the gamma distribution as

$$\alpha = \frac{2-p}{p-1}$$

The parameters θ and λ of the STWEEDIE distribution are related to the parameters μ and ϕ 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 α 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 HPSEVERITY 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 305. The gamma scale parameter θ 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 θ. Then, you can easily estimate how the mean of the distribution μ 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 λ 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 $(1 \le i \le n)$, the estimate of kth raw moment is denoted by m_k and computed as

$$m_k = \frac{1}{n}\sum_{i=1}^{n} y_i^k$$

The $100p$th percentile is denoted by π_p $(0 \le p \le 1)$. By definition, π_p satisfies

$$F(\pi_p-) \le p \le F(\pi_p)$$

where $F(\pi_p-) = \lim_{h\downarrow 0} F(\pi_p - h)$. PROC HPSEVERITY uses the following practical method of computing π_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 $\hat{F}_n(y_p^+) \ge p$. 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 ϵ 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 2006). The predefined BURR distribution in PROC HPSEVERITY implements Burr's Type XII

distribution. The parameters are initialized by using the method of moments. The kth 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 $2m_3 - 3m_1m_2 > 0$, 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 $2m_3 - 3m_1m_2 < \epsilon$, 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 kth 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 α 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:

$$\begin{aligned} l &= \sum_{i=1}^{n} \log\left(y_i^{\alpha-1} \frac{e^{-y_i/\theta}}{\theta^\alpha \Gamma(\alpha)} \right) \\ &= (\alpha - 1) \sum_{i=1}^{n} \log(y_i) - \frac{1}{\theta} \sum_{i=1}^{n} y_i - n\alpha \log(\theta) - n \log(\Gamma(\alpha)) \end{aligned}$$

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 θ:

$$\hat{\theta} = \frac{\sum y_i}{n\alpha} = \frac{m_1}{\alpha}$$

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:

$$\hat{\psi}(\alpha) \approx \log(\alpha) - \frac{1}{\alpha}\left(0.5 + \frac{1}{12\alpha + 2}\right)$$

This approximation is within 1.4% of the true value for all the values of $\alpha > 0$ except when α is arbitrarily close to the positive root of the digamma function (which is approximately 1.461632). Even for the values of α that are close to the positive root, the absolute error between true and approximate values is still acceptable ($|\hat{\psi}(\alpha) - \psi(\alpha)| < 0.005$ for $\alpha > 1.07$). Solving the equation that arises from this approximation yields the following estimate of α:

$$\hat{\alpha} = \frac{3 - d + \sqrt{(d-3)^2 + 24d}}{12d}$$

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 α:

$$\hat{\alpha} = \frac{m_1^2}{m_2 - m_1^2}$$

If $m_2 - m_1^2 < \epsilon$ (almost zero sample variance), then α is initialized as follows:

$$\hat{\alpha} = 1$$

After computing the estimate of α, the estimate of θ 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:

$$\begin{aligned} F(x) &= 1 - \left(1 + \frac{\xi x}{\theta}\right)^{-1/\xi} \\ &= 1 - \left(\frac{\theta/\xi}{x + \theta/\xi}\right)^{1/\xi} \end{aligned}$$

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 μ and shape parameter λ, is as follows (Klugman, Panjer, and Willmot 1998, p. 583):

$$f(x) = \sqrt{\frac{\lambda}{2\pi x^3}}\ \exp\left(\frac{-\lambda(x-\mu)^2}{2\mu^2 x}\right)$$
$$F(x) = \Phi\left(\left(\frac{x}{\mu}-1\right)\sqrt{\frac{\lambda}{x}}\right) + \Phi\left(-\left(\frac{x}{\mu}+1\right)\sqrt{\frac{\lambda}{x}}\right)\ \exp\left(\frac{2\lambda}{\mu}\right)$$

For this parameterization, it is known that the mean is $E[X] = \mu$ and the variance is $\mathrm{Var}[X] = \mu^3/\lambda$, which yields the second raw moment as $E[X^2] = \mu^2(1 + \mu/\lambda)$ (computed by using $E[X^2] = \mathrm{Var}[X] + (E[X])^2$).

The predefined IGAUSS distribution in PROC HPSEVERITY uses the following alternate parameterization to allow the distribution to have a scale parameter, θ:

$$f(x) = \sqrt{\frac{\alpha\theta}{2\pi x^3}}\ \exp\left(\frac{-\alpha(x-\theta)^2}{2x\theta}\right)$$
$$F(x) = \Phi\left(\left(\frac{x}{\theta}-1\right)\sqrt{\frac{\alpha\theta}{x}}\right) + \Phi\left(-\left(\frac{x}{\theta}+1\right)\sqrt{\frac{\alpha\theta}{x}}\right)\ \exp\left(2\alpha\right)$$

The parameters θ (scale) and α (shape) of this alternate form are related to the parameters μ and λ 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

$$E[X] = \theta$$
$$E[X^2] = \theta^2\left(1 + \frac{1}{\alpha}\right)$$

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:

$$\hat{\mu} = 2\log(m1) - \frac{\log(m2)}{2}, \quad \hat{\sigma} = \sqrt{\log(m2) - 2\log(m1)}$$

PARETO The predefined PARETO distribution in PROC HPSEVERITY implements the Type II Pareto distribution with the location parameter set to 0. This predefined PARETO distribution is also known as the Lomax distribution. 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 α 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 μ 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 ϕ 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 α is the shape parameter of the gamma random variables in the mixture and the parameter p is determined solely by α. First, the parameter p is initialized by assuming that the sample is generated from a gamma distribution with shape parameter α 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 292, values of θ and λ 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 $q1$ and $q3$ denote the estimates of the 25th and 75th percentiles, respectively. Using the formula for the CDF of Weibull distribution, they can be written as

$$1 - \exp(-(q1/\theta)^{\tau}) = 0.25$$
$$1 - \exp(-(q3/\theta)^{\tau}) = 0.75$$

Simplifying and solving these two equations yields the following initial values,

$$\hat{\theta} = \exp\left(\frac{r\log(q1) - \log(q3)}{r - 1}\right), \quad \hat{\tau} = \frac{\log(\log(4))}{\log(q3) - \log(\hat{\theta})}$$

where $r = \log(\log(4))/\log(\log(4/3))$. 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 8.3. 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 8.3 Parameter Initialization for Predefined Distributions

Distribution	Parameter	Name for INIT Option	Default Initial Value
BURR	θ	theta	$\sqrt{\frac{m_2 m_3}{2m_3 - 3m_1 m_2}}$
	α	alpha	$1 + \frac{m_3}{2m_3 - 3m_1 m_2}$
	γ	gamma	2
EXP	θ	theta	m_1
GAMMA	θ	theta	m_1/α
	α	alpha	$\frac{3-d+\sqrt{(d-3)^2+24d}}{12d}$
GPD	θ	theta	$m_1 m_2/(2(m_2 - m_1^2))$
	ξ	xi	$(m_2 - 2m_1^2)/(2(m_2 - m_1^2))$
IGAUSS	θ	theta	m_1
	α	alpha	$m_1^2/(m_2 - m_1^2)$
LOGN	μ	mu	$2\log(m1) - \log(m2)/2$
	σ	sigma	$\sqrt{\log(m2) - 2\log(m1)}$
PARETO	θ	theta	$m_1 m_2/(m_2 - 2m_1^2)$
	α	alpha	$2(m_2 - m_1^2)/(m_2 - 2m_1^2)$
TWEEDIE	μ	mu	m_1
	ϕ	phi	$(m_2 - m_1^2)/m_1^p$
	p	p	$(\alpha + 2)/(\alpha + 1)$ where $\alpha = \frac{3-d+\sqrt{(d-3)^2+24d}}{12d}$
STWEEDIE	θ	theta	$(m_2 - m_1^2)(p-1)/m_1$
	λ	lambda	$m_1^2/(\alpha(m_2 - m_1^2)(p-1))$
	p	p	$(\alpha + 2)/(\alpha + 1)$ where $\alpha = \frac{3-d+\sqrt{(d-3)^2+24d}}{12d}$
WEIBULL	θ	theta	$\exp\left(\frac{r\log(q1)-\log(q3)}{r-1}\right)$
	τ	tau	$\log(\log(4))/(\log(q3) - \log(\hat{\theta}))$

Notes:
1. m_k denotes the kth raw moment.
2. $d = \log(m_1) - (\sum \log(y_i))/n$
3. $q1$ and $q3$ 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 HPSEVERITY 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 HPSEVERITY 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 HPSEVERITY 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 HPSEVERITY also checks whether each observation satisfies the definitions of the specified censoring and truncation effects. If you specify left-truncation, then PROC HPSEVERITY ignores observations where $Y \leq T^l$, because such observations are not observable by definition. Similarly, if you specify right-truncation, then PROC HPSEVERITY ignores observations where $Y > T^r$. If you specify left-censoring, then PROC HPSEVERITY 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 HPSEVERITY 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 HPSEVERITY assumes that observation to be uncensored. If all the observations in a data set 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 302) and how the empirical distribution function (EDF) is estimated (see the section "Empirical Distribution Function Estimation Methods" on page 320).

Probability of Observability

For left-truncated data, PROC HPSEVERITY 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 HPSEVERITY can use the information about the number of those events.

In particular, for each left-truncated observation, PROC HPSEVERITY 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 303) and an unconditional estimate of the empirical distribution function (see the section "EDF Estimates and Truncation" on page 325).

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 325. In such cases, PROC HPSEVERITY uses conditional estimates of the CDF for computational or visual comparison to the EDF estimates.

Let $t^l_{\min} = \min_i\{t^l_i\}$ be the smallest value of the left-truncation threshold (t^l_i is the left-truncation threshold for observation i) and $t^r_{\max} = \max_i\{t^r_i\}$ be the largest value of the right-truncation threshold (t^r_i 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

$$\hat{F}^c(y) = \frac{\hat{F}(y) - \hat{F}(t^l_{\min})}{\hat{F}(t^r_{\max}) - \hat{F}(t^l_{\min})}$$

  If an observation is left-truncated but not right-truncated, then

$$\hat{F}^c(y) = \frac{\hat{F}(y) - \hat{F}(t^l_{\min})}{1 - \hat{F}(t^l_{\min})}$$

  If an observation is right-truncated but not left-truncated, then

$$\hat{F}^c(y) = \frac{\hat{F}(y)}{\hat{F}(t^r_{\max})}$$

- 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

$$\hat{F}^c(y) = \frac{\hat{F}(y)}{\hat{F}(t^r_{\max})}$$

If you specify regression effects, then $\hat{F}(y)$, $\hat{F}(t^l_{\min})$, and $\hat{F}(t^r_{\max})$ are all computed from a mixture distribution, as described in the section "CDF and PDF Estimates with Regression Effects" on page 309.

Parameter Estimation Method

If you do not specify a custom objective function by specifying programming statements and the OBJECTIVE= option in the PROC HPSEVERITY statement, then PROC HPSEVERITY 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. If you specify a custom objective function, then PROC HPSEVERITY uses a nonlinear optimization algorithm to estimate the parameters of each model that minimize the value of your specified objective function. For more information, see the section "Custom Objective Functions" on page 359.

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 Θ. Let Y denote the random response variable, and let y denote its value recorded in an observation in the input data set. 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 $t^l = \tau^l$, where τ^l is the smallest value in the support of the distribution; so $F(t^l) = 0$. If there is no right-truncation, then $t^r = \tau_h$, where τ_h 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 $c^l = \tau_h$; so $F(c^l) = 1$. If there is no right-censoring, then $c^r = \tau^l$; 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

$$l_{E_t} = \Pr(Y = y | t^l < Y \le t^r) = \frac{f_\Theta(y)}{F_\Theta(t^r) - F_\Theta(t^l)}$$

- C is the set of censored observations that are not truncated. The likelihood of an observation C is

$$l_C = \Pr(c^r < Y \le c^l) = F_\Theta(c^l) - F_\Theta(c^r)$$

- 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:

$$L = \prod_E f_\Theta(y) \prod_{E_t} \frac{f_\Theta(y)}{F_\Theta(t^r) - F_\Theta(t^l)} \prod_C F_\Theta(c^l) - F_\Theta(c^r) \prod_{C_t} \frac{F_\Theta(c^l) - F_\Theta(c^r)}{F_\Theta(t^r) - F_\Theta(t^l)}$$

The maximum likelihood procedure used by PROC HPSEVERITY 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*_LOWERBOUNDS and *dist*_UPPERBOUNDS subroutines. For more information, see the section "Defining a Severity Distribution Model with the FCMP Procedure" on page 334. Some aspects of the optimization process can be controlled by using the NLOPTIONS statement.

Probability of Observability and Likelihood

If you specify the probability of observability for the left-truncation, then PROC HPSEVERITY 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 302, the likelihood of the data is as follows:

$$\begin{aligned} L = & \prod_E f_\Theta(y) \prod_{E_t, t^l = \tau^l} \frac{f_\Theta(y)}{F_\Theta(t^r)} \prod_{E_t, t^l > \tau^l} \frac{f_\Theta(y)}{F_\Theta(t^r)} F_\Theta(t^l)^{\frac{1-p}{p}} \\ & \prod_C F_\Theta(c^l) - F_\Theta(c^r) \prod_{C_t, t^l = \tau^l} \frac{F_\Theta(c^l) - F_\Theta(c^r)}{F_\Theta(t^r)} \prod_{C_t, t^l > \tau^l} \frac{F_\Theta(c^l) - F_\Theta(c^r)}{F_\Theta(t^r)} F_\Theta(t^l)^{\frac{1-p}{p}} \end{aligned}$$

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 $t^l = \tau^l$) is not affected.

If you specify a custom objective function, then PROC HPSEVERITY 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 HPSEVERITY 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 $\boldsymbol{\theta}$, then the theory states that as $N \rightarrow \infty$, the distribution of $\hat{\boldsymbol{\theta}}$, the estimate of $\boldsymbol{\theta}$, converges to a normal distribution with mean $\boldsymbol{\theta}$ and covariance $\hat{\mathbf{C}}$ such that $\mathbf{I}(\boldsymbol{\theta}) \cdot \hat{\mathbf{C}} \rightarrow 1$, where $\mathbf{I}(\boldsymbol{\theta}) = -E\left[\nabla^2 \log(L(\boldsymbol{\theta}))\right]$ is the information matrix for the likelihood of the data, $L(\boldsymbol{\theta})$. The covariance estimate is obtained by using the inverse of the information matrix.

In particular, if $\mathbf{G} = \nabla^2(-\log(L(\boldsymbol{\theta})))$ denotes the Hessian matrix of the negative of log likelihood, then the covariance estimate is computed as

$$\hat{\mathbf{C}} = \frac{N}{d}\mathbf{G}^{-1}$$

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 θ_i is computed as the square root of the ith 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 $\mathbf{G}$ is computed as $\mathbf{G} = \nabla^2 \log(U(\boldsymbol{\theta}))$, 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 HPSEVERITY 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= or INSTORE= option	You can use either the INEST= data set or the INSTORE= item store, but not both.
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 334.

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=,

INSTORE=, 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 305.

- The method that uses the INEST= data set or INSTORE= item store takes second precedence. If the INEST= data set or INSTORE= item store 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, the INEST= data set, or the INSTORE= item store, but the distribution model defines a PARMINIT subroutine, then PROC HPSEVERITY 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 305.

PARMINIT-Based Parameter Initialization Method and Distributed Data

If you specify a distributed mode of execution for the procedure, then the input data are distributed across the computational nodes. For more information about the distributed computing model, see the section "Distributed and Multithreaded Computation" on page 331. If the PARMINIT subroutine is used for initializing the distribution parameters, then PROC HPSEVERITY 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. If you distribute the data on the appliance before you run the procedure (alongside-the-database model), then you should try to make the distribution as random as possible in order to increase the chances of computing good initial values. If you specify a data set that is not distributed before you run the procedure, then PROC HPSEVERITY distributes the data for you by sending the first observation to the first node, the second observation to the second node, and so on. If the order of observations is random, then this method ensures random distribution of data across the computational nodes.

Estimating Regression Effects

The HPSEVERITY 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 β_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

$$Y \sim \mathcal{F}(\Theta)$$

where $\mathcal{F}$ is the distribution of Y with parameters Θ. This model is usually referred to as the error model. The regression effects are modeled by extending the error model to the following form:

$$Y \sim \exp(\sum_{j=1}^{k} \beta_j x_j) \cdot \mathcal{F}(\Theta)$$

Under this model, the distribution of Y is valid and belongs to the same parametric family as $\mathcal{F}$ if and only if $\mathcal{F}$ has a scale parameter. Let θ denote the scale parameter and Ω denote the set of nonscale distribution parameters of $\mathcal{F}$. Then the model can be rewritten as

$$Y \sim \mathcal{F}(\theta, \Omega)$$

such that θ is modeled by the regression effects as

$$\theta = \theta_0 \cdot \exp(\sum_{j=1}^{k} \beta_j x_j)$$

where θ_0 is the *base* value of the scale parameter. Thus, the scale regression model consists of the following parameters: θ_0, Ω, 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 HPSEVERITY 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 set, then the regression coefficient is added as a variable to that data set. The value of the offset variable in OUTEST= data set is equal to 1 for the estimates row (_TYPE_='EST') and is equal to a 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 θ_0 and β_j parameters for all j.

  You can specify the initial value for θ_0 by using either the INEST= data set, the INSTORE= item store, 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 θ_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 $\log(\theta_0)$.

  You can use only the INEST= data set or the INSTORE= item store, but not both, to specify the initial values for β_j. The requirements for each option are as follows:

  – If you use the INEST= data set, then it must contain nonmissing initial values for all the regressors that you specify in the SCALEMODEL statement. The only missing value that is allowed is the special missing value .R, which indicates that the regressor is linearly dependent on other regressors. If you specify .R for a regressor for one distribution in a BY group, you must specify it the same way for all the distributions in that BY group.

    Note that you cannot specify INEST= data set if the regression model contains effects that have CLASS variables or interaction effects.

  – The parameter estimates in the INSTORE= item store are used to initialize the parameters of a model if the item store contains a model specification that matches the model specification in the current PROC HPSEVERITY step according to the following rules:

    * The distribution name and the number and names of the distribution parameters must match.
    * The model in the item store must include a scale regression model whose regression parameters match as follows:
      · If the regression model in the item store does not contain any redundant parameters, then at least one regression parameter must match. Initial values of the parameters that match are set equal to the estimates that are read from the item store, and initial values of the other regression parameters are set equal to the default value of 0.001.
      · If the regression model in the item store contains any redundant parameters, then all the regression parameters must match, and the initial values of all parameters are set equal to the estimates that are read from the item store.

    Note that a regression parameter is defined by the variables that form the underlying regression effect and by the levels of the CLASS variables if the effect contains any CLASS variables.

- If you do not specify valid initial values for θ_0 or β_j parameters for all j, then PROC HPSEVERITY initializes those parameters by using the following method:

  Let a random variable Y be distributed as $\mathcal{F}(\theta, \Omega)$, where θ is the scale parameter. By the definition of the scale parameter, a random variable $W = Y/\theta$ is distributed as $\mathcal{G}(\Omega)$ such that $\mathcal{G}(\Omega) = \mathcal{F}(1, \Omega)$. Given a random error term e that is generated from a distribution $\mathcal{G}(\Omega)$, 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 θ with the regression effects yields

  $$\log(y) = \log(\theta_0) + \sum_{j=1}^{k} \beta_j x_j + \log(e)$$

  PROC HPSEVERITY makes use of the preceding relationship to initialize parameters of a regression model with distribution *dist* as follows:

  1. The following linear regression problem is solved to obtain initial estimates of β_0 and β_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 β_0 is later used to initialize the value of θ_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 a special missing value of .R in the OUTEST= data set and in any displayed output.

  2. Let s_0 denote the initial value of the scale parameter.

     If the distribution model of *dist* does not contain the *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 *dist*_PARMINIT subroutine. For more information, see the section "Defining a Severity Distribution Model with the FCMP Procedure" on page 334. 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 jth regression effect in the ith input observation. These w_i values are used to compute the input arguments for the *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 θ_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 set), 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 θ_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 HPSEVERITY 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 kth mixture component, and d_k denote the weight associated with kth mixture component.

Let $f(y; s_k, \hat{\Omega})$ and $F(y; s_k, \hat{\Omega})$ denote the PDF and CDF, respectively, of the kth 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 HPSEVERITY uses the $F^*(y)$ values to compute the EDF-based statistics of fit and to create the OUTCDF= data set and the CDF plots. The PDF estimates that it plots in the PDF plots are the $f^*(y)$ values.

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 jth 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 set; 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 sets.

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 kth 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 HPSEVERITY 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 $q << N$, 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 jth observation in the sample $(j = 1, \ldots, r)$, such that

$1 \leq l_j \leq N$, then the scale of kth 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 $r << N$, the RANDOM method is significantly faster than the FULL method.

Levelization of Classification Variables

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. You can determine the sort order of the levels by specifying the ORDER= option in the CLASS statement. You can also control the sort order separately for each variable in the CLASS statement.

Consider the data on nine observations in Table 8.4. 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 8.4 apply two different formats to the variable X.

Table 8.4 Example Data for Levelization

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

By default, levelization of the variables groups the observations by the formatted value of the variable, except for numerical variables for which no explicit format is provided. Those numerical variables are sorted by their internal value. The levelization of the four columns in Table 8.4 leads to the level assignment in Table 8.5.

Table 8.5 Values and Levels

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

You can specify the sort order for the levels of CLASS variables 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 class levels that have no explicit format by their BEST12. formatted values, you can specify the BEST12. format explicitly for the CLASS variables.

Table 8.6 shows how values of the ORDER= option are interpreted.

Table 8.6 Interpretation of Values of ORDER= Option

Value of ORDER=	Levels Sorted By
DATA	Order of appearance in the input data set
FORMATTED	External formatted value, except for numeric variables that have no explicit format, which are sorted by their unformatted (internal) value
FREQ	Descending frequency count (levels that have the most observations come first in the order)
INTERNAL	Unformatted value
FREQDATA	Order of descending frequency count, and within counts by order of appearance in the input data set when counts are tied
FREQFORMATTED	Order of descending frequency count, and within counts by formatted value when counts are tied
FREQINTERNAL	Order of descending frequency count, and within counts by unformatted (internal) value when counts are tied

For FORMATTED, FREQFORMATTED, FREQINTERNAL, and INTERNAL values, the sort order is machine-dependent. For more information about sort order, see the chapter about the SORT procedure in

the *Base SAS Procedures Guide* and the discussion of BY-group processing in *SAS Language Reference: Concepts*.

When you specify the MISSING option 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 8.7 displays the results of levelizing the values in Table 8.4 when the MISSING option is in effect.

Table 8.7 Values and Levels with the MISSING Option

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

When you do not specify the MISSING option, it is important to understand the implications of missing values for your statistical analysis. When PROC HPSEVERITY levelizes the CLASS variables, any observations for which a CLASS variable has a missing value are excluded from the analysis. This is true regardless of whether the variable is used to form the statistical model. For example, consider the case in which some observations contain missing values for variable A but the records for these observations are otherwise complete with respect to all other variables in the model. The analysis results that come from the following statements do not include any observations for which variable A contains missing values, even though A is not specified in the SCALEMODEL statement:

```
class A B;
scalemodel B x B*x;
```

You can request PROC HPSEVERITY to print the "Descriptive Statistics" table, which 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 set contains missing values—to ensure that no observations are unintentionally excluded from the analysis.

Specification and Parameterization of Model Effects

PROC HPSEVERITY supports formation of regression effects in the SCALEMODEL statement. A *regression effect* is formed from one or more regressor variables according to effect construction rules (*parameterization*). Each regression effect forms one element of $\mathbf{X}$ in the linear model structure $\mathbf{X}\boldsymbol{\beta}$ that affects the scale parameter. The SCALEMODEL statement in conjunction with the CLASS statement supports a rich set of effects. In order to correctly interpret the results, you need to understand the specification and parameterization of effects that are discussed in this section.

Effects are specified by a special notation that uses 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. For more information, see the section "Levelization of Classification Variables" on page 311. A regressor variable that is not declared in the CLASS statement is assumed to be *continuous*.

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 314.

If you specify the CLASS statement, then PROC HPSEVERITY supports a general linear model (GLM) parameterization and a reference parameterization for the classification variables. The GLM parameterization is the default. For more information, see the sections "GLM Parameterization of Classification Variables and Effects" on page 316 and "Reference Parameterization" on page 320.

Effect Operators

Table 8.8 summarizes the operators that are available for selecting and constructing effects. These operators are discussed in the following sections.

Table 8.8 Available Effect Operators

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

Bar and At Sign Operators

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

```
scalemodel A B C   A*B A*C B*C   A*B*C;

scalemodel A|B|C;
```

When you use the bar (|), 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 from Searle (1971, p. 390).

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

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

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

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

```
scalemodel A|B|C@2;
```

The preceding example is equivalent to the following SCALEMODEL statement:

```
scalemodel A B C   A*B A*C B*C;
```

More examples of using the bar and at sign operators follow:

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

NOTE: The preceding examples assume the following CLASS statement specification:

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

Colon, Dash, and Double Dash Operators

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

```
scalemodel X1 X2 X3 X4 X5 X6 X7 X8 X9;

scalemodel X1-X9;

scalemodel X:;
```

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

```
scalemodel X:;
```

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

```
scalemodel 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 SCALEMODEL statement with a double dash, as follows, then all three variables are selected:

```
scalemodel 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:

```
scalemodel A--C;
```

GLM Parameterization of Classification Variables and Effects

Table 8.9 shows the types of effects that are available in the HPSEVERITY procedure; they are discussed in more detail in the following sections. Let A, B, and C represent classification variables, and let X and Z represent continuous variables.

Table 8.9 Available Types of Effects

Effect	Example	Description
Singleton continuous	X Z	Continuous variables
Polynomial continuous	X*Z	Interaction of continuous variables
Main	A B	CLASS variables
Interaction	A*B	Crossing of CLASS variables
Nested	A(B)	Main effect A nested within CLASS effect B
Continuous-by-class	X*A	Crossing of continuous and CLASS variables
Continuous-nesting-class	X(A)	Continuous variable X nested within CLASS variable A
General	X*Z*A(B)	Combinations of different types of effects

Continuous Effects

Continuous variables or polynomial terms that involve them can be included in the model as continuous effects. An effect that contains a single continuous variable is referred to as a *singleton continuous* effect, and an effect that contains an interaction of only continuous variables is referred to as a *polynomial continuous* effect. The actual values of such terms are included as columns of the relevant model matrices. You can use the bar operator along with a continuous variable 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 given 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 8.10 is an example where β_0 denotes the intercept and A and B are classification variables that have two and three levels, respectively.

Table 8.10 Example of Main Effects

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

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

Interaction Effects

Often a regression model includes interaction (crossed) effects to account for how the effect of a variable changes along with the values of other variables. In 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, as illustrated in Table 8.11.

Table 8.11 Example of Interaction Effects

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

In the matrix in Table 8.11, 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 regression model contains many interaction effects, you might be able to code them more parsimoniously by using the bar operator (|). The bar operator generates all possible interaction effects. For example, A | B | C expands to A B A*B C A*C B*C A*B*C. To eliminate higher-order interaction effects, use the at sign (@) in conjunction with the bar operator. For example, A | B | C | D@2 expands to A B A*B C A*C B*C D A*D B*D C*D.

Nested Effects

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

```
scalemodel A B(A);

scalemodel A A*B;
```

The nesting operator in PROC HPSEVERITY is more of a notational convenience than an operation that is distinct from crossing. Nested effects are usually 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, as illustrated in Table 8.12.

Table 8.12 Example of Nested Effects

Data		I	A		B(A)					
A	B	β_0	A1	A2	B1A1	B2A1	B3A1	B1A2	B2A2	B3A2
1	1	1	1	0	1	0	0	0	0	0
1	2	1	1	0	0	1	0	0	0	0
1	3	1	1	0	0	0	1	0	0	0
2	1	1	0	1	0	0	0	1	0	0
2	2	1	0	1	0	0	0	0	1	0
2	3	1	0	1	0	0	0	0	0	1

Continuous-Nesting-Class Effects

When a continuous variable nests or crosses with a classification variable, the design columns are constructed by multiplying the continuous values into the design columns for the classification effect, as illustrated in Table 8.13.

Table 8.13 Example of Continuous-Nesting-Class Effects

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

Continuous-by-Class Effects

Continuous-by-class effects generate the same design columns as continuous-nesting-class effects. Table 8.14 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 8.13.

Table 8.14 Example of Continuous-by-Class Effects

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

General Effects

An example that combines all the effects is X1*X2*A*B*C(D E). The continuous list comes first, followed by the crossed list, followed by the nested list in parentheses. PROC HPSEVERITY might rename effects 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 a model includes four effects—A, B, C, and D—each of which has two levels, 1 and 2. Assume the CLASS statement is

```
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

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

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

Reference Parameterization

Classification variables can be represented in the reference parameterization. Consider the classification variable A that has four values, 1, 2, 5, and 7. The reference parameterization generates three columns (one less than the number of variable levels). The columns indicate group membership of the nonreference levels. For the reference level, the three dummy variables have a value of 0. If the reference level is 7 (REF='7'), the design columns for variable A are as shown in Table 8.15.

Table 8.15 Reference Coding

	Design Matrix		
A	**A1**	**A2**	**A5**
1	1	0	0
2	0	1	0
5	0	0	1
7	0	0	0

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

Empirical Distribution Function Estimation Methods

The empirical distribution function (EDF) is a nonparametric estimate of the cumulative distribution function (CDF) of the distribution. PROC HPSEVERITY 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 set 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 been computed by using the entire input data set. PROC HPSEVERITY 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 HPSEVERITY 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 $(y_i, t_i^l, t_i^r, c_i^r, c_i^l), i = 1, \ldots, N$, where y_i is the value of the response variable, t_i^l is the value of the left-truncation threshold, t_i^r is the value of the right-truncation threshold, c_i^r is the value of the right-censoring limit, and c_i^l is the value of the left-censoring limit.

If an observation is not left-truncated, then $t_i^l = \tau^l$, where τ^l is the smallest value in the support of the distribution; so $F(t_i^l) = 0$. If an observation is not right-truncated, then $t_i^r = \tau_h$, where τ_h is the largest value in the support of the distribution; so $F(t_i^r) = 1$. If an observation is not right-censored, then $c_i^r = \tau^l$; so $F(c_i^r) = 0$. If an observation is not left-censored, then $c_i^l = \tau_h$; so $F(c_i^l) = 1$.

Let w_i denote the weight associated with ith 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 HPSEVERITY 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 323.

The method descriptions assume that all observations are either uncensored or right-censored; that is, each observation is of the form $(y_i, t_i^l, t_i^r, \tau^l, \tau_h)$ or $(y_i, t_i^l, t_i^r, c_i^r, \tau_h)$.

If all observations are either uncensored or left-censored, then each observation is of the form $(y_i, t_i^l, t_i^r, \tau_l, c_i^l)$. It is converted to an observation $(-y_i, -t_i^r, -t_i^l, -c_i^l, \tau_h)$; 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. After the EDF estimates are computed, the observations are transformed back to the original form and EDF estimates are adjusted such $F_n(y_i) = 1 - F_n(-y_i-)$, where $F_n(-y_i-)$ denotes the EDF estimate of the value slightly less than the transformed value $-y_i$.

Further, a set of uncensored or right-censored observations can be converted to a set of observations of the form $(y_i, t_i^l, t_i^r, \delta_i)$, where δ_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, $\delta_i = I[Y \le C^r]$ and $y_i = \min(y_i, c_i^r)$.

Given this notation, the EDF is estimated as

$$F_n(y) = \begin{cases} 0 & \text{if } y < y^{(1)} \\ \hat{F}_n(y^{(k)}) & \text{if } y^{(k)} \le y < y^{(k+1)}, k = 1, \ldots, N-1 \\ \hat{F}_n(y^{(N)}) & \text{if } y^{(N)} \le y \end{cases}$$

where $y^{(k)}$ denotes the kth-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 HPSEVERITY choose an appropriate method by using the EMPIRICALCDF= option in the PROC HPSEVERITY 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:

$$\hat{F}_n(y_i) = \frac{1}{N}\sum_{j=1}^{N} w_j \cdot I[y_j \le y_i]$$

If you do not specify any censoring or truncation information, then this method is chosen. If you explicitly specify this method, then PROC HPSEVERITY 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:

$$\hat{\sigma}_n(y_i) = \sqrt{\hat{F}_n(y_i)(1 - \hat{F}_n(y_i))/N}$$

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 HPSEVERITY 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

$$\hat{F}_n(y_i) = 1 - \prod_{\tau \le y_i} \left(1 - \frac{n(\tau)}{R_n(\tau)}\right)$$

where $n(\tau)$ and $R_n(\tau)$ are defined as follows:

- $n(\tau) = \sum_{k=1}^{N} w_k \cdot I[y_k = \tau \text{ and } \tau \le t_k^r \text{ and } \delta_k = 1]$, which is the number of uncensored observations ($\delta_k = 1$) for which the response variable value is equal to τ and τ is observable according to the right-truncation threshold of that observation ($\tau \le t_k^r$).

- $R_n(\tau) = \sum_{k=1}^{N} w_k \cdot I[y_k \geq \tau > t_k^l]$, which is the size (cardinality) of the risk set at τ. The term *risk set* has its origins in survival analysis; it contains the events that are at risk of failure at a given time, τ. In other words, it contains the events that have survived up to time τ and might fail at or after τ. For PROC HPSEVERITY, *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):

$$\hat{\sigma}_n(y_i) = \sqrt{(1 - \hat{F}_n(y_i))^2 \cdot \sum_{\tau \leq y_i} \left(\frac{n(\tau)}{R_n(\tau)(R_n(\tau) - n(\tau))} \right)}$$

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

$$\hat{F}_n(y_i) = 1 - \prod_{\tau \leq y_i} \left(1 - \frac{n(\tau)}{R_n(\tau)} \cdot I[R_n(\tau) \geq cN^{\alpha}] \right)$$

where the definitions of $n(\tau)$ and $R_n(\tau)$ are identical to those used for the KAPLANMEIER method described previously.

You can specify the values of c and α 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 α, 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 $I[R_n(\tau) \geq cN^{\alpha}]$ is replaced by $I[R_n(\tau) \geq L]$ in the definition.

The standard error of $\hat{F}_n(y_i)$ is computed by using Greenwood's formula (Greenwood 1926):

$$\hat{\sigma}_n(y_i) = \sqrt{(1 - \hat{F}_n(y_i))^2 \cdot \sum_{\tau \leq y_i} \left(\frac{n(\tau)}{R_n(\tau)(R_n(\tau) - n(\tau))} \cdot I[R_n(\tau) \geq cN^{\alpha}] \right)}$$

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 HPSEVERITY 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 321. 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 $q_j \leq p_j$ 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+1}). 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 HPSEVERITY 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 HPSEVERITY 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 ϵ. 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 ϵ by specifying the EPS= suboption of the EDF=TURNBULL option in the PROC HPSEVERITY 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 HPSEVERITY 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 HPSEVERITY statement, then PROC HPSEVERITY 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 HPSEVERITY 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 HPSEVERITY declares these values to be zero if they are less than or equal to a threshold δ. You can control the value of δ by specifying the ZEROPROB= suboption of the EDF=TURNBULL option in the PROC HPSEVERITY 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 estimator essentially attempts to maximize the likelihood L, which depends on the parameters s_j $(j = 1, \ldots, M)$. Let $\boldsymbol{s} = \{s_j\}$ denote the set of these parameters. If $\mathbf{G}(\boldsymbol{s}) = \nabla^2(-\log(L(\boldsymbol{s})))$ denotes the Hessian matrix of the negative of log likelihood, then the variance-covariance matrix of $\boldsymbol{s}$ is estimated as $\hat{\mathbf{C}}(\boldsymbol{s}) = \mathbf{G}^{-1}(\boldsymbol{s})$. Given this matrix, the standard error of $F_n(y)$ is computed as

$$\sigma_n(y) = \sqrt{\sum_{k=1}^{j}\left(\hat{C}_{kk} + 2\cdot\sum_{l=1}^{k-1}\hat{C}_{kl}\right)}, \text{ if } p_j < y < q_{j+1}, 1 \le j \le M-1$$

The standard error is undefined outside of these intervals.

EDF Estimates and Truncation

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, $\hat{F}_n(y) = \Pr(Y \le y | \tau_G < Y \le \tau_H)$, 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, τ_G denotes the smallest left-truncation threshold with a nonzero cumulative probability, and τ_H denotes the largest right-truncation threshold with a nonzero cumulative probability. Formally, $\tau_G = \inf\{s : G(s) > 0\}$ and $\tau_H = \sup\{s : H(s) > 0\}$. For computational purposes, PROC HPSEVERITY estimates τ_G and τ_H by $t^l_{\min}$ and $t^r_{\max}$, respectively, defined as

$$t^l_{\min} = \min\{t^l_k : 1 \le k \le N\}$$
$$t^r_{\max} = \max\{t^r_k : 1 \le k \le N\}$$

These estimates of $t^l_{\min}$ and $t^r_{\max}$ are used to compute the conditional estimates of the CDF as described in the section "Truncation and Conditional CDF Estimates" on page 301.

If you specify left-truncation *with* the probability of observability p, then PROC HPSEVERITY 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^l_i, an observation j is added with *weight* $w_j = (1-p)/p$ and $y_j = t^l_j$. 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 $n(\tau) = \sum_{k=1}^{N^a} w_k I[y_k = \tau \text{ and } \tau \leq t_k^r \text{ and } \delta_k = 1]$, 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=TURNBULL 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 334 and "Predefined Utility Functions" on page 346.

PROC HPSEVERITY 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 kth 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 HPSEVERITY 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 sets, 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 set 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 302. 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 ith observation. If w_i^o denotes the original, unnormalized weight of the ith 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 $r_i = \sum_{j=1}^{N} w_j I[y_j \le y_i]$ 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 309.
- 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 301.

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:

$$\text{KS} = \sup_y |F_n(y) - F(y)|$$

If the STANDARD method is used to compute the EDF, then the following formula is used:

$$D^+ = \max_i \left(\frac{r_i}{W} - Z_i\right)$$
$$D^- = \max_i \left(Z_i - \frac{r_{i-1}}{W}\right)$$
$$\text{KS} = \sqrt{W}\, \max(D^+, D^-) + \frac{0.19}{\sqrt{W}}$$

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:

$$D^+ = \max_i (F_n(Z_i) - Z_i), \text{ if } F_n(Z_i) \geq Z_i$$
$$D^- = \max_i (Z_i - F_n(Z_i)), \text{ if } F_n(Z_i) < Z_i$$
$$\text{KS} = \sqrt{W}\, \max(D^+, D^-) + \frac{0.19}{\sqrt{W}}$$

AD 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:

$$\text{AD} = N \int_{-\infty}^{\infty} \frac{(F_n(y) - F(y))^2}{F(y)(1 - F(y))} dF(y)$$

If the STANDARD method is used to compute the EDF, then the following formula is used:

$$\text{AD} = -W - \frac{1}{W} \sum_{i=1}^{N_u} W_i \left[(2r_i - 1)\log(Z_i) + (2W + 1 - 2r_i)\log(1 - Z_i)\right]$$

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

$$\text{AD} = N \int_{-\infty}^{\infty} \frac{(F_n(z) - z)^2}{z(1-z)} dz$$

The computation formula can then be derived from the approximation,

$$\begin{aligned}
\text{AD} &= N \sum_{i=1}^{K+1} \int_{Z_{i-1}}^{Z_i} \frac{(F_n(z) - z)^2}{z(1-z)} dz \\
&= N \sum_{i=1}^{K+1} \int_{Z_{i-1}}^{Z_i} \frac{(F_n(Z_{i-1}) + S_i(z - Z_{i-1}) - z)^2}{z(1-z)} dz \\
&= N \sum_{i=1}^{K+1} \int_{Z_{i-1}}^{Z_i} \frac{(P_i - Q_i z)^2}{z(1-z)} dz
\end{aligned}$$

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,

$$\begin{aligned}
\text{AD} = &- N(Z_1 + \log(1 - Z_1) + \log(Z_K) + (1 - Z_K)) \\
&+ N \sum_{i=2}^{K} \left[P_i^2 A_i - (Q_i - P_i)^2 B_i - Q_i^2 C_i \right]
\end{aligned}$$

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

$$\begin{aligned}
\text{AD} = &- N(1 + \log(1 - Z_1) + \log(Z_K)) \\
&+ N \sum_{i=2}^{K} \left[F_n(Z_{i-1})^2 A_i - (1 - F_n(Z_{i-1}))^2 B_i \right]
\end{aligned}$$

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:

$$\text{CvM} = N \int_{-\infty}^{\infty} (F_n(y) - F(y))^2 dF(y)$$

If the STANDARD method is used to compute the EDF, then the following formula is used:

$$\text{CvM} = \frac{1}{12W} + \sum_{i=1}^{N_u} W_i \left(Z_i - \frac{(2r_i - 1)}{2W} \right)^2$$

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

$$\text{CvM} = N \int_{-\infty}^{\infty} (F_n(z) - z)^2 dz$$

The computation formula can then be derived from the following approximation,

$$\begin{aligned}
\text{CvM} &= N \sum_{i=1}^{K+1} \int_{Z_{i-1}}^{Z_i} (F_n(z) - z)^2 dz \\
&= N \sum_{i=1}^{K+1} \int_{Z_{i-1}}^{Z_i} (F_n(Z_{i-1}) + S_i(z - Z_{i-1}) - z)^2 dz \\
&= N \sum_{i=1}^{K+1} \int_{Z_{i-1}}^{Z_i} (P_i - Q_i z)^2 dz
\end{aligned}$$

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 $Z_0 = 0$, $Z_{K+1} = 1$, and $F_n(0) = 0$ yields the following computation formula,

$$\text{CvM} = N \frac{Z_1^3}{3} + N \sum_{i=2}^{K+1} \left[P_i^2 A_i - P_i Q_i B_i - \frac{Q_i^2}{3} C_i \right]$$

where $A_i = Z_i - Z_{i-1}$, $B_i = Z_i^2 - Z_{i-1}^2$, and $C_i = Z_i^3 - Z_{i-1}^3$.

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).

Distributed and Multithreaded Computation

PROC HPSEVERITY makes an attempt to use all the computational resources that you specify in the PERFORMANCE statement in order to complete the assigned tasks as fast as possible. This section describes the distributed and multithreading computing methods that PROC HPSEVERITY uses.

Distributed Computing

Distributed computing refers to the organization of computation work into multiple tasks that are processed on different nodes; a node is one of the machines that constitute the grid. The number of nodes that PROC HPSEVERITY uses is determined by the distributed processing execution mode. If you specify the client-data (or local-data) mode of execution, then the number of nodes is determined by the NODES= option in the PERFORMANCE statement. If you are using the alongside-the-database mode of execution, then PROC HPSEVERITY determines the number of nodes internally by using the information that is associated with the DATA= data set and the grid information that you specify either in the PERFORMANCE statement or in the grid environment variables. For more information about distributed processing modes, see the section "Processing Modes" on page 6.

In the client-data model, PROC HPSEVERITY distributes the input data across the number of nodes that you specify by sending the first observation to the first node, the second observation to the second node, and so on.

In the alongside-the-database model, PROC HPSEVERITY uses the existing distributed organization of the data. You do not need to specify the NODES= option.

The number of nodes that are used for distributed computing is displayed in the "Performance Information" table, which is part of the default output.

Multithreading

Threading refers to the organization of computational work into multiple tasks (processing units that can be scheduled by the operating system). A task is associated with a thread. Multithreading refers to the concurrent execution of threads. When multithreading is possible, you can achieve more substantial performance gains than you can with sequential (single-threaded) execution.

The number of threads the HPSEVERITY procedure spawns is determined by the number of CPUs on a machine. You can control the number of CPUs in the following ways:

- You can use the CPUCOUNT= SAS system option to specify the CPU count. For example, if you specify the following statement, then PROC HPSEVERITY schedules threads as if it were executing on a system that had four CPUs, regardless of the actual CPU count:

  ```
  options cpucount=4;
  ```

  You can use this specification only in single-machine mode, and it does not take effect if the THREADS system option is turned off.

  The default value of the CPUCOUNT= system option might not equal the number of all the logical CPU cores available on your machine, such as those available because of hyperthreading. To allow PROC HPSEVERITY to use all the logical cores in single-machine mode, specify the following OPTIONS statement:

  ```
  options cpucount=actual;
  ```

- You can specify the NTHREADS= option in the PERFORMANCE statement. This specification overrides the THREADS and CPUCOUNT= system options. Specify NTHREADS=1 to force single-threaded execution.

If you do not specify the NTHREADS= option and the THREADS system option is turned on, then the number of threads that are used in distributed mode is equal to the total number of logical CPU cores available on each node of the grid, and the number of threads used in single-machine mode is determined by the CPUCOUNT= system option.

If you do not specify the NTHREADS= option and the THREADS system option is turned off, then only one thread of execution is used in both single-machine and distributed modes.

The number of threads per machine is displayed in the "Performance Information" table, which is part of the default output.

Performance improvement is not always guaranteed when you use more threads, for several reasons: the increased cost of communication and synchronization among threads might offset the reduced cost of computation, the hyperthreading feature of the processor might not be very efficient for floating-point computations, and other applications might be running on the machine.

Combining the Power of Distributed and Multithreading Computing

The HPSEVERITY procedure combines the powers of distributed and multithreading paradigms by using a data-parallel model. In particular, the distributed tasks are defined by dividing the data among multiple nodes, and within one node, the multithreading tasks are defined by further dividing the local data among the threads. For example, if the input data set has 10,000 observations and you are running on a grid that has five nodes, then each node processes 2,000 observations (this assumes that if you specify an alongside-the-database model, then you have equally and randomly divided the input data among the nodes). Further, if each node has eight CPUs, then 250 observations are associated with each thread within the node. All computations that require access to the data are then distributed and multithreaded.

Note that in single-machine mode (see the section "Processing Modes" on page 6), only multithreading is available.

When you specify more than one candidate distribution model, for some tasks PROC HPSEVERITY exploits the independence among models by processing multiple models in parallel on a single node such that each model is assigned to one of the threads executing in parallel. When a thread finishes processing the assigned model, it starts processing the next unprocessed model, if one exists.

The computations that take advantage of the distributed and multithreaded model include the following:

- Validation and preparation of data: In this stage, the observations in the input data set are validated and transformed, if necessary. The summary statistics of the data are prepared. Because each observation is independent, the computations can be distributed among nodes and among threads within nodes without significant communication overhead.
- Initialization of distribution parameters: In this stage, the parallelism is achieved by initializing multiple models in parallel. The only computational step that is not fully parallelized in this release is the step of computing empirical distribution function (EDF) estimates, which are required when PROC HPSEVERITY needs to invoke a distribution's PARMINIT subroutine to initialize distribution parameters. The EDF estimation step is not amenable to full-fledged parallelism because it requires sequential access to sorted data, especially when the loss variable is modified by truncation effects. When the data are distributed across nodes, the EDF computations take place on local data and the PARMINIT function is invoked on the local data by using the local EDF estimates. The initial values

that are supplied to the nonlinear optimizer are computed by averaging the local estimates of the distribution parameters that are returned by the PARMINIT functions on each node.

- Initialization of regression parameters (if you specify the SCALEMODEL statement): In this stage, if you do not specify initial values for the regression parameters by using the INEST= data set or the INSTORE= item store, then PROC HPSEVERITY initializes those parameters by solving a linear regression problem $\log(y) = \beta_0 + \sum_{i=1}^{k} \beta_j x_j$. For more information, see the section "Parameter Initialization for Regression Models" on page 307. The most computationally intensive step is the formation of the crossproducts matrix. PROC HPSEVERITY exploits the parallelism by observing the fact that the contribution to the crossproducts matrix due to one observation is independent from the contribution due to another observation. Each node computes the contribution of its local data to each entry of the crossproducts matrix. Within each node, each thread computes the contribution of its chunk of data to each entry of the crossproducts matrix. On each node, the contributions from all the threads are added up to form the contribution due to all of the local data. The partial crossproducts matrices are then gathered from all nodes on a central node, which sums them up to form the final crossproducts matrix.

- Optimization: In this stage, the nonlinear optimizer iterates over the parameter space in search of the optimal set of parameters. In each iteration, it evaluates the objective function along with the gradient and Hessian of the objective function, if needed by the optimization method. Within one iteration, for the current estimates of the parameters, each observation's contribution to the objective function, gradient, and Hessian is independent of another observation. This enables PROC HPSEVERITY to fully exploit the distributed and multithreaded paradigms to efficiently parallelize each iteration of the algorithm.

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 <*distribution-name*>_<*keyword*>, where *distribution-name* is the identifying short name of the distribution and *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 8.16.

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 HPSEVERITY 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 HPSEVERITY 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 ∞. 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 ∞). 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 HPSEVERITY 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 HPSEVERITY 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 HPSEVERITY, 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 Global Statements: Reference*. For more information about the CMPLIB= system option, see *SAS System Options: Reference*.

Each predefined distribution mentioned in the section "Predefined Distributions" on page 290 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 8.2. You do not need to use the CMPLIB= option in order to use the predefined distributions with PROC HPSEVERITY. However, if you need to use the functions or subroutines of the

predefined distributions in SAS statements other than the PROC HPSEVERITY 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 8.16 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 8.16 List of Functions and Subroutines That Define a Distribution Model

Name	Type	Required	Expected to Return
*dist*_CDF	Function	YES[1]	Cumulative distribution function value
*dist*_CDFGRADIENT	Subroutine	NO	Gradient of the CDF
*dist*_CDFHESSIAN	Subroutine	NO	Hessian of the CDF
*dist*_CONSTANTPARM	Subroutine	NO	Constant parameters
*dist*_DESCRIPTION	Function	NO	Description of the distribution
*dist*_LOGCDF	Function	YES[1]	Log of cumulative distribution function value
*dist*_LOGCDFGRADIENT	Subroutine	NO	Gradient of the LOGCDF
*dist*_LOGCDFHESSIAN	Subroutine	NO	Hessian of the LOGCDF
*dist*_LOGPDF	Function	YES[2]	Log of probability density function value
*dist*_LOGPDFGRADIENT	Subroutine	NO	Gradient of the LOGPDF
*dist*_LOGPDFHESSIAN	Subroutine	NO	Hessian of the LOGPDF
*dist*_LOGSDF	Function	NO	Log of survival function value
*dist*_LOGSDFGRADIENT	Subroutine	NO	Gradient of the LOGSDF
*dist*_LOGSDFHESSIAN	Subroutine	NO	Hessian of the LOGSDF
*dist*_LOWERBOUNDS	Subroutine	NO	Lower bounds on parameters
*dist*_PARMINIT	Subroutine	NO	Initial values for parameters
*dist*_PDF	Function	YES[2]	Probability density function value
*dist*_PDFGRADIENT	Subroutine	NO	Gradient of the PDF
*dist*_PDFHESSIAN	Subroutine	NO	Hessian of the PDF
*dist*_QUANTILE	Function	NO	Quantile for a given CDF value
*dist*_SCALETRANSFORM	Function	NO	Type of relationship between the first distribution parameter and the scale parameter
*dist*_SDF	Function	NO	Survival function value
*dist*_SDFGRADIENT	Subroutine	NO	Gradient of the SDF
*dist*_SDFHESSIAN	Subroutine	NO	Hessian of the SDF
*dist*_UPPERBOUNDS	Subroutine	NO	Upper bounds on parameters

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

  p1 Numeric value of the first parameter

  p2 Numeric value of the second parameter

  ...

  pm Numeric value of the mth 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:

$$F(x; p_1, p_2, \ldots, p_m) = F(\frac{x}{p_1}; 1, p_2, \ldots, p_m)$$

If the distribution has a log-transformed scale parameter, then the following equation must be satisfied:

$$F(x; p_1, p_2, \ldots, p_m) = F(\frac{x}{\exp(p_1)}; 0, p_2, \ldots, p_m)$$

Here is a sample structure of the function for a distribution named 'FOO':

```
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 HPSEVERITY. 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 HPSEVERITY. You can estimate them inside the *dist*_PARMINIT subroutine. Once initialized, the parameters remain constant in the context of PROC HPSEVERITY; that is, they retain their initial value. PROC HPSEVERITY 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 *k*th 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:

```
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 *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.

*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':

```
function FOO_DESCRIPTION() $48;
    length desc $48;
    desc = "A model for a continuous distribution named foo";
    return (desc);
endsub;
```

There is no restriction on the length of the description (the length of 48 used in the previous example is for illustration purposes only). However, if the length is greater than 256, then only the first 256 characters appear in the displayed output and in the _DESCRIPTION_ variable of the OUTMODELINFO= data set. Hence, the recommended length of the description is less than or equal to 256.

*dist*_**LOG***core*

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

  p1 Numeric value of the first parameter

  p2 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':

```
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 given distribution, then the HPSEVERITY 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 HPSEVERITY procedure assumes that $l_i < p_i$ (strict inequality). If a missing value is returned for some parameter, then the HPSEVERITY 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*:

  p1 Output argument that returns the lower bound on the first parameter. You must specify this in the OUTARGS statement inside the subroutine's definition.

  p2 Output argument that returns the lower bound on the second parameter. You must specify this in the OUTARGS statement inside the subroutine's definition.

  ...

  p*m* 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*:

  dim Input numeric value that contains the dimension of the x, nx, and F array arguments.

  x{*} Input numeric array of dimension *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 *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 *dim*. Each F[i] contains the EDF estimate for x[i]. This estimate is computed by the HPSEVERITY procedure based on the options that you specify in the LOSS statement and the EMPIRICALCDF= option.

  Ftype 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 326.

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.

...

p*m* 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':

```
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

  ...

  p*m* 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:

$$f(x; p_1, p_2, \ldots, p_m) = \frac{1}{p_1} f(\frac{x}{p_1}; 1, p_2, \ldots, p_m)$$

If the distribution has a log-transformed scale parameter, then the following equation must be satisfied:

$$f(x; p_1, p_2, \ldots, p_m) = \frac{1}{\exp(p_1)} f(\frac{x}{\exp(p_1)}; 0, p_2, \ldots, p_m)$$

Here is a sample structure of the function for a distribution named 'FOO':

```
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*:

  cdf Numeric value of the cumulative distribution function (CDF) for which the quantile should be evaluated

  p1 Numeric value of the first parameter

  p2 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':

```
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 μ and shape σ, 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 θ instead of the parameter μ such that $\mu = \log(\theta)$. The parameter θ would then be a scale parameter. However, if you want to specify the distribution in terms of μ and σ (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 θ and σ, you can simply define the distribution with μ as the first parameter and specify that it is the logarithm of the scale parameter.

- *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':

```
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*: $m + 1$, 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
p1	Numeric value of the first parameter

p2 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:

$$S(x; p_1, p_2, \ldots, p_m) = S(\frac{x}{p_1}; 1, p_2, \ldots, p_m)$$

If the distribution has a log-transformed scale parameter, then the following equation must be satisfied:

$$S(x; p_1, p_2, \ldots, p_m) = S(\frac{x}{\exp(p_1)}; 0, p_2, \ldots, p_m)$$

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 given distribution, then the HPSEVERITY 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 HPSEVERITY procedure assumes that $p_i < u_i$ (strict inequality). If a missing value is returned for some parameter, then the HPSEVERITY procedure assumes that there is no upper bound for that parameter (equivalent to an upper bound of ∞).

- *Type*: Subroutine
- *Required*: NO
- *Number of arguments*: *m*, where *m* is the number of distribution parameters
- *Sequence and type of arguments*:

p1 Output argument that returns the upper bound on the first parameter. You must specify this in the OUTARGS statement inside the subroutine's definition.

p2 Output argument that returns the upper bound on the second parameter. You must specify this in the OUTARGS statement inside the subroutine's definition.
...

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_core*__GRADIENT__

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
- *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
p1	Numeric value of the first parameter
p2	Numeric value of the second parameter
	...
p*m*	Numeric value of the *m*th parameter
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: $\frac{\partial h}{\partial p_1}\ \frac{\partial h}{\partial p_2} \cdots \frac{\partial h}{\partial p_m}$

- *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':

```
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_core*__HESSIAN__

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
p1	Numeric value of the first parameter
p2	Numeric value of the second parameter
	...
p*m*	Numeric value of the *m*th parameter
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: $\frac{\partial^2 h}{\partial p_1^2} \mid \frac{\partial^2 h}{\partial p_1 \partial p_2} \; \frac{\partial^2 h}{\partial p_2^2} \mid \cdots \mid \frac{\partial^2 h}{\partial p_1 \partial p_m} \; \frac{\partial^2 h}{\partial p_2 \partial p_m} \cdots \frac{\partial^2 h}{\partial p_m^2}$

- *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':

```
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 HPSEVERITY 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*:

  y — Value of the random variable at which the EDF estimate is desired

  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 326.

- *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 326.

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 326.

- *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 326 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:

$$k_{\text{opt}} = \frac{k_1^2}{k_2} \left(\frac{\log(k_1)}{2\log(m_1) - \log(k_1)} \right)^{2 - 2\log(k_1)/\log(m_1)}$$

6. Set the cutoff point equal to $x_{(k_{\text{opt}}+1)}$.

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 jth bootstrap sample is computed as

$$\text{MSE}_j(m, k) = (M_j(m, k) - 2(\gamma_j(m, k))^2)^2$$

where $M_j(m,k)$ is a control variate proposed by Danielsson et al. 2001,

$$M_j(m,k) = \frac{1}{k}\sum_{i=1}^{k}\left(\log(x^{j,m}_{(m-i+1)}) - \log(x^{j,m}_{(m-k)})\right)^2$$

and $\gamma_j(m,k)$ is the Hill's estimator of the tail index (Hill 1975),

$$\gamma_j(m,k) = \frac{1}{k}\sum_{i=1}^{k}\log(x^{j,m}_{(m-i+1)}) - \log(x^{j,m}_{(m-k)})$$

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 100pth 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 326.

- *Return value*: The 100pth 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 $\lfloor x \rfloor$ denote the greatest integer less than or equal to x. Define $g = \lfloor p(n+1) \rfloor$ 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 $\hat{\pi}_p = x[n]$.

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 ϵ 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 ϵ 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 *k*th element in the array (raw{*k*}) contains the *k*th 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 the given array of numeric values in an ascending or descending order.

- *Type*: Subroutine
- *Signature*: SVRTUTIL_SORT(n, x{*}, flag)
- *Argument description*:

  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.

Scoring Functions

Scoring refers to the act of evaluating a distribution function, such as LOGPDF, SDF, or QUANTILE, on an observation by using the fitted parameter estimates of that distribution. You can do scoring in a DATA step by using the OUTEST= data set that you create with PROC HPSEVERITY. However, that approach requires some cumbersome programming. In order to simplify the scoring process, you can specify that PROC HPSEVERITY create scoring functions for each fitted distribution.

As an example, assume that you have fitted the Pareto distribution by using PROC HPSEVERITY and that it converges. Further assume that you want to use the fitted distribution to evaluate the probability of observing a loss value greater than some set of regulatory limits {L} that are encoded in a data set. You can simplify this scoring process as follows. First, in the PROC HPSEVERITY step that fits your distributions, you create the scoring functions library by specifying the OUTSCORELIB statement as illustrated in the following steps:

```
proc hpseverity data=input;
   loss lossclaim;
   dist pareto;
   outscorelib outlib=sasuser.fitdist;
run;
```

Upon successful completion, if the Pareto distribution model has converged, then the Sasuser.Fitdist library contains the *SEV_SDF* scoring function in addition to other scoring functions, such as *SEV_PDF*, *SEV_LOGPDF*, and so on. Further, PROC HPSEVERITY also sets the CMPLIB system option to include the Sasuser.Fitdist library. If the set of limits {L} is recorded in the variable Limit in the scoring data set Work.Limits, then you can submit the following DATA step to compute the probability of seeing a loss greater than each limit:

```
data prob;
   set work.limits;
   exceedance_probability = sev_sdf(limit);
run;
```

Without the use of scoring functions, you can still perform this scoring task, but the DATA step that you need to write to accomplish it becomes more complicated and less flexible. For example, you would need to read the parameter estimates from some output created by PROC HPSEVERITY. To do that, you would need to know the parameter names, which are different for different distributions; this in turn would require you to write a specific DATA step for each distribution or to write a SAS macro. With the use of scoring functions, you can accomplish that task much more easily.

If you fit multiple distributions, then you can specify the COMMONPACKAGE option in the OUTSCORELIB statement as follows:

```
proc hpseverity data=input;
   loss lossclaim;
   dist exp pareto weibull;
   outscorelib outlib=sasuser.fitdist commonpackage;
run;
```

The preceding step creates scoring functions such as *SEV_SDF_Exp*, *SEV_SDF_Pareto*, and *SEV_SDF_Weibull*. You can use them to compare the probabilities of exceeding the limit for different distributions by using the following DATA step:

```
data prob;
   set work.limits;
   exceedance_exp = sev_sdf_exp(limit);
   exceedance_pareto = sev_sdf_pareto(limit);
   exceedance_weibull = sev_sdf_weibull(limit);
run;
```

Formal Description

PROC HPSEVERITY creates a scoring function for each distribution function. A distribution function is defined as any function named *dist_suffix*, where *dist* is the name of a distribution that you specify in the DIST statement and the function's signature is identical to the signature of the required distribution function such as *dist*_CDF or *dist*_LOGCDF. For example, for the function 'FOO_BAR' to be a distribution 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':

```
function FOO_BAR(y, P1, P2);
   /* Code to compute BAR by using y, P1, and P2 */
   R = <computed BAR>;
   return (R);
endsub;
```

For more information about the signature that defines a distribution function, see the description of the *dist*_CDF function in the section "Defining a Severity Distribution Model with the FCMP Procedure" on page 334.

The name and package of the scoring function of a distribution function depend on whether you specify the COMMONPACKAGE option in the OUTSCORELIB statement.

When you do not specify the COMMONPACKAGE option, the scoring function that corresponds to the distribution function *dist_suffix* is named *SEV_suffix*, where *SEV_* is the standard prefix of all scoring functions. The scoring function is created in a package named *dist*. Each scoring function accepts only one argument, the value of the loss variable, and returns the same value as the value returned by the corresponding

distribution function for the final estimates of the distribution's parameters. For example, for the preceding 'FOO_BAR' distribution function, the scoring function named 'SEV_BAR' is created in the package named 'FOO' and 'SEV_BAR' has the following signature:

```
function SEV_BAR(y);
   /* returns value of FOO_BAR for the supplied value
      of y and fitted values of P1, P2 */
endsub;
```

If you specify the COMMONPACKAGE option in the OUTSCORELIB statement, then the scoring function that corresponds to the distribution function *dist_suffix* is named *SEV_suffix_dist*, where *SEV_* is the standard prefix of all scoring functions. The scoring function is created in a package named *sevfit*. For example, for the preceding 'FOO_BAR' distribution function, if you specify the COMMONPACKAGE option, the scoring function named 'SEV_BAR_FOO' is created in the *sevfit* package and 'SEV_BAR_FOO' has the following signature:

```
function SEV_BAR_FOO(y);
   /* returns value of FOO_BAR for the supplied value
      of y and fitted values of P1, P2 */
endsub;
```

Scoring Functions for the Scale Regression Model

If you use the SCALEMODEL statement to specify a scale regression model, then PROC HPSEVERITY generates the scoring functions when you specify only singleton continuous effects. If you specify interaction or classification effects, then scoring functions are not generated.

For a scale regression model, the estimate of the scale parameter or the log-transformed scale parameter of the distribution depends on the values of the regressors. So PROC HPSEVERITY creates a scoring function that has the following signature, where *x{*}* represents the array of regressors:

```
function SEV_BAR(y, x{*});
   /* returns value of FOO_BAR for the supplied value of x and fitted values of P1, P2 */
endsub;
```

As an illustration of using this form, assume that you submit the following PROC HPSEVERITY step to create the scoring library Sasuser.Scalescore:

```
proc hpseverity data=input;
   loss lossclaim;
   scalemodel x1-x3;
   dist pareto;
   outscorelib outlib=sasuser.scalescore;
run;
```

Your scoring data set must contain all the regressors that you specify in the SCALEMODEL statement. You can submit the following DATA step to score observations by using the scale regression model:

```
data prob;
   array regvals{*} x1-x3;
   set work.limits;
   exceedance_probability = sev_sdf(limit, regvals);
run;
```

PROC HPSEVERITY creates two utility functions, SEV_NUMREG and SEV_REGNAME, in the OUTLIB= library that return the number of regressors and name of a given regressor, respectively. They are described in detail in the next section. These utility functions are useful when you do not have easy access to the regressor names in the SCALEMODEL statement. You can use the utility functions as follows:

```
data prob;
   array regvals{10} _temporary_;
   set work.limits;
   do i = 1 to sev_numreg();
      regvals(i) = input(vvaluex(sev_regname(i)), best12.);
   end;
   exceedance_probability = sev_sdf(limit, regvals);
run;
```

The dimension of the regressor values array that you supply to the scoring function must be equal to $K + L$, where K is the number of regressors that you specify in the SCALEMODEL statement irrespective of whether PROC HPSEVERITY deems any of those regressors to be redundant. L is 1 if you specify an OFFSET= variable in the SCALEMODEL statement, and 0 otherwise.

Utility Functions and Subroutines in the OUTLIB= Library

In addition to creating the scoring functions for all distribution functions, PROC HPSEVERITY creates the following utility functions and subroutines in the OUTLIB= library.

SEV_NUMPARM | SEV_NUMPARM_*dist*
is a function that returns the number of distribution parameters and has the following signature:

- *Type*: Function
- *Number of arguments*: 0
- *Sequence and type of arguments*: Not applicable
- *Return value*: Numeric value that contains the number of distribution parameters

If you do not specify the COMMONPACKAGE option in the OUTSCORELIB statement, then a function named SEV_NUMPARM is created in the package of each distribution. Here is a sample structure of the code that PROC HPSEVERITY uses to define the function:

```
function SEV_NUMPARM();
    n = <number of distribution parameters>;
    return (n);
endsub;
```

If you specify the COMMONPACKAGE option in the OUTSCORELIB statement, then for each distribution *dist*, the function named SEV_NUMPARM_*dist* is created in the *sevfit* package. SEV_NUMPARM_*dist* has the same structure as the SEV_NUMPARM function that is described previously.

SEV_PARMEST | SEV_PARMEST_*dist*
is a subroutine that returns the estimate and standard error of a specified distribution parameter and has the following signature:

- *Type*: Subroutine
- *Number of arguments*: 3
- *Sequence and type of arguments*:

  index specifies the numeric value of the index of the distribution parameter for which you want the information. The value of *index* must be in the interval [1,*m*], where *m* is the number of parameters in the distribution to which this subroutine belongs.

  est specifies the output argument that returns the estimate of the requested parameter.

  stderr specifies the output argument that returns the standard error of the requested parameter.

- *Return value*: Estimate and standard error of the requested distribution parameter that are returned in the output arguments *est* and *stderr*, respectively

If you do not specify the COMMONPACKAGE option in the OUTSCORELIB statement, then a subroutine named SEV_PARMEST is created in the package of each distribution. Here is a sample structure of the code that PROC HPSEVERITY uses to define the subroutine:

```
subroutine SEV_PARMEST(index, est, stderr);
    outargs est, stderr;
    est = <value of the estimate for the distribution parameter
           at position 'index'>;
    stderr =  <value of the standard error for distribution parameter
               at position 'index'>;
endsub;
```

If you specify the COMMONPACKAGE option in the OUTSCORELIB statement, then for each distribution *dist*, the subroutine named SEV_PARMEST_*dist* is created in the *sevfit* package. SEV_PARMEST_*dist* has the same structure as the SEV_PARMEST subroutine that is described previously.

If you use the SCALEMODEL statement to specify a scale regression model, and if you specify only singleton continuous effects, then for *index*=1, the returned estimates are of θ_0, the base value of the scale parameter, or $\log(\theta_0)$ if the distribution has a log-scale parameter. For more information about θ_0, see the section "Estimating Regression Effects" on page 305.

SEV_PARMNAME | SEV_PARMNAME_*dist*

is a function that returns the name of a specified distribution parameter and has the following signature:

- *Type*: Function
- *Number of arguments*: 1
- *Sequence and type of arguments*:

  index specifies the numeric value of the index of the distribution parameter for which you want the information. The value of *index* must be in the interval [1,*m*], where *m* is the number of parameters in the distribution to which this function belongs.

- *Return value*: Character value that contains the name of the distribution parameter that appears at the position *index* in the distribution's definition

If you do not specify the COMMONPACKAGE option in the OUTSCORELIB statement, then a function named SEV_PARMNAME is created in the package of each distribution.

Here is a sample structure of the code that PROC HPSEVERITY uses to define the function:

```
function SEV_PARMNAME(index) $32;
    name = <name of the distribution parameter at position 'index'>;
    return (name);
endsub;
```

If you specify the COMMONPACKAGE option in the OUTSCORELIB statement, then for each distribution *dist*, a function named SEV_PARMNAME_*dist* is created in the *sevfit* package. SEV_PARMNAME_*dist* has the same structure as the SEV_PARMNAME function that is described previously.

If you use the SCALEMODEL statement to specify a scale regression model, and if you specify only singleton continuous effects, then the following helper functions and subroutines are also created in the OUTLIB= library.

SEV_NUMREG
is a function that returns the number of regressors and has the following signature:

- *Type*: Function
- *Number of arguments*: 0
- *Sequence and type of arguments*: Not applicable
- *Return value*: Numeric value that contains the number of regressors that you specify in the SCALEMODEL statement. If you specify an OFFSET= variable in the SCALEMODEL statement, then the returned value is equal to 1 plus the number of regressors that you specify in the SCALEMODEL statement.

Here is a sample structure of the code that PROC HPSEVERITY uses to define the function:

```
function SEV_NUMREG();
    m = <number of regressors>;
    if (<offset variable is specified>) then m = m + 1;
    return (m);
endsub;
```

This function does not depend on any distribution, so it is always created in the *sevfit* package.

SEV_REGEST | SEV_REGEST_*dist*
is a subroutine that returns the estimate and standard error of a specified regression parameter and has the following signature:

- *Type*: Subroutine
- *Number of arguments*: 3
- *Sequence and type of arguments*:

index specifies the numeric value of the index of the regression parameter for which you want the information. The value of *index* must be in the interval [1,*K*], where *K* is the number of regressors as returned by the SEV_NUMREG function. If you specify an OFFSET= variable in the SCALEMODEL statement, then an *index* value of *K* corresponds to the offset variable.

est specifies the output argument that returns the estimate of the requested regression parameter.

stderr specifies the output argument that returns the standard error of the requested regression parameter.

- *Return value*: Estimate and standard error of the requested regression parameter that are returned in the output arguments *est* and *stderr*, respectively

If you do not specify the COMMONPACKAGE option in the OUTSCORELIB statement, then a subroutine named SEV_REGEST is created in the package of each distribution. Here is a sample structure of the code that PROC HPSEVERITY uses to define the subroutine:

```
subroutine SEV_REGEST(index, est, stderr);
    outargs est, stderr;
    est = <value of the estimate for the regression parameter
           at position 'index'>;
    stderr =  <value of the standard error for regression parameter
              at position 'index'>;
endsub;
```

If you specify the COMMONPACKAGE option in the OUTSCORELIB statement, then for each distribution *dist*, the subroutine named SEV_REGEST_*dist* is created in the *sevfit* package. SEV_REGEST_*dist* has the same structure as the SEV_REGEST subroutine that is described previously.

If the regressor that corresponds to the specified *index* value is a redundant regressor, the returned values of both *est* and *stderr* are equal to the special missing value of .R. If you specify an OFFSET= variable in the SCALEMODEL statement and if the *index* value corresponds to the offset variable—that is, it is equal to the value that the SEV_NUMREG function returns—then the returned value of *est* is equal to 1 and the returned value of *stderr* is equal to the special missing value of .F.

SEV_REGNAME

is a function that returns the name of a specified regressor and has the following signature:

- *Type*: Function
- *Number of arguments*: 1
- *Sequence and type of arguments*:

index specifies the numeric value of the index of the regressor for which you want the name. The value of *index* must be in the interval [1,*K*], where *K* is the number of regressors as returned by the SEV_NUMREG function. If you specify an OFFSET= variable in the SCALEMODEL statement, then an *index* value of *K* corresponds to the offset variable.

- *Return value*: Character value that contains the name of the regressor that appears at the position *index* in the SCALEMODEL statement. If you specify an OFFSET= variable in the SCALEMODEL statement, then for an *index* value of *K*, the returned value contains the name of the offset variable.

Here is a sample structure of the code that PROC HPSEVERITY uses to define the function:

```
function SEV_REGNAME(index) $32;
    name = <name of regressor at position 'index'>;
    return (name);
endsub;
```

This function does not depend on any distribution, so it is always created in the *sevfit* package.

Custom Objective Functions

You can use a series of programming statements that use variables in the DATA= data set to assign a value to an objective function symbol. You must specify the objective function symbol by using the OBJECTIVE= option in the PROC HPSEVERITY statement.

The objective function can be programmed such that it is applicable to any distribution that is used in the model. For that purpose, PROC HPSEVERITY recognizes the following *keyword* functions in the programming statements:

PDF(x) returns the probability density function (PDF) of a distribution evaluated at the current value of a data set variable x.

CDF(x) returns the cumulative distribution function (CDF) of a distribution evaluated at the current value of a data set variable x.

SDF(x) returns the survival distribution function (SDF) of a distribution evaluated at the current value of a data set variable x.

LOGPDF(x) returns the natural logarithm of the PDF of a distribution evaluated at the current value of a data set variable x.

LOGCDF(x) returns the natural logarithm of the CDF of a distribution evaluated at the current value of a data set variable x.

LOGSDF(x) returns the natural logarithm of the SDF of a distribution evaluated at the current value of a data set variable x.

EDF(x) returns the empirical distribution function (EDF) estimate evaluated at the current value of a data set variable x. Internally, PROC HPSEVERITY computes the estimate using the SVRTUTIL_EDF function as described in the section "Predefined Utility Functions" on page 346. The EDF estimate that is required by the SVRTUTIL_EDF function is computed by using the response variable values in the current BY group or in the entire input data set if you do not specify the BY statement.

EMPLIMMOMENT(k, u)
returns the empirical limited moment of order *k* evaluated at the current value of a data

set variable u that represents the upper limit of the limited moment. The order *k* can also be a data set variable. Internally, PROC HPSEVERITY computes the moment using the SVRTUTIL_EMPLIMMOMENT function as described in the section "Predefined Utility Functions" on page 346. The EDF estimate that is required by the SVRTUTIL_EMPLIMMOMENT function is computed by using the response variable values in the current BY group or in the entire input data set if you do not specify the BY statement.

LIMMOMENT(k, u)
returns the limited moment of order *k* evaluated at the current value of a data set variable u that represents the upper limit of the limited moment. The order *k* can be a data set variable or a constant. Internally, for each candidate distribution, PROC HPSEVERITY computes the moment using the LIMMOMENT function as described in the section "Predefined Utility Functions" on page 346.

All the preceding functions are right-hand side functions. They act as placeholders for distribution-specific functions, with the exception of _EDF_ and _EMPLIMMOMENT_ functions.

As an example, let the data set Work.Test contain a response variable Y and a left-truncation threshold variable T. The following statements use the values in this data set to fit a model with distribution D such that the parameters of the model minimize the value of the objective function symbol MYOBJ:

```
options cmplib=(work.mydist);
proc hpseverity data=work.test objective=myobj;
    loss y / lt=t;

    myobj = -_LOGPDF_(y);
    if (not(missing(t))) then
        myobj = myobj + log(1-_CDF_(t));

    dist d;
run;
```

The symbol MYOBJ is designated as an objective function symbol by using the OBJECTIVE= option in the PROC HPSEVERITY statement. The response variable Y and left-truncation variable T are specified in the LOSS statement. The distribution D is specified in the DIST statement. The remaining statements constitute a program that computes the value of the MYOBJ symbol.

Let the distribution D have parameters P1 and P2. In order to estimate the model for this distribution, PROC HPSEVERITY internally converts the generic program to the following program specific to distribution D:

```
myobj = -D_LOGPDF(y, p1, p2);
if (not(missing(t))) then
    myobj = myobj + log(1-D_CDF(t, p1, p2));
```

Note that the generic keyword functions _LOGPDF_ and _CDF_ have been replaced with distribution-specific functions D_LOGPDF and D_CDF, respectively, with appropriate distribution parameters. The D_LOGPDF and D_CDF functions must have been defined previously and are assumed to be available in the Work.Mydist library that you specify in the CMPLIB= option.

The program is executed for each observation in Work.Test to compute the value of MYOBJ by using the values of variables Y and T in that observation and internally computed values of the model parameters P1 and P2. The values of MYOBJ are then added over all the observations of the data set or over all the observations of the current BY group if you specify the BY statement. The resulting aggregate value is the

value of the objective function, and it is supplied to the optimizer. If the optimizer requires derivatives of the objective function, then PROC HPSEVERITY automatically differentiates MYOBJ with respect to the parameters P1 and P2. The optimizer iterates over various combinations of the values of parameters P1 and P2, each time computing a new value of the objective function and the needed derivatives of it, until it finds a combination that minimizes the objective function.

Note the following points when you define your own program to compute the custom objective function:

- The value of the objective function is always minimized by PROC HPSEVERITY. If you want to maximize the value of a certain objective, then add a statement that assigns the negated value of the maximization objective to the objective function symbol that you specify in the OBJECTIVE= option. Minimization of the negated objective is equivalent to the maximization of the original objective.
- The contributions of individual observations are always added to compute the overall objective function in a given iteration of the optimizer. If you specify the WEIGHT statement, then the contribution of each observation is weighted by multiplying it with the normalized value of the weight variable for that observation.
- If you are fitting multiple distributions in one PROC HPSEVERITY step and use any of the keyword functions in your program, then it is recommended that you do not explicitly use the parameters of any of the specified distributions in your programming statements.
- If you use a specific keyword function in your programming statements, then the corresponding distribution functions must be defined in a library that you specify in the CMPLIB= system option or in Sashelp.Svrtdist, the predefined functions library. In the preceding example, it is assumed that the functions D_LOGPDF and D_CDF are defined in the Work.Mydist library that is specified in the CMPLIB= option.
- You can use most DATA step statements and functions in your program. The DATA step file and the data set I/O statements (for example, INPUT, FILE, SET, and MERGE) are not available. However, some functionality of the PUT statement is supported. For more information, see the section "PROC FCMP and DATA Step Differences" in *Base SAS Procedures Guide*. In addition to the differences listed in that section, the following differences exist:
  - Only numeric-valued variables can be used in PROC HPSEVERITY programming statements. This restriction also implies that you cannot use SAS functions or call routines that require character-valued arguments, unless you pass those arguments as constant (literal) strings or characters.
  - You cannot use functions that create lagged versions of a variable in PROC HPSEVERITY programming statements. If you need lagged versions, then you can use a DATA step prior to the PROC HPSEVERITY step to add those versions to the input data set.
- When coding your programming statements, avoid defining variables that begin with an underscore (_), because they might conflict with internal variables created by PROC HPSEVERITY.

Custom Objective Functions and Regression Effects

If you specify regression effects by using the SCALEMODEL statement, then PROC HPSEVERITY automatically adds a statement prior to your programming statements to compute the value of the scale parameter or the log-transformed scale parameter of the distribution using the values of the regression variables and internally created regression parameters. For example, if your specification of the SCALEMODEL statement results in three regression effects x1, x2, and x3, then for a model that contains the distribution D with scale parameter S, PROC HPSEVERITY adds a statement that is equivalent to the following statement to the beginning of your program:

```
S = _SEVTHETA0 * exp(_SEVBETA1 * x1 + _SEVBETA2 * x2 + _SEVBETA3 * x3);
```

If a model contains a distribution D1 with a log-transformed scale parameter M, PROC HPSEVERITY adds a statement that is equivalent to the following statement to the beginning of your program:

```
M = _SEVTHETA0 + _SEVBETA1 * x1 + _SEVBETA2 * x2 + _SEVBETA3 * x3;
```

The _SEVTHETA0, _SEVBETA1, _SEVBETA2, and _SEVBETA3 are the internal regression parameters associated with the intercept and the regression effects x1, x2, and x3, respectively.

Since the names of the internal regression parameters start with a prefix _SEV, if you use a variable in your program with a name that begins with _SEV, then PROC HPSEVERITY writes an error message to the SAS log and stops processing.

Input Data Sets

PROC HPSEVERITY accepts DATA= and INEST= data sets as input data sets. This section details the information they are expected to contain.

DATA= Data Set

The DATA= data set is expected to contain the values of the analysis variables that you specify in the LOSS statement and the SCALEMODEL statement.

If you specify the BY statement, then the DATA= data set must contain all the BY variables that you specify in the BY statement and the data set must be sorted by the BY variables unless you specify the NOTSORTED option in the BY statement.

INEST= Data Set

The INEST= data set is expected to contain the initial values of the parameters for the parameter estimation process.

If you specify the SCALEMODEL statement, then you can use the INEST= data set only if the SCALEMODEL statement contains singleton continuous effects.

If you specify the BY statement, then the INEST= data set must contain all the BY variables that you specify in the BY statement. If you do not specify the NOTSORTED option in the BY statement, then the INEST= data set must be sorted by the BY variables. However, it is not required to contain all the BY groups present in the DATA= data set. For the BY groups that are not present in the INEST= data set, the default parameter initialization method is used. If you specify the NOTSORTED option in the BY statement, then the INEST=

data set must contain all the BY groups that are present in the DATA= data set and they must appear in the same order as they appear in the DATA= data set.

In addition to any variables that you specify in the BY statement, the data set must contain the following variables:

MODEL identifying name of the distribution for which the estimates are provided.

TYPE type of the estimate. The value of this variable must be EST for an observation to be valid.

<Parameter 1> . . . <Parameter M>
M variables, named after the parameters of all candidate distributions, that contain initial values of the respective parameters. *M* is the cardinality of the union of parameter name sets from all candidate distributions. In an observation, estimates are read only from variables for parameters that correspond to the distribution that is indicated by the _MODEL_ variable.

If you specify a missing value for some parameters, then default initial values are used unless the parameter is initialized by using the INIT= option in the DIST statement. If you want to use the *dist*_PARMINIT subroutine for initializing the parameters of a model, then you should either not specify the model in the INEST= data set or specify missing values for all the distribution parameters in the INEST= data set and not use the INIT= option in the DIST statement.

If you specify regressors, then the initial value that you provide for the first parameter of each distribution must be the base value of the scale or log-transformed scale parameter. For more information, see the section "Estimating Regression Effects" on page 305.

<Regressor 1> . . . <Regressor K>
If you specify *K* regressors in the SCALEMODEL statement, then the INEST= data set must contain *K* variables that are named for each regressor. The variables contain initial values of the respective regression coefficients. If a regressor is linearly dependent on other regressors for a given BY group, then you can indicate this by providing a special missing value of .R for the respective variable. In a given BY group, if you mark a variable as linearly dependent for one model, then you must mark that variable as linearly dependent for all the models. Similarly, in a given BY group, if you do not mark a variable as linearly dependent for one model, then you must not mark that variable as linearly dependent for all the models.

Output Data Sets

PROC HPSEVERITY writes the OUTCDF=, OUTEST=, OUTMODELINFO=, and OUTSTAT= data sets when requested by their respective options in the PROC HPSEVERITY statement. It also writes the OUT= data set when you specify the OUTPUT statement. The data sets and their contents are described in the following sections.

OUT= Data Set

The OUT= data set 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 set 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 <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 HPSEVERITY uses. For more information about variable names, see the description of the OUTPUT statement.

The OUT= data set also contains the variables that you specify in the COPYVARS= option. If you specify the BY statement and if you want PROC HPSEVERITY to copy the BY variables from the DATA= data set to the OUT= data set, then you must specify them in the COPYVARS= option.

The number of observations in the OUT= data set 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 set is equal to the number of observations in the DATA= data set:

- 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 set is equal to the number of BY groups, which is equal to 1 if you do not specify the BY statement.

OUTCDF= Data Set

The OUTCDF= data set records the estimates of the cumulative distribution function (CDF) of each of the specified model distributions and an estimate of the empirical distribution function (EDF). This data set is created only when you run PROC HPSEVERITY in single-machine mode.

If you specify BY variables, then the data are organized in BY groups and the data set contains variables that you specify in the BY statement. In addition, the data set contains the following variables:

<response variable>
: value of the response variable. The values are sorted. If there are multiple BY groups, the values are sorted within each BY group.

OBSNUM
: observation number in the DATA= data set. This is a sequence number that indicates the order in which the procedure accesses the observation; it does not necessarily reflect the actual observation number in the data set.

EDF
: estimate of the empirical distribution function (EDF). This estimate is computed by using the EMPIRICALCDF= option that you specify in the PROC HPSEVERITY statement.

_EDF_STD
: estimate of the standard error of EDF. This estimate is computed by using a method that is appropriate for the EMPIRICALCDF= option that you specify in the PROC HPSEVERITY statement.

_EDF_LOWER
: estimate of the lower confidence limit of EDF for a pointwise $100(1-\alpha)\%$ confidence interval, where α is the value of the EDFALPHA= option that you specify in the PROC HPSEVERITY statement (default is $\alpha = 0.05$). For an EDF estimate F_n that has standard error σ_n, it is computed as $\text{MAX}(0, F_n - z_{(1-\alpha/2)}\sigma_n)$, where z_p is the pth quantile from the standard normal distribution.

_EDF_UPPER estimate of the upper confidence limit of EDF for a pointwise $100(1-\alpha)\%$ confidence interval, where α is the value of the EDFALPHA= option that you specify in the PROC HPSEVERITY statement (default is $\alpha = 0.05$). For an EDF estimate F_n that has standard error σ_n, it is computed as MIN(1, $F_n + z_{(1-\alpha/2)}\sigma_n$), where z_p is the pth quantile from the standard normal distribution.

<distribution1>_CDF ... <distribution*D*>_CDF
estimate of the cumulative distribution function (CDF) for each of the D candidate distributions, computed by using the final parameter estimates for that distribution. This value is missing if the parameter estimation process does not converge for the given distribution.

If you specify regressor variables, then the reported estimates are from a mixture distribution. For more information, see the section "CDF and PDF Estimates with Regression Effects" on page 309.

If you specify truncation, then the data set contains the following additional variables:

<distribution1>_COND_CDF ... <distribution*D*>_COND_CDF
estimate of the conditional CDF for each of the D candidate distributions, computed by using the final parameter estimates for that distribution. This value is missing if the parameter estimation process does not converge for the distribution. The conditional estimates are computed by using the method that is described in the section "Truncation and Conditional CDF Estimates" on page 301.

OUTEST= Data Set

The OUTEST= data set 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 set contains variables that you specify in the BY statement.

If you do not specify the COVOUT option, then the data set 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 368.

<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 305.

<Regression Effect 1> . . . <Regression Effect K>

If your effect specification in the SCALEMODEL statement results in *K* regression effects, then the OUTEST= data set contains *K* regression variables. The name of each variable is formed by using the name of the effect and the names of 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 partial effect name and integer identifiers for BY groups and CLASS variable levels. The label of the variable is more descriptive than the name of the variable. The variables contain estimates for their respective regression coefficients. If an effect is deemed to be linearly dependent on other effects for a given BY group, then a warning message is written to the SAS log and a special missing value of .R is written in the respective 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.

<Offset Variable>

If you specify an OFFSET= variable in the SCALEMODEL statement, then the OUTEST= data set 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 a special missing value of .F.

If you specify the COVOUT option in the PROC HPSEVERITY statement, then the OUTEST= data set contains additional observations that contain the estimates of the covariance structure. Given the symmetric nature of the covariance structure, only the lower triangular portion is reported. In addition to the variables listed and described previously, the data set 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.

OUTMODELINFO= Data Set

The OUTMODELINFO= data set records the information about each candidate distribution that you specify in the DIST statement. It contains the following variables:

MODEL identifying name of the distribution model. The observation contains information about this distribution.

DEPVAR name of the loss variable.

DESCRIPTION descriptive name of the model. This has a nonmissing value only if the DESCRIPTION function has been defined for this model.

VALID validity of the distribution definition. This has a value of 1 for valid definitions and a value of 0 for invalid definitions. If the definition is invalid, then PROC HPSEVERITY writes the reason for invalidity to the SAS log.

_PARMNAME1 . . . _PARMNAME*M* *M* variables that contain names of parameters of the distribution model, where *M* is the maximum number of parameters across all the specified distribution models. For a given distribution with *m* parameters, values of variables _PARMNAME*j* ($j > m$) are missing.

OUTSTAT= Data Set

The OUTSTAT= data set records statistics of fit and model selection information. If you specify BY variables, then the data are organized in BY groups and the data set contains variables that you specify in the BY statement. The data set contains the following variables:

MODEL identifying name of the distribution model. The observation contains information about this distribution.

NMODELPARM number of parameters in the distribution.

NESTPARM number of estimated parameters. This includes the regression parameters, if you specify any regression effects.

NOBS number of nonmissing observations used for parameter estimation.

STATUS status of the parameter estimation process for this model. The possible values are listed in the section "_STATUS_ Variable Values" on page 368.

SELECTED indicator of the best distribution model. If the value is 1, then this model is the best model for the current BY group according to the specified model selection criterion. This value is missing if the parameter estimation process does not converge for this model.

Neg2LogLike value of the log likelihood, multiplied by –2, that is attained at the end of the parameter estimation process. This value is missing if the parameter estimation process does not converge for this model.

AIC value of the Akaike's information criterion (AIC) that is attained at the end of the parameter estimation process. This value is missing if the parameter estimation process does not converge for this model.

AICC value of the corrected Akaike's information criterion (AICC) that is attained at the end of the parameter estimation process. This value is missing if the parameter estimation process does not converge for this model.

BIC value of the Schwarz Bayesian information criterion (BIC) that is attained at the end of the parameter estimation process. This value is missing if the parameter estimation process does not converge for this model.

KS — value of the Kolmogorov-Smirnov (KS) statistic that is attained at the end of the parameter estimation process. This value is missing if the parameter estimation process does not converge for this model.

AD — value of the Anderson-Darling (AD) statistic that is attained at the end of the parameter estimation process. This value is missing if the parameter estimation process does not converge for this model.

CVM — value of the Cramér–von Mises (CvM) statistic that is attained at the end of the parameter estimation process. This value is missing if the parameter estimation process does not converge for this model.

STATUS Variable Values

The _STATUS_ variable in the OUTEST= and OUTSTAT= data sets 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 set for _TYPE_=EST observations and in the OUTSTAT= data set:

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 set 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 HPSEVERITY procedure optionally produces displayed output by using the Output Delivery System (ODS). All output is controlled by the PRINT= option in the PROC HPSEVERITY statement. Table 8.17 relates the ODS tables to PRINT= options.

Table 8.17 ODS Tables Produced in PROC HPSEVERITY

ODS Table Name	Description	Option
AllFitStatistics	Statistics of fit for all the distribution models	PRINT=ALLFITSTATS
ConvergenceStatus	Convergence status of parameter estimation process	PRINT=CONVSTATUS
DescStats	Descriptive statistics for the response variable	PRINT=DESCSTATS
DistributionInfo	Distribution information	PRINT=DISTINFO
EstimationDetails	Details of the estimation process for all the distribution models	PRINT=ESTIMATIONDETAILS
InitialValues	Initial parameter values and bounds	PRINT=INITIALVALUES
IterationHistory	Optimization iteration history	PRINT=NLOHISTORY
ModelSelection	Model selection summary	PRINT=SELECTION
OptimizationSummary	Optimization summary	PRINT=NLOSUMMARY
ParameterEstimates	Final parameter estimates	PRINT=ESTIMATES
PerformanceInfo	Execution environment information that pertains to the computational performance	Default
RegDescStats	Descriptive statistics for the regression effects that do not contain a CLASS variable	PRINT=DESCSTATS
StatisticsOfFit	Statistics of fit	PRINT=STATISTICS
Timing	Timing information for various computational stages of the procedure	DETAILS (PERFORMANCE statement)
TurnbullSummary	Turnbull EDF estimation summary	PRINT=ALL

If you do not specify the PRINT= option, then by default PROC HPSEVERITY produces ModelSelection, PerformanceInfo, ConvergenceStatus, OptimizationSummary, StatisticsOfFit, and ParameterEstimates ODS tables.

The following describes the content that each table displays:

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 all the models fail to 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 with an asterisk (*) in that statistic's column.

ConvergenceStatus (PRINT=CONVSTATUS)
displays the convergence status of the parameter estimation process.

DescStats (PRINT=DESCSTATS)
displays the descriptive statistics for the response variable.

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.

EstimationDetails (PRINT=ESTIMATIONDETAILS)
displays the comparative details of the estimation process that is used to fit each candidate distribution. If you specify the DETAILS option in the PERFORMANCE statement, then this table contains a column that indicates the time taken to estimate each candidate model.

InitialValues (PRINT=INITIALVALUES)
displays the initial values and bounds used for estimating each model.

IterationHistory (PRINT=NLOHISTORY)
displays the iteration history of the nonlinear optimization process used for estimating the parameters.

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.

OptimizationSummary (PRINT=NLOSUMMARY)
displays the summary of the nonlinear optimization process used for estimating the parameters.

ParameterEstimates (PRINT=ESTIMATES)
displays the final estimates of parameters. The estimates are not displayed for models whose parameter estimation process does not converge.

PerformanceInfo
displays information about the execution mode. For single-machine mode, the table displays the number of threads that are used. For distributed mode, the table displays the grid mode (symmetric or asymmetric), the number of compute nodes, and the number of threads per node. PROC HPSEVERITY produces this table by default.

RegDescStats (PRINT=DESCSTATS)
displays the descriptive statistics for the regression effects in the SCALEMODEL statement that do not contain a CLASS variable.

StatisticsOfFit (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.

Timing (DETAILS option in the PERFORMANCE statement
displays elapsed times (absolute and relative) for the main tasks of the procedure. PROC HPSEVERITY produces this table when you specify the DETAILS option in the PERFORMANCE statement,

TurnbullSummary (PRINT=ALL)
displays the summary of Turnbull's estimation process if Turnbull's method is used for computing EDF estimates. The summary includes whether the nonlinear optimization converged, the number of iterations, the maximum absolute relative error, the maximum absolute reduced gradient, and whether the final estimates are maximum likelihood estimates. This table is produced only if you specify PRINT=ALL and Turnbull's method is used for computing EDF estimates.

ODS Graphics

Statistical procedures use ODS Graphics to create graphs as part of their output. ODS Graphics is described in detail in Chapter 21, "Statistical Graphics Using ODS" (*SAS/STAT User's Guide*).

Before you create graphs, ODS Graphics must be enabled (for example, by using the ODS GRAPHICS ON statement). For more information, see the section "Enabling and Disabling ODS Graphics" (Chapter 21, *SAS/STAT User's Guide*).

The overall appearance of graphs is controlled by ODS styles. Styles and other aspects of using ODS Graphics are discussed in the section "A Primer on ODS Statistical Graphics" (Chapter 21, *SAS/STAT User's Guide*).

This section describes how the HPSEVERITY procedure uses ODS to create graphics.

NOTE: The graphics are created only when you run PROC HPSEVERITY in single-machine mode.

ODS Graph Names

PROC HPSEVERITY assigns a name to each graph that it creates by using ODS. You can use these names to selectively reference the graphs. The names are listed in Table 8.18.

Table 8.18 ODS Graphics Produced by PROC HPSEVERITY

ODS Graph Name	Plot Description	PLOTS= Option
CDFPlot	Comparative CDF plot	CDF
CDFDistPlot	CDF plot per distribution	CDFPERDIST
PDFPlot	Comparative PDF plot	PDF
PDFDistPlot	PDF plot per distribution	PDFPERDIST
PPPlot	P-P plot of CDF and EDF	PP
QQPlot	Q-Q plot	QQ

Comparative CDF Plot

The comparative CDF plot helps you visually compare the cumulative distribution function (CDF) estimates of all the candidate distribution models and the empirical distribution function (EDF) estimate. The plot does not contain CDF estimates for models whose parameter estimation process does not converge. The horizontal axis represents the values of the response variable. The vertical axis represents the values of the CDF or EDF estimates.

If you specify truncation, then conditional CDF estimates are plotted. Otherwise, unconditional CDF estimates are plotted. The conditional estimates are computed by using the method that is described in the section "Truncation and Conditional CDF Estimates" on page 301.

If you specify regression effects, then the plotted CDF estimates are from a mixture distribution. For more information, see the section "CDF and PDF Estimates with Regression Effects" on page 309.

CDF Plot per Distribution

The CDF plot per distribution shows the CDF estimates of each candidate distribution model unless that model's parameter estimation process does not converge. The plot also contains estimates of the EDF. The horizontal axis represents the values of the response variable. The vertical axis represents the values of the CDF or EDF estimates.

This plot shows the lower and upper pointwise confidence limits for the EDF estimates. For an EDF estimate F_n with standard error σ_n, they are computed as $\text{MAX}(0, F_n - z_{(1-\alpha/2)}\sigma_n)$ and $\text{MIN}(1, F_n + z_{(1-\alpha/2)}\sigma_n)$, respectively, where z_p is the pth quantile from the standard normal distribution and α denotes the confidence level that you specify in the EDFALPHA= option (the default is $\alpha = 0.05$).

If you specify truncation, then conditional CDF estimates are plotted. Otherwise, unconditional CDF estimates are plotted. The conditional estimates are computed by using the method that is described in the section "Truncation and Conditional CDF Estimates" on page 301.

If you specify regression effects, then the plotted CDF estimates are from a mixture distribution. For more information, see the section "CDF and PDF Estimates with Regression Effects" on page 309.

Comparative PDF Plot

The comparative PDF plot helps you visually compare the probability density function (PDF) estimates of all the candidate distribution models. The plot does not contain PDF estimates for models whose parameter estimation process does not converge. The horizontal axis represents the values of the response variable. The vertical axis represents the values of the PDF estimates.

If you specify the HISTOGRAM option, then the plot also contains the histogram of response variable values. If you specify the KERNEL option, then the plot also contains the kernel density estimate of the response variable values.

If you specify regression effects, then the plotted PDF estimates are from a mixture distribution. For more information, see the section "CDF and PDF Estimates with Regression Effects" on page 309.

PDF Plot per Distribution

The PDF plot per distribution shows the PDF estimates of each candidate distribution model unless that model's parameter estimation process does not converge. The horizontal axis represents the values of the response variable. The vertical axis represents the values of the PDF estimates.

If you specify the HISTOGRAM option, then the plot also contains the histogram of response variable values. If you specify the KERNEL option, then the plot also contains the kernel density estimate of the response variable values.

If you specify regression effects, then the plotted PDF estimates are from a mixture distribution. For more information, see the section "CDF and PDF Estimates with Regression Effects" on page 309.

P-P Plot of CDF and EDF

The P-P plot of CDF and EDF is the probability-probability plot that compares the CDF estimates of a distribution to the EDF estimates. A plot is not prepared for models whose parameter estimation process does not converge. The horizontal axis represents the CDF estimates of a candidate distribution, and the vertical axis represents the EDF estimates.

This plot can be interpreted as displaying the data that are used for computing the EDF-based statistics of fit for the given candidate distribution. As described in the section "EDF-Based Statistics" on page 328, these statistics are computed by comparing the EDF, denoted by $F_n(y)$, to the CDF, denoted by $F(y)$, at each of the response variable values y. Using the probability inverse transform $z = F(y)$, this is equivalent to comparing the EDF of the z, denoted by $F_n(z)$, to the CDF of z, denoted by $F(z)$ (D'Agostino and Stephens 1986, Ch. 4). Because the CDF of z is a uniform distribution ($F(z) = z$), the EDF-based statistics can be computed by comparing the EDF estimate of z to the estimate of z. The horizontal axis of the plot represents the estimated CDF $\hat{z} = \hat{F}(y)$. The vertical axis represents the estimated EDF of z, $\hat{F}_n(z)$. The plot contains a scatter plot of $(\hat{z}, \hat{F}_n(z))$ points and a reference line $F_n(z) = z$ that represents the expected uniform distribution of z. Points that are scattered closer to the reference line indicate a better fit than the points that are scattered farther away from the reference line.

If you specify truncation, then the EDF estimates are conditional, as described in the section "EDF Estimates and Truncation" on page 325. So conditional estimates of CDF are displayed, which are computed by using the method that is described in the section "Truncation and Conditional CDF Estimates" on page 301.

If you specify regression effects, then the displayed CDF estimates, both unconditional and conditional, are from a mixture distribution. For more information, see the section "CDF and PDF Estimates with Regression Effects" on page 309.

Q-Q Plot

The Q-Q plot is a quantile-quantile scatter plot that compares the empirical quantiles to the quantiles from a candidate distribution. A plot is not prepared for models whose parameter estimation process does not converge. The horizontal axis represents the quantiles from a candidate distribution, and the vertical axis represents the empirical quantiles.

Each point in the plot corresponds to a specific value of the EDF estimate, F_n. The Y coordinate is the value of the response variable for which F_n is computed. The X coordinate is computed by using one of the two following methods for a candidate distribution named *dist*:

- If you have defined the *dist*_QUANTILE function that satisfies the requirements listed in the section "*dist*_QUANTILE" on page 342, then that function is invoked by using F_n and estimated distribution parameters as arguments. The QUANTILE function is defined in the Sashelp.Svrtdist library for all the predefined distributions.
- If the *dist*_QUANTILE function is not defined, then PROC HPSEVERITY numerically inverts the *dist*_CDF function at the CDF value of F_n for the estimated distribution parameters. If the *dist*_CDF function is not defined, then the exp(*dist*_LOGCDF) function is inverted. If the inversion fails, the corresponding point is not plotted in the Q-Q plot.

If you specify truncation, then the EDF estimates are conditional, as described in the section "EDF Estimates and Truncation" on page 325. The CDF inversion process, whether done numerically or by evaluating the

*dist*_QUANTILE function, needs to accept an unconditional CDF value. So the F_n value is first transformed to an unconditional estimate F_n^u as

$$F_n^u = F_n \cdot (\hat{F}(t_{\max}^r) - \hat{F}(t_{\min}^l)) + \hat{F}(t_{\min}^l)$$

where $\hat{F}(t_{\max}^r)$ and $\hat{F}(t_{\min}^l)$ are as defined in the section "Truncation and Conditional CDF Estimates" on page 301.

If you specify regression effects, then the value of the first distribution parameter is determined by using the DFMIXTURE=MEAN method that is described in the section "CDF and PDF Estimates with Regression Effects" on page 309.

Examples: HPSEVERITY Procedure

Example 8.1: Defining a Model for Gaussian Distribution

Suppose you want to fit a distribution model other than one of the predefined ones available to you. Suppose you want to define a model for the Gaussian distribution with the following typical parameterization of the PDF (f) and CDF (F):

$$f(x;\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$
$$F(x;\mu,\sigma) = \frac{1}{2}\left(1 + \operatorname{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right)$$

For PROC HPSEVERITY, a *distribution model* consists of a set of functions and subroutines that are defined with the FCMP procedure. Each function and subroutine should be written following certain rules. For more information, see the section "Defining a Severity Distribution Model with the FCMP Procedure" on page 334.

NOTE: The Gaussian distribution is not a commonly used severity distribution. It is used in this example primarily to illustrate the process of defining your own distribution models. Although the distribution has a support over the entire real line, you can fit the distribution with PROC HPSEVERITY only if the input sample contains nonnegative values.

The following SAS statements define a distribution model named NORMAL for the Gaussian distribution. The OUTLIB= option in the PROC FCMP statement stores the compiled versions of the functions and subroutines in the 'models' package of the Work.Sevexmpl library. The LIBRARY= option in the PROC FCMP statement enables this PROC FCMP step to use the SVRTUTIL_RAWMOMENTS utility subroutine that is available in the Sashelp.Svrtdist library. The subroutine is described in the section "Predefined Utility Functions" on page 346.

```
/*-------- Define Normal Distribution with PROC FCMP  ----------*/
proc fcmp library=sashelp.svrtdist outlib=work.sevexmpl.models;
   function normal_pdf(x,Mu,Sigma);
      /* Mu    : Location */
      /* Sigma : Standard Deviation */
      return ( exp(-(x-Mu)**2/(2 * Sigma**2)) /
```

```
            (Sigma * sqrt(2*constant('PI'))) );
   endsub;

   function normal_cdf(x,Mu,Sigma);
      /* Mu    : Location */
      /* Sigma : Standard Deviation */
      z = (x-Mu)/Sigma;
      return (0.5 + 0.5*erf(z/sqrt(2)));
   endsub;

   subroutine normal_parminit(dim, x[*], nx[*], F[*], Ftype, Mu, Sigma);
      outargs Mu, Sigma;
      array m[2] / nosymbols;

      /* Compute estimates by using method of moments */
      call svrtutil_rawmoments(dim, x, nx, 2, m);
      Mu    = m[1];
      Sigma = sqrt(m[2] - m[1]**2);
   endsub;

   subroutine normal_lowerbounds(Mu, Sigma);
      outargs Mu, Sigma;
      Mu = .;   /* Mu has no lower bound */
      Sigma = 0; /* Sigma > 0 */
   endsub;
quit;
```

The statements define the two functions required of any distribution model (NORMAL_PDF and NORMAL_CDF) and two optional subroutines (NORMAL_PARMINIT and NORMAL_LOWERBOUNDS). The name of each function or subroutine must follow a specific structure. It should start with the model's short or identifying name, which is 'NORMAL' in this case, followed by an underscore '_', followed by a keyword suffix such as 'PDF'. Each function or subroutine has a specific purpose. For more information about all the functions and subroutines that you can define for a distribution model, see the section "Defining a Severity Distribution Model with the FCMP Procedure" on page 334. Following is the description of each function and subroutine defined in this example:

- The PDF and CDF suffixes define functions that return the probability density function and cumulative distribution function values, respectively, given the values of the random variable and the distribution parameters.
- The PARMINIT suffix defines a subroutine that returns the initial values for the parameters by using the sample data or the empirical distribution function (EDF) estimate computed from it. In this example, the parameters are initialized by using the method of moments. Hence, you do not need to use the EDF estimates, which are available in the F array. The first two raw moments of the Gaussian distribution are as follows:

  $$E[x] = \mu,\ E[x^2] = \mu^2 + \sigma^2$$

  Given the sample estimates, m_1 and m_2, of these two raw moments, you can solve the equations $E[x] = m_1$ and $E[x^2] = m_2$ to get the following estimates for the parameters: $\hat{\mu} = m_1$ and $\hat{\sigma} = \sqrt{m_2 - m_1^2}$. The NORMAL_PARMINIT subroutine implements this solution. It uses the SVRTUTIL_RAWMOMENTS utility subroutine to compute the first two raw moments.

- The LOWERBOUNDS suffix defines a subroutine that returns the lower bounds on the parameters. PROC HPSEVERITY assumes a default lower bound of 0 for all the parameters when a LOWERBOUNDS subroutine is not defined. For the parameter μ (*Mu*), there is no lower bound, so you need to define the NORMAL_LOWERBOUNDS subroutine. It is recommended that you assign bounds for all the parameters when you define the LOWERBOUNDS subroutine or its counterpart, the UPPERBOUNDS subroutine. Any unassigned value is returned as a missing value, which PROC HPSEVERITY interprets to mean that the parameter is unbounded, and that might not be what you want.

You can now use this distribution model with PROC HPSEVERITY. Let the following DATA step statements simulate a normal sample with $\mu = 10$ and $\sigma = 2.5$:

```
/*-------- Simulate a Normal sample ----------*/
data testnorm(keep=y);
   call streaminit(12345);
   do i=1 to 100;
      y = rand('NORMAL', 10, 2.5);
      output;
   end;
run;
```

Prior to using your distribution with PROC HPSEVERITY, you must communicate the location of the library that contains the definition of the distribution and the locations of libraries that contain any functions and subroutines used by your distribution model. The following OPTIONS statement sets the CMPLIB= system option to include the FCMP library Work.Sevexmpl in the search path used by PROC HPSEVERITY to find FCMP functions and subroutines:

```
/*--- Set the search path for functions defined with PROC FCMP ---*/
options cmplib=(work.sevexmpl);
```

Now, you are ready to fit the NORMAL distribution model with PROC HPSEVERITY. The following statements fit the model to the values of Y in the Work.Testnorm data set:

```
/*--- Fit models with PROC HPSEVERITY ---*/
proc hpseverity data=testnorm print=all;
   loss y;
   dist Normal;
run;
```

The DIST statement specifies the identifying name of the distribution model, which is 'NORMAL'. Neither the INEST= option nor the INSTORE= option is specified in the PROC HPSEVERITY statement, and the INIT= option is not specified in the DIST statement. So PROC HPSEVERITY initializes the parameters by invoking the NORMAL_PARMINIT subroutine.

Some of the results prepared by the preceding PROC HPSEVERITY step are shown in Output 8.1.1 and Output 8.1.2. The descriptive statistics of variable Y and the "Model Selection" table, which includes just the normal distribution, are shown in Output 8.1.1.

Output 8.1.1 Summary of Results for Fitting the Normal Distribution

The HPSEVERITY Procedure

Input Data Set	
Name	WORK.TESTNORM

Descriptive Statistics for y	
Observations	100
Observations Used for Estimation	100
Minimum	3.88249
Maximum	16.00864
Mean	10.02059
Standard Deviation	2.37730

Model Selection

Distribution	Converged	-2 Log Likelihood	Selected
Normal	Yes	455.97541	Yes

The initial values for the parameters, the optimization summary, and the final parameter estimates are shown in Output 8.1.2. No iterations are required to arrive at the final parameter estimates, which are identical to the initial values. This confirms the fact that the maximum likelihood estimates for the Gaussian distribution are identical to the estimates obtained by the method of moments that was used to initialize the parameters in the NORMAL_PARMINIT subroutine.

Output 8.1.2 Details of the Fitted Normal Distribution Model

The HPSEVERITY Procedure
Normal Distribution

Distribution Information	
Name	Normal
Distribution Parameters	2

Initial Parameter Values and Bounds

Parameter	Initial Value	Lower Bound	Upper Bound
Mu	10.02059	-Infty	Infty
Sigma	2.36538	1.05367E-8	Infty

Optimization Summary	
Optimization Technique	Trust Region
Iterations	0
Function Calls	4
Log Likelihood	-227.98770

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Mu	1	10.02059	0.23894	41.94	<.0001
Sigma	1	2.36538	0.16896	14.00	<.0001

The NORMAL distribution defined and illustrated here has no scale parameter, because all the following inequalities are true:

$$f(x;\mu,\sigma) \neq \frac{1}{\mu} f(\frac{x}{\mu};1,\sigma)$$
$$f(x;\mu,\sigma) \neq \frac{1}{\sigma} f(\frac{x}{\sigma};\mu,1)$$
$$F(x;\mu,\sigma) \neq F(\frac{x}{\mu};1,\sigma)$$
$$F(x;\mu,\sigma) \neq F(\frac{x}{\sigma};\mu,1)$$

This implies that you cannot estimate the influence of regression effects on a model for the response variable based on this distribution.

Example 8.2: 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 the Gaussian distribution by replacing the location parameter μ with another parameter, $\alpha = \mu/\sigma$, and defining the PDF (f) and the CDF (F) as follows:

$$f(x;\sigma,\alpha) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x}{\sigma}-\alpha\right)^2\right)$$
$$F(x;\sigma,\alpha) = \frac{1}{2}\left(1+\operatorname{erf}\left(\frac{1}{\sqrt{2}}\left(\frac{x}{\sigma}-\alpha\right)\right)\right)$$

You can verify that σ is the scale parameter, because both of the following equalities are true:

$$f(x;\sigma,\alpha) = \frac{1}{\sigma} f(\frac{x}{\sigma};1,\alpha)$$
$$F(x;\sigma,\alpha) = F(\frac{x}{\sigma};1,\alpha)$$

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 HPSEVERITY 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.

```
/*-------- Define Normal Distribution With Scale Parameter  ----------*/
proc fcmp library=sashelp.svrtdist outlib=work.sevexmpl.models;
   function normal_s_pdf(x, Sigma, Alpha);
      /* Sigma : Scale & Standard Deviation */
      /* Alpha : Scaled mean */
      return ( exp(-(x/Sigma - Alpha)**2/2) /
```

```
            (Sigma * sqrt(2*constant('PI'))) );
   endsub;

   function normal_s_cdf(x, Sigma, Alpha);
      /* Sigma : Scale & Standard Deviation */
      /* Alpha : Scaled mean */
      z = x/Sigma - Alpha;
      return (0.5 + 0.5*erf(z/sqrt(2)));
   endsub;

   subroutine normal_s_parminit(dim, x[*], nx[*], F[*], Ftype, Sigma, Alpha);
      outargs Sigma, Alpha;
      array m[2] / nosymbols;
      /* Compute estimates by using method of moments */
      call svrtutil_rawmoments(dim, x, nx, 2, m);
      Sigma = sqrt(m[2] - m[1]**2);
      Alpha = m[1]/Sigma;
   endsub;

   subroutine normal_s_lowerbounds(Sigma, Alpha);
      outargs Sigma, Alpha;
      Alpha = .; /* Alpha has no lower bound */
      Sigma = 0; /* Sigma > 0 */
   endsub;
quit;
```

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 HPSEVERITY, so that it can identify which is the scale parameter. When you specify regression effects, PROC HPSEVERITY 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 σ is affected by the regressors as follows:

$$\sigma = \exp(1 + 0.5\ \mathrm{X1} + 0.75\ \mathrm{X3} - 2\ \mathrm{X4} + \mathrm{X5})$$

The sample is simulated such that the regressor X2 is linearly dependent on regressors X1 and X3.

```
/*--- Simulate a Normal 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);
   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 statements use PROC HPSEVERITY 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 HPSEVERITY --------*/
proc hpseverity data=testnorm_reg print=all;
   loss y;
   scalemodel x1-x5;
   dist Normal_s burr logn pareto weibull;
run;
```

The "Model Selection" table in Output 8.2.1 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.

Output 8.2.1 Summary of Results for Fitting the Normal Distribution with Regressors

The HPSEVERITY Procedure

Input Data Set	
Name	WORK.TESTNORM_REG

Model Selection

Distribution	Converged	-2 Log Likelihood	Selected
Normal_s	Yes	603.95786	Yes
Burr	Maybe	612.81685	No
Logn	Yes	749.20125	No
Pareto	Yes	841.07022	No
Weibull	Yes	612.77496	No

Output 8.2.1 *continued*

All Fit Statistics

Distribution	-2 Log Likelihood		AIC		AICC		BIC		KS		AD		CvM	
Normal_s	603.95786	*	615.95786	*	616.86108	*	631.58888	*	1.52388		4.00152		0.70769	
Burr	612.81685		626.81685		628.03424		645.05304		1.50448	*	3.90072	*	0.63399	*
Logn	749.20125		761.20125		762.10448		776.83227		2.88110		16.20558		3.04825	
Pareto	841.07022		853.07022		853.97345		868.70124		4.83810		31.60568		6.84046	
Weibull	612.77496		624.77496		625.67819		640.40598		1.50490		3.90559		0.63458	

Note: The asterisk (*) marks the best model according to each column's criterion.

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 8.2.2. The initial values table indicates that the regressor X2 is redundant, which is expected. More importantly, the convergence status indicates that it requires more than 50 iterations. PROC HPSEVERITY 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 8.2.2 Details of the Fitted Burr Distribution Model

The HPSEVERITY Procedure
Burr Distribution

Distribution Information	
Name	Burr
Description	Burr Distribution (Type XII Family)
Distribution Parameters	3
Regression Parameters	4

Initial Parameter Values and Bounds

Parameter	Initial Value	Lower Bound	Upper Bound
Theta	25.75198	1.05367E-8	Infty
Alpha	2.00000	1.05367E-8	Infty
Gamma	2.00000	1.05367E-8	Infty
x1	0.07345	-709.78271	709.78271
x2	Redundant		
x3	-0.14056	-709.78271	709.78271
x4	0.27064	-709.78271	709.78271
x5	-0.23230	-709.78271	709.78271

Convergence Status
Needs more than 50 iterations.

Optimization Summary	
Optimization Technique	Trust Region
Iterations	50
Function Calls	137
Log Likelihood	-306.40842

The following PROC HPSEVERITY 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 X2 from the model:

```
/*--- Refit and compare models with higher limit on iterations ---*/
proc hpseverity data=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 8.2.3 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 8.2.3 Summary of Results after Changing Maximum Number of Iterations

The HPSEVERITY Procedure

Input Data Set	
Name	WORK.TESTNORM_REG

Model Selection

Distribution	Converged	-2 Log Likelihood	Selected
Normal_s	Yes	603.95786	Yes
Burr	Yes	612.79276	No
Weibull	Yes	612.77496	No

All Fit Statistics

Distribution	-2 Log Likelihood		AIC		AICC		BIC		KS		AD		CvM	
Normal_s	603.95786	*	615.95786	*	616.86108	*	631.58888	*	1.52388		4.00152		0.70769	
Burr	612.79276		626.79276		628.01015		645.02895		1.50472	*	3.90351	*	0.63433	*
Weibull	612.77496		624.77496		625.67819		640.40598		1.50490		3.90559		0.63458	

Note: The asterisk (*) marks the best model according to each column's criterion.

Example 8.3: 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:

$g(x)$	PDF of the body distribution
$G(x)$	CDF of the body distribution
$h(x)$	PDF of the tail distribution
$H(x)$	CDF of the tail distribution
θ	scale parameter for the body distribution
Ω	set of nonscale parameters for the body distribution
ξ	shape parameter for the GPD tail distribution
x_r	normalized value of the response variable at which the tail starts
p_n	mixing probability

Given these notations, the PDF $f(x)$ and the CDF $F(x)$ of the mixed-tail distribution are defined as

$$f(x) = \begin{cases} \frac{p_n}{G(x_b)} g(x) & \text{if } x \le x_b \\ (1-p_n)h(x-x_b) & \text{if } x > x_b \end{cases}$$

$$F(x) = \begin{cases} \frac{p_n}{G(x_b)} G(x) & \text{if } x \le x_b \\ p_n + (1-p_n)H(x-x_b) & \text{if } x > x_b \end{cases}$$

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 $X \le x_b$, where X denotes the random response variable.
- The tail distribution is the generalized Pareto distribution of the $(X - x_b)$ values.
- 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 θ, Ω, ξ, x_r, and p_n. The scale of the GPD tail distribution θ_t is computed as

$$\theta_t = \frac{G(x_b;\theta,\Omega)}{g(x_b;\theta,\Omega)} \frac{(1-p_n)}{p_n} = \theta \frac{G(x_r;\theta=1,\Omega)}{g(x_r;\theta=1,\Omega)} \frac{(1-p_n)}{p_n}$$

The parameter x_r is usually estimated 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.Svrtdist library. The algorithm and the utility function are described in detail in the section "Predefined Utility Functions" on page 346. The function computes an estimate of x_b, which can be used to compute an estimate of x_r because $x_r = x_b/\hat{\theta}$, 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:

```
/*------- Define Lognormal Body-GPD Tail Mixed Distribution -------*/
proc fcmp library=sashelp.svrtdist outlib=work.sevexmpl.models;
   function LOGNGPD_DESCRIPTION() $256;
      length desc $256;
      desc1 = "Lognormal Body-GPD Tail Distribution.";
      desc2 = " Mu, Sigma, and Xi are free parameters.";
      desc3 = " Xr and Pn are constant parameters.";
      desc = desc1 || desc2 || desc3;
      return(desc);
   endsub;

   function LOGNGPD_SCALETRANSFORM() $3;
      length xform $3;
      xform = "LOG";
      return (xform);
   endsub;

   subroutine LOGNGPD_CONSTANTPARM(Xr,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;
   endsub;

   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;

         /* 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 */
   endsub;
quit;
```

Note the following points about the LOGNGPD definition:

- The parameters x_r and p_n are not estimated with the maximum likelihood method used by PROC HPSEVERITY, so you need to specify them as *constant* parameters by defining the

*dist*_CONSTANTPARM subroutine. The signature of the LOGNGPD_CONSTANTPARM subroutine lists only the constant parameters *Xr* and *Pn*.

- The parameter x_r is estimated by first using the SVRTUTIL_HILLCUTOFF utility function to compute an estimate of the cutoff point $\hat{x}_b$ and then computing $x_r = \hat{x}_b / e^{\hat{\mu}}$. 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_nth 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 HPSEVERITY interprets as having no lower bound. You do not need to specify any bounds for the constant parameters *Xr* and *Pn*, because they are 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.

```
/*----- 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 = 100;
   Mu = 1.5;
   Sigma = 0.25;
   Xi = 1.5;
   Pn = 0.8;

   /* Generate data comprising the lognormal body */
   Nbody = N*Pn;
   do i=1 to Nbody;
      y = exp(Mu) * rand('LOGNORMAL')**Sigma;
      output;
   end;

   /* Generate data comprising 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 statements use PROC HPSEVERITY 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 Work.Parmest data set.

```
/*--- Set the search path for functions defined with PROC FCMP ---*/
options cmplib=(work.sevexmpl);
/*-------- Fit LOGNGPD model with PROC HPSEVERITY --------*/
proc hpseverity data=testmixdist print=all outest=parmest;
   loss y;
   dist logngpd burr logn gpd;
run;
```

Some of the results prepared by PROC HPSEVERITY are shown in Output 8.3.1 and Output 8.3.2. The "Model Selection" table in Output 8.3.1 indicates that all models converged. The last table in Output 8.3.1 shows that the model with LOGNGPD distribution has the best fit according to almost all the statistics of fit. The Burr distribution model is the closest contender to the LOGNGPD model, but the GPD distribution model fits the data very poorly.

Output 8.3.1 Summary of Fitting Mixed-Tail Distribution

The HPSEVERITY Procedure

Input Data Set	
Name	WORK.TESTMIXDIST
Label	Lognormal Body-GPD Tail Sample

Model Selection

Distribution	Converged	-2 Log Likelihood	Selected
logngpd	Yes	418.78232	Yes
Burr	Yes	424.93728	No
Logn	Yes	459.43471	No
Gpd	Yes	558.13444	No

All Fit Statistics

Distribution	-2 Log Likelihood		AIC		AICC		BIC		KS		AD		CvM	
logngpd	418.78232	*	428.78232	*	429.42062	*	441.80817		0.62140	*	0.31670	*	0.04972	*
Burr	424.93728		430.93728		431.18728		438.75280	*	0.71373		0.57649		0.07860	
Logn	459.43471		463.43471		463.55842		468.64505		1.55267		3.27122		0.48448	
Gpd	558.13444		562.13444		562.25815		567.34478		3.43470		16.74156		3.31860	

Note: The asterisk (*) marks the best model according to each column's criterion.

The detailed results for the LOGNGPD distribution are shown in Output 8.3.2. The initial values table indicates the values computed by LOGNGPD_PARMINIT subroutine for the *Xr* and *Pn* parameters. It also uses the bounds columns to indicate the constant parameters. 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 estimates of the constant parameters *Xr* and *Pn* have not changed from their initial values.

Output 8.3.2 Detailed Results for the LOGNGPD Distribution

The HPSEVERITY Procedure
logngpd Distribution

Distribution Information	
Name	logngpd
Description	Lognormal Body-GPD Tail Distribution. Mu, Sigma, and Xi are free parameters. Xr and Pn are constant parameters.
Distribution Parameters	5

Initial Parameter Values and Bounds			
Parameter	**Initial Value**	**Lower Bound**	**Upper Bound**
Mu	1.49954	-Infty	Infty
Sigma	0.76306	1.05367E-8	Infty
Xi	0.36661	1.05367E-8	Infty
Xr	1.27395	Constant	Constant
Pn	0.80000	Constant	Constant

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

Optimization Summary	
Optimization Technique	Trust Region
Iterations	11
Function Calls	33
Failed Function Calls	1
Log Likelihood	-209.39116

Parameter Estimates					
Parameter	**DF**	**Estimate**	**Standard Error**	**t Value**	**Approx Pr > |t|**
Mu	1	1.57921	0.06426	24.57	<.0001
Sigma	1	0.31868	0.04459	7.15	<.0001
Xi	1	1.03771	0.38205	2.72	0.0078
Xr	1	1.27395	Constant	.	.
Pn	1	0.80000	Constant	.	.

The following SAS statements use the parameter estimates to compute the value where the tail region is estimated to start ($x_b = e^{\hat{\mu}}\hat{x_r}$) and the scale of the GPD tail distribution ($\theta_t = \frac{G(x_b)}{g(x_b)}\frac{(1-p_n)}{p_n}$):

```
/*-------- Compute tail cutoff and tail distribution's scale --------*/
data xb_thetat(keep=x_b theta_t);
   set 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 8.3.3 Start of the Tail and Scale of the GPD Tail Distribution

x_b	theta_t
6.18005	1.27865

The computed values of x_b and θ_t are shown as *x_b* and *theta_t* in Output 8.3.3. Equipped with this additional derived information, you can now interpret the results of fitting the mixed-tail distribution as follows:

- The tail starts at $y \approx 6.18$. The primary benefit of using the scale-normalized cutoff (x_r) as the constant parameter instead of using the actual cutoff (x_b) is that the absolute cutoff is optimized by virtue of optimizing the scale of the body region ($\theta = e^{\mu}$). It works well for this example. However, by keeping x_r constant, you must rely on Hill's tail index estimator to yield an initial estimate of x_b that is close to an optimal estimate. In general, you might want to optimize x_r by making it a free parameter, which gives you more flexibility in optimizing x_b. You can make x_r a free parameter by removing *Xr* from the signature of the LOGNGPD_CONSTANTPARM subroutine.
- The values $y \leq 6.18$ follow the lognormal distribution with parameters $\mu \approx 1.58$ and $\sigma \approx 0.32$. These parameter estimates are reasonably close to the parameters of the body distribution that is used for simulating the sample.
- If X_t denotes the loss random variable for the tail defined as $X_t = X - x_b$, where X is the original loss variable, then for this example, $\Pr[X_t = X - 6.18 | X_t > 0]$ follows the GPD density function with scale $\theta_t \approx 1.28$ and shape $\xi \approx 1.04$.

Example 8.4: 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 HPSEVERITY 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 rate-making and pure premium modeling.

Assume a Tweedie process, whose mean μ 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 μ_0 represents the base value of the mean (you can think of μ_0 as $\exp(\beta_0)$, where β_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 HPSEVERITY for the scale parameter θ,

$$\theta = \theta_0 \exp\left(\sum_{j=1}^{k} \beta_j x_j\right)$$

where θ_0 represents the base value of the scale parameter. As described in the section "Tweedie Distributions" on page 292, 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 λ 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:

$$\mu = 5\ \exp(0.25\,\mathrm{x1} - \mathrm{x2} + 3\,\mathrm{x3})$$

Thus, the population values of regression parameters are $\mu_0 = 5$, $\beta_1 = 0.25$, $\beta_2 = -1$, and $\beta_3 = 3$. You can find the code used to generate the sample in the PROC HPSEVERITY sample program *hsevex04.sas*.

The following PROC HPSEVERITY step uses the sample in Work.Test_Sevtw data set 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 hpseverity data=test_sevtw outest=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 prepared by PROC HPSEVERITY are shown in Output 8.4.1 and Output 8.4.2. The distribution information and the convergence results are shown in Output 8.4.1.

Output 8.4.1 Convergence Results for the STWEEDIE Model with Regressors

The HPSEVERITY Procedure
stweedie Distribution

Distribution Information	
Name	stweedie
Description	Tweedie Distribution with Scale Parameter
Distribution Parameters	3
Regression Parameters	3

Output 8.4.1 *continued*

Convergence Status
Convergence criterion (FCONV=2.220446E-16) satisfied.

Optimization Summary	
Optimization Technique	Dual Quasi-Newton
Iterations	41
Function Calls	258
Log Likelihood	-1044.3

The final parameter estimates of the STWEEDIE regression model are shown in Output 8.4.2. The estimate that is reported for the parameter Theta is the estimate of the base value θ_0. The estimates of regression coefficients β_1, β_2, and β_3 are indicated by the rows of x1, x2, and x3, respectively.

Output 8.4.2 Parameter Estimates for the STWEEDIE Model with Regressors

Parameter Estimates					
Parameter	**DF**	**Estimate**	**Standard Error**	**t Value**	**Approx Pr > |t|**
Theta	1	0.82281	0.24835	3.31	0.0010
Lambda	1	16.22604	11.58066	1.40	0.1622
P	1	1.74911	0.18595	9.41	<.0001
x1	1	0.27954	0.09870	2.83	0.0049
x2	1	-0.76686	0.10307	-7.44	<.0001
x3	1	3.03218	0.10135	29.92	<.0001

If your goal is to explain the influence of regression effects on the scale parameter, then the output displayed in Output 8.4.2 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 μ and θ, μ 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 β_j are identical for the mean and the scale model, and the base value μ_0 of the mean model is

$$\mu_0 = \theta_0 \lambda \frac{2-p}{p-1}$$

The estimate of μ_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 Ω, then the MLE of g has an asymptotic normal distribution with mean $g(\hat{\Omega})$ and covariance $C = (\partial \mathbf{g})' \Sigma (\partial \mathbf{g})$, where $\hat{\Omega}$ is the MLE of Ω, Σ is the estimate of covariance matrix of Ω, and $\partial \mathbf{g}$ is the gradient vector of g with respect to Ω evaluated at $\hat{\Omega}$. For μ_0, the function is $g(\Omega) = \theta_0 \lambda (2-p)/(p-1)$.

The gradient vector is

$$\begin{aligned}\partial \mathbf{g} &= \left(\frac{\partial g}{\partial \theta_0} \quad \frac{\partial g}{\partial \lambda} \quad \frac{\partial g}{\partial p} \quad \frac{\partial g}{\partial \beta_1} \cdots \frac{\partial g}{\partial \beta_k}\right) \\ &= \left(\frac{\mu_0}{\theta_0} \quad \frac{\mu_0}{\lambda} \quad \frac{-\mu_0}{(p-1)(2-p)} \quad 0 \ldots 0\right)\end{aligned}$$

You can write a DATA step that implements these computations by using the parameter and covariance estimates prepared by PROC HPSEVERITY step. The DATA step program is available in the sample program *hsevex04.sas*. The estimates of μ_0 prepared by that program are shown in Output 8.4.3. These estimates and the estimates of β_j as shown in Output 8.4.2 are reasonably close (that is, within one or two standard errors) to the parameters of the population from which the sample in Work.Test_Sevtw data set was drawn.

Output 8.4.3 Estimate of the Base Value Mu0 of the Mean Parameter

Parameter	Estimate	Standard Error	t Value	Approx Pr > |t|
Mu0	4.47141	0.42190	10.5982	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 ϕ is the dispersion parameter. Using the relationship between the parameters TWEEDIE and STWEEDIE distributions as described in the section "Tweedie Distributions" on page 292, the regression model for the dispersion parameter is

$$\begin{aligned}\log(\phi) &= (2-p)\log(\mu) - \log(\lambda(2-p)) \\ &= ((2-p)\log(\mu_0) - \log(\lambda(2-p))) + (2-p)\sum_{j=1}^{k}\beta_j x_j\end{aligned}$$

Subsequently, the regression model for the variance is

$$\begin{aligned}\log(V) &= 2\log(\mu) - \log(\lambda(2-p)) \\ &= (2\log(\mu_0) - \log(\lambda(2-p))) + 2\sum_{j=1}^{k}\beta_j x_j\end{aligned}$$

In summary, PROC HPSEVERITY enables you to estimate regression effects on various parameters and statistics of the Tweedie model.

Example 8.5: 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 HPSEVERITY 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):

```
/* 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;
```

The following PROC HPSEVERITY step fits all the predefined distributions to the data in the Work.Gdental data set:

```
/* Fit all predefined distributions */
proc hpseverity data=gdental edf=turnbull print=all criterion=aicc;
   loss / rc=lowerbd lc=upperbd;
   weight count;
   dist _predef_;
   performance nthreads=1;
run;
```

The EDF= option in the PROC HPSEVERITY 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 set is censored. The PERFORMANCE statement specifies that just one thread of execution be used, to minimize the overhead associated with multithreading, because the input data set is very small.

Some of the key results prepared by PROC HPSEVERITY are shown in Output 8.5.1. According to the "Model Selection" table in Output 8.5.1, all distribution models have converged. The "All Fit Statistics" table in Output 8.5.1 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 8.5.1 Statistics of Fit for Interval-Censored Data

The HPSEVERITY Procedure

Input Data Set	
Name	WORK.GDENTAL

Model Selection			
Distribution	**Converged**	**AICC**	**Selected**
Burr	Yes	51.41112	No
Exp	Yes	44.64768	Yes
Gamma	Yes	47.63969	No
Igauss	Yes	48.05874	No
Logn	Yes	47.34027	No
Pareto	Yes	47.16908	No
Gpd	Yes	47.16908	No
Weibull	Yes	47.47700	No

Output 8.5.1 *continued*

All Fit Statistics

Distribution	-2 Log Likelihood		AIC		AICC		BIC		KS		AD		CvM
Burr	41.41112	*	47.41112		51.41112		48.31888		0.08974	*	0.00103	*	0.0000816	*
Exp	42.14768		44.14768	*	44.64768	*	44.45026	*	0.26412		0.09936		0.01866	
Gamma	41.92541		45.92541		47.63969		46.53058		0.19569		0.04608		0.00759	
Igauss	42.34445		46.34445		48.05874		46.94962		0.34514		0.12301		0.02562	
Logn	41.62598		45.62598		47.34027		46.23115		0.16853		0.01884		0.00333	
Pareto	41.45480		45.45480		47.16908		46.05997		0.11423		0.00739		0.0009084	
Gpd	41.45480		45.45480		47.16908		46.05997		0.11423		0.00739		0.0009084	
Weibull	41.76272		45.76272		47.47700		46.36789		0.17238		0.03293		0.00472	

Note: The asterisk (*) marks the best model according to each column's criterion.

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.

Example 8.6: Benefits of Distributed and Multithreaded Computing

One of the key features of the HPSEVERITY procedure is that it takes advantage of the distributed and multithreaded computing machinery in order to solve a given problem faster. This example illustrates the benefits of using multithreading and distributed computing.

The example uses a simulated data set Work.Largedata, which contains 10,000,000 observations, some of which are right-censored or left-truncated. The losses are affected by three external effects. The DATA step program that generates this data set is available in the accompanying sample program *hsevex06.sas*.

The following PROC HPSEVERITY step fits all the predefined distributions to the data in the Work.Largedata data set on the client machine with just one thread of computation:

```
/* Fit all predefined distributions without any multithreading or
   distributed computing */
proc hpseverity data=largedata criterion=aicc initsample(size=20000);
   loss y / lt=threshold rc=limit;
   scalemodel x1-x3;
   dist _predef_;
   performance nthreads=1 bufsize=1000000 details;
run;
```

The NTHREADS=1 option in the PERFORMANCE statement specifies that just one thread of computation be used. The absence of the NODES= option in the PERFORMANCE statement specifies that single-machine

mode of execution be used. That is, this step does not use any multithreading or distributed computing. The BUFSIZE= option in the PERFORMANCE statement specifies the number of observations to read at one time. Specifying a larger value tends to decrease the time it takes to load the data. The DETAILS option in the performance statement enables reporting of the timing information. The INITSAMPLE option in the PROC HPSEVERITY statement specifies that a uniform random sample of maximum 20,000 observations be used for parameter initialization.

The "Performance Information" and "Procedure Task Timing" tables that PROC HPSEVERITY creates are shown in Output 8.6.1. The "Performance Information" table contains the information about the execution environment. The "Procedure Task Timing" table indicates the total time and relative time taken by each of the four main steps of PROC HPSEVERITY. As that table shows, it takes around 26.3 minutes for the task of estimating parameters, which is usually the most time-consuming of all the tasks.

Output 8.6.1 Performance for Single-Machine Mode with No Multithreading

Performance Information	
Execution Mode	Single-Machine
Number of Threads	1

Procedure Task Timing		
Task	**Seconds**	**Percent**
Load and Prepare Models	4.57	0.29%
Load and Prepare Data	10.36	0.65%
Initialize Parameters	0.81	0.05%
Estimate Parameters	1576.51	98.93%
Compute Fit Statistics	1.32	0.08%

If the grid appliance is not available, you can improve the performance by using multiple threads of computation; this is in fact the default. The following PROC HPSEVERITY step fits all the predefined distributions by using all the logical CPU cores of the machine:

```
/* Specify that all the logical CPU cores on the machine be used */
options cpucount=actual;

/* Fit all predefined distributions with multithreading, but no
   distributed computing */
proc hpseverity data=largedata criterion=aicc initsample(size=20000);
   loss y / lt=threshold rc=limit;
   scalemodel x1-x3;
   dist _predef_;
   performance bufsize=1000000 details;
run;
```

When you do not specify the NTHREADS= option in the PERFORMANCE statement, the HPSEVERITY procedure uses the value of the CPUCOUNT= system option to decide the number of threads to use in single-machine mode. Setting the CPUCOUNT= option to ACTUAL before the PROC HPSEVERITY step enables the procedure to use all the logical cores of the machine. The machine that is used to obtain these results (and the earlier results in Output 8.6.1) has four physical CPU cores, each with a clock speed of 3.4 GHz. Hyperthreading is enabled on the CPUs to yield eight logical CPU cores; this number is confirmed by the "Performance Information" table in Output 8.6.2. The results in the "Procedure Task Timing" table in Output 8.6.2 indicate that the use of multithreading has improved the performance by reducing the time to estimate parameters to around 5.5 minutes.

Output 8.6.2 Performance for Single-Machine Mode with Eight Threads

Performance Information	
Execution Mode	Single-Machine
Number of Threads	8

Procedure Task Timing		
Task	**Seconds**	**Percent**
Load and Prepare Models	0.75	0.22%
Load and Prepare Data	1.06	0.31%
Initialize Parameters	0.67	0.20%
Estimate Parameters	332.05	97.44%
Compute Fit Statistics	6.25	1.83%

When a grid appliance is available, performance can be further improved by using more than one node in the distributed mode of execution. Large data sets are usually predistributed on the grid appliance that hosts a distributed database. In other words, large problems are best suited for the alongside-the-database model of execution. However, for the purpose of illustration, this example assumes that the data set is available on the client machine and is then distributed to the grid nodes by the HPSEVERITY procedure according to the options that are specified in the PERFORMANCE statement.

The next few PROC HPSEVERITY steps are run on a grid appliance by varying the number of nodes and the number of threads that are used within each node.

You can specify your distributed computing environment by using SAS environment variables or by specifying options in the PERFORMANCE statement, or by a combination of these methods. For example, you can submit the following statements to specify the appliance host (GRIDHOST= SAS environment variable) and the installation location of shared libraries on the appliance (GRIDINSTALLLOC= SAS environment variable):

```
/* Set the appliance host and installation location that are
   appropriate for your distributed mode setup */
option set=GRIDHOST      ="&GRIDHOST";
option set=GRIDINSTALLLOC="&GRIDINSTALLLOC";
```

To run the preceding statements successfully, you need to set the macro variables GRIDHOST and GRIDINSTALLLOC to resolve to appropriate values, or you can replace the references to macro variables with the appropriate values. Alternatively, you can specify the HOST= and INSTALL= options in the PERFORMANCE statement; this method is used in the PROC HPSEVERITY steps of this example. You can use other SAS environment variables and PERFORMANCE statement options to describe your distributed computing environment. For more information, see the section "PERFORMANCE Statement" on page 31.

To establish a reference point for the performance of one CPU of a grid node, the results of using only one node of the grid appliance without any multithreading are presented first. The particular grid appliance that is used to obtain these results has more than sixteen nodes. Each node has 8 dual-core CPUs with a clock speed of 2.7 GHz. The following PROC HPSEVERITY step fits all the predefined distributions to the data in the Work.Largedata data set:

```
/* Fit all predefined distributions on 1 grid node without
   any multithreading */
proc hpseverity data=largedata criterion=aicc initsample(size=20000);
   loss y / lt=threshold rc=limit;
   scalemodel x1-x3;
   dist _predef_;
   performance nodes=1 nthreads=1 details
      host="&GRIDHOST" install="&GRIDINSTALLLOC";
run;
```

The PERFORMANCE statement specifies that only one node be used to fit the models, with only one thread of computation on that node. The "Performance Information" and "Procedure Task Timing" tables that PROC HPSEVERITY creates are shown in Output 8.6.3. It takes around 35.7 minutes to complete the task of estimating parameters. Note that this time is longer than the time taken for the single-machine mode with one thread of computation, because the CPUs of an individual grid node are slower than the CPUs of the machine that is used in single-machine mode. When the performance is measured, the grid node is shared among multiple users, unlike the machine that is used in single-machine mode.

Output 8.6.3 Performance on One Grid Appliance Node with No Multithreading

Performance Information	
Host Node	<< your grid host >>
Install Location	<< your grid install location >>
Execution Mode	Distributed
Number of Compute Nodes	1
Number of Threads per Node	1

Procedure Task Timing		
Task	**Seconds**	**Percent**
Load and Prepare Models	0.54	0.03%
Load and Prepare Data	0.94	0.04%
Initialize Parameters	1.09	0.05%
Estimate Parameters	2141.83	99.80%
Compute Fit Statistics	1.78	0.08%

The computations and time taken to fit each model are shown in the "Estimation Details" table of Output 8.6.4, which is generated whenever you specify the DETAILS option in the PERFORMANCE statement. This table can be useful for comparing the relative effort required to fit each model and drawing some broader conclusions. For example, even if the Pareto distribution takes a larger number of iterations, function calls, and gradient and Hessian updates than the gamma distribution, it takes less time to complete; this indicates that the individual PDF and CDF computations of the gamma distribution are more expensive than those of the Pareto distribution.

Output 8.6.4 Estimation Details

Estimation Details						
Distribution	Converged	Iterations	Function Calls	Gradient Updates	Hessian Updates	Time (Seconds)
Burr	Yes	11	28	104	90	343.31
Exp	Yes	4	12	27	20	34.03
Gamma	Yes	5	15	35	27	695.61
Igauss	Yes	4	14	27	20	217.14
Logn	Yes	4	12	27	20	123.94
Pareto	Maybe	50	137	1430	1377	512.71
Gpd	Yes	6	17	44	35	138.71
Weibull	Yes	4	12	27	20	76.36

To obtain the next reference point for performance, the following PROC HPSEVERITY step specifies that 16 computation threads be used on one node of the grid appliance:

```
/* Fit all predefined distributions on 1 grid node with multithreading */
proc hpseverity data=largedata criterion=aicc initsample(size=20000);
   loss y / lt=threshold rc=limit;
   scalemodel x1-x3;
   dist _predef_;
   performance nodes=1 nthreads=16 details
      host="&GRIDHOST" install="&GRIDINSTALLLOC";
run;
```

The performance tables that are created by the preceding statements are shown in Output 8.6.5. As the "Procedure Task Timing" table shows, use of multithreading has improved the performance significantly over that of the single-threaded case. Now, it takes around 2.8 minutes to complete the task of estimating parameters.

Output 8.6.5 Performance Information with Multithreading but No Distributed Computing

Performance Information	
Host Node	<< your grid host >>
Install Location	<< your grid install location >>
Execution Mode	Distributed
Number of Compute Nodes	1
Number of Threads per Node	16

Procedure Task Timing		
Task	Seconds	Percent
Load and Prepare Models	0.53	0.31%
Load and Prepare Data	0.48	0.28%
Initialize Parameters	1.01	0.59%
Estimate Parameters	167.67	98.27%
Compute Fit Statistics	0.94	0.55%

You can combine the power of multithreading and distributed computing by specifying that multiple nodes of the grid be used to accomplish the task. The following PROC HPSEVERITY step specifies that 16 nodes of the grid appliance be used:

```
/* Fit all predefined distributions with distributed computing and
   multithreading within each node */
proc hpseverity data=largedata criterion=aicc initsample(size=20000);
   loss y / lt=threshold rc=limit;
   scalemodel x1-x3;
   dist _predef_;
   performance nodes=16 nthreads=16 details
      host="&GRIDHOST" install="&GRIDINSTALLLOC";
run;
```

When the DATA= data set is local to the client machine, as it is in this example, you must specify a nonzero value for the NODES= option in the PERFORMANCE statement in order to enable the distributed mode of execution. In other words, for the distributed mode that is not executing alongside the database, omitting the NODES= option is equivalent to specifying NODES=0, which is single-machine mode.

The performance tables that are created by the preceding statements are shown in Output 8.6.6. If you compare these tables to the tables in Output 8.6.3 and Output 8.6.5, you see that the task that would have taken a long time with a single thread of execution on a single machine (around half an hour) can be performed in a much shorter time (around 17 seconds) by using the computational resources of the grid appliance to combine the power of multithreaded and distributed computing.

Output 8.6.6 Performance Information with Distributed Computing and Multithreading

Performance Information	
Host Node	<< your grid host >>
Install Location	<< your grid install location >>
Execution Mode	Distributed
Number of Compute Nodes	16
Number of Threads per Node	16

Procedure Task Timing		
Task	**Seconds**	**Percent**
Load and Prepare Models	0.82	4.23%
Load and Prepare Data	0.04	0.23%
Initialize Parameters	0.86	4.41%
Estimate Parameters	16.86	86.79%
Compute Fit Statistics	0.84	4.34%

The machines that were used to obtain these performance results are relatively modest machines, and PROC HPSEVERITY was run in a multiuser environment; that is, background processes were running in single-machine mode or other users were using the grid in distributed mode. For time-critical applications, you can use a larger, dedicated grid that consists of more powerful machines to achieve more dramatic performance improvement.

Example 8.7: Estimating Parameters Using the Cramér–von Mises Estimator

PROC HPSEVERITY enables you to estimate model parameters by minimizing your own objective function. This example illustrates how you can use PROC HPSEVERITY to implement the Cramér–von Mises estimator. Let $F(y_i;\Theta)$ denote the estimate of CDF at y_i for a distribution with parameters Θ, and let $F_n(y_i)$

denote the empirical estimate of CDF (EDF) at y_i that is computed from a sample y_i, $1 \leq i \leq N$. Then, the Cramér–von Mises estimator of the parameters is defined as

$$\hat{\Theta} = \arg\min_{\Theta} \sum_{i=1}^{N} (F(y_i; \Theta) - F_n(y_i))^2$$

This estimator belongs to the class of minimum distance estimators. It attempts to estimate the parameters such that the squared distance between the CDF and EDF estimates is minimized.

The following PROC HPSEVERITY step uses the Cramér–von Mises estimator to fit four candidate distribution models, including the LOGNGPD mixed-tail distribution model that was defined in "Example 8.3: Defining a Model for Mixed-Tail Distributions" on page 382. The input sample is the same as is used in that example.

```
/*--- Set the search path for functions defined with PROC FCMP ---*/
options cmplib=(work.sevexmpl);

/*-------- Fit LOGNGPD model with PROC HPSEVERITY by using -------
  -------- the Cramer-von Mises minimum distance estimator -------*/
proc hpseverity data=testmixdist obj=cvmobj print=all;
   loss y;
   dist logngpd burr logn gpd;

   * Cramer-von Mises estimator (minimizes the distance *
   * between parametric and nonparametric estimates)    *;
   cvmobj = _cdf_(y);
   cvmobj = (cvmobj -_edf_(y))**2;
run;
```

The OBJ= option in the PROC HPSEVERITY statement specifies that the objective function cvmobj should be minimized. The programming statements compute the contribution of each observation in the input data set to the objective function cvmobj. The use of keyword functions _CDF_ and _EDF_ makes the program applicable to all the distributions.

Some of the key results prepared by PROC HPSEVERITY are shown in Output 8.7.1. The "Model Selection" table indicates that all models converged. When you specify a custom objective function, the default selection criterion is the value of the custom objective function. The "All Fit Statistics" table indicates that LOGNGPD is the best distribution according to all the statistics of fit. Comparing the fit statistics of Output 8.7.1 with those of Output 8.3.1 indicates that the use of the Cramér–von Mises estimator has resulted in smaller values for all the EDF-based statistics of fit for all the models, which is expected from a minimum distance estimator.

Output 8.7.1 Summary of Cramér–von Mises Estimation

The HPSEVERITY Procedure

Input Data Set	
Name	WORK.TESTMIXDIST
Label	Lognormal Body-GPD Tail Sample

Output 8.7.1 *continued*

Model Selection			
Distribution	**Converged**	**cvmobj**	**Selected**
logngpd	Yes	0.02694	Yes
Burr	Yes	0.03325	No
Logn	Yes	0.03633	No
Gpd	Yes	2.96090	No

All Fit Statistics

Distribution	cvmobj		-2 Log Likelihood		AIC		AICC		BIC		KS		AD		CvM
logngpd	0.02694	*	419.49635	*	429.49635	*	430.13464	*	442.52220	*	0.51332	*	0.21563	*	0.03030	*
Burr	0.03325		436.58823		442.58823		442.83823		450.40374		0.53084		0.82875		0.03807	
Logn	0.03633		491.88659		495.88659		496.01030		501.09693		0.52469		2.08312		0.04173	
Gpd	2.96090		560.35409		564.35409		564.47780		569.56443		2.99095		15.51378		2.97806	

Note: The asterisk (*) marks the best model according to each column's criterion.

Example 8.8: 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

$$f(x;\Theta,\mathbf{p}) = \sum_{i=1}^{K} p_i\, f_i(x;\Theta_i)$$

$$F(x;\Theta,\mathbf{p}) = \sum_{i=1}^{K} p_i\, F_i(x;\Theta_i)$$

where Θ_i denotes the parameters of component distribution i and Θ denotes the parameters of the mixture distribution, which is a union of all the Θ_i parameters. $\mathbf{p}$ 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 θ_1 and θ_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 $\mathbf{p}$. The PDF of the mixture of these two distributions can be written as

$$f(x; \theta_1, \theta_2, \Phi, p) = \frac{p}{\theta_1} f_1(\frac{x}{\theta_1}; \Phi_1) + \frac{1-p}{\theta_2} f_2(\frac{x}{\theta_2}; \Phi_2)$$

where Φ_1 and Φ_2 denote the sets of nonscale parameters of the first and second components, respectively, and Φ denotes a union of Φ_1 and Φ_2. For the mixture to have the scale parameter θ, the PDF must be of the form

$$f(x; \theta, \Phi', p) = \frac{1}{\theta}\left(p f_1(\frac{x}{\theta}; \Phi'_1) + (1-p) f_2(\frac{x}{\theta}; \Phi'_2)\right)$$

where Φ', Φ'_1, and Φ'_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 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 ρ 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:

$$f(x; \theta, \rho, \Phi, p) = \frac{1}{\theta}\left(p f_1(\frac{x}{\theta}; \Phi_1) + (1-p) f_2(\frac{x}{\theta}; \rho, \Phi_2)\right)$$
$$F(x; \theta, \rho, \Phi, p) = p F_1(\frac{x}{\theta}; \Phi_1) + (1-p) F_2(\frac{x}{\theta}; \rho, \Phi_2)$$

This can be generalized to a mixture of K components by introducing the $K-1$ ratio parameters ρ_i that relate the scale parameters of each of the K components to the scale parameter θ of the mixture distribution as follows:

$$\theta_1 = \theta$$
$$\theta_i = \rho_i \theta;\ i \in [2, K]$$

In order to illustrate this approach, define a mixture of two lognormal distributions by using the following PDF function:

$$f(x; \mu, \sigma_1, p_2, \rho_2, \sigma_2) = \frac{(1-p_2)}{\sigma_1 x\sqrt{2\pi}} \exp\left(\frac{-(\log(x)-\mu)^2}{2\sigma_1^2}\right) + \frac{p_2}{\sigma_2 x\sqrt{2\pi}} \exp\left(\frac{-(\log(x)-\mu-\log(\rho_2))^2}{2\sigma_2^2}\right)$$

You can verify that μ serves as the log of the scale parameter θ ($\mu = \log(\theta)$).

The following PROC FCMP steps encode this formulation in a distribution named SLOGNMIX2 for use with PROC HPSEVERITY:

```
/*- 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 HPSEVERITY 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)));
   endsub;
```

```
   subroutine slognmix2_lowerbounds(Mu, Sigma1, p2, Rho2, Sigma2);
      outargs Mu, Sigma1, p2, Rho2, Sigma2;
      Mu = .; /* Mu has no lower bound */
      Sigma1 = 0; /* Sigma1 > 0 */
      p2 = 0; /* p2 > 0 */
      Rho2 = 0; /* Rho2 > 0 */
      Sigma2 = 0; /* Sigma2 > 0 */
   endsub;

   subroutine slognmix2_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 HPSEVERITY 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 ρ_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 μ in terms of ρ_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 σ_1 and σ_2:

  $$\sigma_1^2 = 2(\log(m_1) - \mu)$$
  $$\sigma_2^2 = 2(\log(m_1) - \mu - \log(\rho_2))$$

Note that σ_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 ρ_2.

Even when the preceding assumptions are not true for a given 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):

```
/*-------- 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;

/*-- Fit and compare scale regression models with 2-component  --*/
/*-- lognormal mixture and the standard lognormal distribution --*/
options cmplib=(work.sevexmpl);

proc hpseverity data=testlognmix print=all;
   loss y;
   dist slognmix2 logn;
run;
```

The comparison of the fit statistics of SLOGNMIX2 and LOGN, as shown in Output 8.8.1, confirms that the two-component mixture is certainly a better fit to these data than the single lognormal distribution.

Output 8.8.1 Comparison of Fitting One versus Two Lognormal Components to Mixture Data

The HPSEVERITY Procedure

All Fit Statistics

Distribution	-2 Log Likelihood		AIC		AICC		BIC		KS		AD		CvM	
slognmix2	38343	*	38353	*	38353	*	38386	*	0.52221	*	0.19843	*	0.02728	*
Logn	39073		39077		39077		39090		5.86522		66.93414		11.72703	

Note: The asterisk (*) marks the best model according to each column's criterion.

The detailed results for the SLOGNMIX2 distribution are shown in Output 8.8.2. According to the "Initial Parameter Values and Bounds" table, the initial value of ρ_2 is not 0.5, indicating that a linear search was conducted to ensure $\log(m_1) > \mu$.

Output 8.8.2 Detailed Estimation Results for the SLOGNMIX2 Distribution

The HPSEVERITY Procedure
slognmix2 Distribution

Distribution Information	
Name	slognmix2
Description	Mixture of two lognormals with a log-scale parameter Mu
Distribution Parameters	5

Initial Parameter Values and Bounds

Parameter	Initial Value	Lower Bound	Upper Bound
Mu	2.92006	-Infty	Infty
Sigma1	0.10455	1.05367E-8	Infty
P2	0.50000	1.05367E-8	1.00000
Rho2	0.72000	1.05367E-8	1.00000
Sigma2	0.81728	1.05367E-8	Infty

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

Optimization Summary	
Optimization Technique	Trust Region
Iterations	7
Function Calls	18
Log Likelihood	-19171.5

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 8.9: Predicting Mean and Value-at-Risk by Using Scoring Functions

If you work in the risk management department of an insurance company or a bank, then one of your primary applications of severity loss distribution models is to predict the value-at-risk (VaR) so that there is a very low probability of experiencing a loss value that is greater than the VaR. The probability level at which VaR

is measured is prescribed by industry regulations such as Basel III and Solvency II. The VaR level is usually specified in terms of $(1 - \alpha)$, where $\alpha \in (0, 1)$ is the probability that a loss value exceeds the VaR. Typical VaR levels are 0.95, 0.975, and 0.995.

In addition to predicting the VaR, which is regarded as an estimate of the worst-case loss, businesses are often interested in predicting the average loss by estimating either the mean or median of the distribution.

The estimation of the mean and VaR combined with the scale regression model is very potent tool for analyzing worst-case and average losses for various scenarios. For example, if the regressors that are used in a scale regression model represent some key macroeconomic and operational indicators, which are widely referred to as key risk indicators (KRIs), then you can analyze the VaR and mean loss estimates over various values for the KRIs to get a more comprehensive picture of the risk profile of your organization across various market and internal conditions.

This example illustrates the use of scoring functions to simplify the process of predicting the mean and VaR of scale regression models.

To compute the mean, you need to ensure that the function to compute the mean of a distribution is available in the function library. If you define and fit your own distribution and you want to compute its mean, then you need to use the FCMP procedure to define that function and you need to use the CMPLIB= system option to specify the location of that function. For your convenience, the *dist*_MEAN function (which computes the mean of the *dist* distribution) is already defined in the Sashelp.Svrtdist library for each of the 10 predefined distributions. The following statements display the definitions of MEAN functions of all distributions. Note that the MEAN functions for the Burr, Pareto, and generalized Pareto distributions check the existence of the first moment for specified parameter values.

```
/*--------- Definitions distribution functions that compute the mean ----------*/
proc fcmp library=sashelp.svrtdist outlib=work.means.scalemod;
   function BURR_MEAN(x, Theta, Alpha, Gamma);
      if not(Alpha * Gamma > 1) then
         return (.); /* first moment does not exist */
      return (Theta*gamma(1 + 1/Gamma)*gamma(Alpha - 1/Gamma)/gamma(Alpha));
   endsub;
   function EXP_MEAN(x, Theta);
      return (Theta);
   endsub;
   function GAMMA_MEAN(x, Theta, Alpha);
      return (Theta*Alpha);
   endsub;
   function GPD_MEAN(x, Theta, Xi);
      if not(Xi < 1) then
         return (.); /* first moment does not exist */
      return (Theta/(1 - Xi));
   endsub;
   function IGAUSS_MEAN(x, Theta, Alpha);
      return (Theta);
   endsub;
   function LOGN_MEAN(x, Mu, Sigma);
      return (exp(Mu + Sigma*Sigma/2.0));
   endsub;
```

```
   function PARETO_MEAN(x, Theta, Alpha);
      if not(Alpha > 1) then
         return (.); /* first moment does not exist */
      return (Theta/(Alpha - 1));
   endsub;
   function STWEEDIE_MEAN(x, Theta, Lambda, P);
      return (Theta* Lambda * (2 - P) / (P - 1));
   endsub;
   function TWEEDIE_MEAN(x, P, Mu, Phi);
      return (Mu);
   endsub;
   function WEIBULL_MEAN(x, Theta, Tau);
      return (Theta*gamma(1 + 1/Tau));
   endsub;
quit;
```

For your further convenience, the *dist*_QUANTILE function (which computes the quantile of the *dist* distribution) is also defined in the Sashelp.Svrtdist library for each of the 10 predefined distributions. Because the MEAN and QUANTILE functions satisfy the definition of a distribution function as described in the section "Formal Description" on page 353, you can submit the following PROC HPSEVERITY step to fit all regression-friendly predefined distributions and generate the scoring functions for the MEAN, QUANTILE, and other distribution functions:

```
/*----- Fit all distributions and generate scoring functions ------*/
proc hpseverity data=test_sev9 outest=est print=all;
   loss y;
   scalemodel x1-x5;
   dist _predefined_ stweedie;
   outscorelib outlib=scorefuncs commonpackage;
run;
```

The SAS statements that simulate the sample in the Work.Test_sev9 data set are available in the PROC HPSEVERITY sample program *hsevex09.sas*. The OUTLIB= option in the OUTSCORELIB statement requests that the scoring functions be written to the Work.Scorefuncs library, and the COMMONPACKAGE option in the OUTSCORELIB statement requests that all the functions be written to the same package. Upon completion, PROC HPSEVERITY sets the CMPLIB system option to the following value:

```
(sashelp.svrtdist work.scorefuncs)
```

The "All Fit Statistics" table in Output 8.9.1 shows that the lognormal distribution's scale model is the best and the inverse Gaussian's scale model is a close second according to the likelihood-based statistics.

You can examine the scoring functions that are written to the Work.Scorefuncs library by using the FCMP Function Editor, which is available in the Display Manager session of Base SAS when you select **Solutions→Analysis** from the main menu. For example, PROC HPSEVERITY automatically generates and submits the following PROC FCMP statements to define the scoring functions SEV_MEAN_LOGN and SEV_QUANTILE_IGAUSS:

```
proc fcmp library=(sashelp.svrtdist) outlib=work.scorefuncs.sevfit;
   function SEV_MEAN_LOGN(y, x{*});
      _logscale_=0;
      _logscale_ = _logscale_ + ( 7.64722278930350E-01 * x{1});
      _logscale_ = _logscale_ + ( 2.99209540369860E+00 * x{2});
      _logscale_ = _logscale_ + (-1.00788916253430E+00 * x{3});
```

```
      _logscale_ = _logscale_ + ( 2.58883602184890E-01 * x{4});
      _logscale_ = _logscale_ + ( 5.00927479793970E+00 * x{5});
      _logscale_ = _logscale_ + ( 9.95078833050690E-01);
      return (LOGN_MEAN(y, _logscale_,  2.31592981635590E-01));
   endsub;

   function SEV_QUANTILE_IGAUSS(y, x{*});
      _logscale_=0;
      _logscale_ = _logscale_ + ( 7.64581738373520E-01 * x{1});
      _logscale_ = _logscale_ + ( 2.99159055015310E+00 * x{2});
      _logscale_ = _logscale_ + (-1.00793496641510E+00 * x{3});
      _logscale_ = _logscale_ + ( 2.58870460543840E-01 * x{4});
      _logscale_ = _logscale_ + ( 5.00996884646730E+00 * x{5});
      _scale_ =  2.77854870591020E+00 * exp(_logscale_);
      return (IGAUSS_QUANTILE(y, _scale_,  1.81511227238720E+01));
   endsub;
quit;
```

Output 8.9.1 Comparison of Fitted Scale Models for Mean and VaR Illustration

The HPSEVERITY Procedure

All Fit Statistics

Distribution	-2 Log Likelihood		AIC		AICC		BIC		KS		AD		CvM
stweedie	460.65755		476.65755		476.95083		510.37442		10.44549		4765		37.07708	
Burr	451.42238		467.42238		467.71565		501.13924		10.32782		4431		37.19808	
Exp	1515		1527		1527		1552		8.85827		2062		23.98267	
Gamma	448.28222		462.28222		462.50986		491.78448		10.42272		6068		37.19450	
Igauss	444.44512		458.44512		458.67276		487.94738		10.33028		6257		37.30880	
Logn	444.43670	*	458.43670	*	458.66434	*	487.93895	*	10.37035		6155		37.18553	
Pareto	1515		1529		1529		1559		8.85775	*	2061	*	23.98149	*
Gpd	1515		1529		1529		1559		8.85827		2062		23.98267	
Weibull	527.28676		541.28676		541.51440		570.78902		10.48084		4947		36.36039	

Note: The asterisk (*) marks the best model according to each column's criterion.

PROC HPSEVERITY detects all the distribution functions that are available in the current CMPLIB= search path (which always includes the Sashelp.Svrtdist library) for the distributions that you specify in the DIST statement, and it creates the corresponding scoring functions. You can define any distribution function that has the desired signature to compute an estimate of your choice, include its library in the CMPLIB= system option, and then specify the OUTSCORELIB statement to generate the corresponding scoring functions. Specifying the COMMONPACKAGE option in the OUTSCORELIB statement causes the name of the scoring function to take the form SEV_*function-suffix_dist*. If you do not specify the COMMONPACKAGE option, PROC HPSEVERITY creates a scoring function named SEV_*function-suffix* in a package named *dist*. You can invoke functions from a specific package only inside the FCMP procedure. If you want to invoke the scoring functions from a DATA step, then it is recommended that you specify the COMMONPACKAGE option when you specify multiple distributions in the DIST statement.

To illustrate the use of scoring functions, let Work.Reginput contain the scoring data, where the values of regressors in each observation define one scenario. Scoring functions make it very easy to compute the mean and VaR of each distribution's scale model for each of the scenarios, as the following steps illustrate for the lognormal and inverse Gaussian distributions by using a VaR level of 97.5%:

```
/*--- Set VaR level ---*/
%let varLevel=0.975;

/*--- Compute scores (mean and var) for the       ---
  --- scoring data by using the scoring functions ---*/
data scores;
   array x{*} x1-x5;
   set reginput;

   igauss_mean = sev_mean_igauss(., x);
   igauss_var  = sev_quantile_igauss(&varLevel, x);
   logn_mean   = sev_mean_logn(., x);
   logn_var    = sev_quantile_logn(&varLevel, x);
run;
```

The following DATA step accomplishes the same task by reading the parameter estimates that were written to the Work.Est data set by the previous PROC HPSEVERITY step:

```
/*--- Compute scores (mean and var) for the       ---
  --- scoring data by using the OUTEST= data set  ---*/
data scoresWithOutest(keep=x1-x5 igauss_mean igauss_var logn_mean logn_var);
   array _x_{*} x1-x5;
   array _xparmIgauss_{5} _temporary_;
   array _xparmLogn_{5} _temporary_;

   retain _Theta0_ Alpha0;
   retain _Mu0_ Sigma0;
   *--- read parameter estimates for igauss and logn models ---*;
   if (_n_ = 1) then do;
      set est(where=(upcase(_MODEL_)='IGAUSS' and _TYPE_='EST'));
      _Theta0_ = Theta; Alpha0 = Alpha;
      do _i_=1 to dim(_x_);
         if (_x_(_i_) = .R) then _xparmIgauss_(_i_) = 0;
         else _xparmIgauss_(_i_) = _x_(_i_);
      end;

      set est(where=(upcase(_MODEL_)='LOGN' and _TYPE_='EST'));
      _Mu0_ = Mu; Sigma0 = Sigma;
      do _i_=1 to dim(_x_);
         if (_x_(_i_) = .R) then _xparmLogn_(_i_) = 0;
         else _xparmLogn_(_i_) = _x_(_i_);
      end;
   end;

   set reginput;

   *---  predict mean and VaR for inverse Gaussian  ---*;
   * first compute X'*beta for inverse Gaussian *;
   _xbeta_ = 0.0;
```

```
    do _i_ = 1 to dim(_x_);
       _xbeta_ = _xbeta_ + _xparmIgauss_(_i_) * _x_(_i_);
    end;
    * now compute scale for inverse Gaussian *;
    _SCALE_ = _Theta0_ * exp(_xbeta_);
    igauss_mean = igauss_mean(., _SCALE_, Alpha0);
    igauss_var = igauss_quantile(&varLevel, _SCALE_, Alpha0);

    *---  predict mean and VaR for lognormal         ---*;
    * first compute X'*beta for lognormal*;
    _xbeta_ = 0.0;
    do _i_ = 1 to dim(_x_);
       _xbeta_ = _xbeta_ + _xparmLogn_(_i_) * _x_(_i_);
    end;
    * now compute Mu=log(scale) for lognormal *;
    _MU_ = _Mu0_ + _xbeta_;
    logn_mean = logn_mean(., _MU_, Sigma0);
    logn_var = logn_quantile(&varLevel, _MU_, Sigma0);
run;
```

The "Values Comparison Summary" table in Output 8.9.2 shows that the difference between the estimates that are produced by both methods is within the acceptable machine precision. However, the comparison of the DATA step complexity of each method clearly shows that the method that uses the scoring functions is much easier because it saves a lot of programming effort. Further, new distribution functions, such as the *dist*_MEAN functions that are illustrated here, are automatically discovered and converted to scoring functions by PROC HPSEVERITY. That enables you to focus your efforts on writing the distribution function that computes your desired score, which needs to be done only once. Then, you can create and use the corresponding scoring functions multiple times with much less effort.

Output 8.9.2 Comparison of Mean and VaR Estimates of Two Scoring Methods

```
                        The COMPARE Procedure
              Comparison of WORK.SCORESWITHOUTEST with WORK.SCORES
                  (Method=RELATIVE(0.0222), Criterion=1.0E-12)

NOTE: All values compared are within the equality criterion used. However, 40
      of the values compared are not exactly equal.
```

Example 8.10: 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 Work.Losses data set. Let the data set 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 SAS statements that simulate an example of the Work.Losses data set are available in the PROC HPSEVERITY sample program *hsevex10.sas*.

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 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 313. 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 HPSEVERITY step fits the scale regression model with a lognormal distribution to data in the Work.Losses data set, and stores the model and parameter estimate information in the Work.EstStore item store:

```
/* Fit scale regression model with different types of regression effects */
proc hpseverity data=losses outstore=eststore
     print=all plots=none;
   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 HPSEVERITY 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). For more information about effect types, see Table 8.9, "GLM Parameterization of Classification Variables and Effects," on page 316.

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 8.10.1. 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 8.10.1 Class Level Information Table

The HPSEVERITY Procedure

Class Level Information		
Class	**Levels**	**Values**
carType	2	SUV Sedan
gender	2	Female Male
education	3	AdvancedDegree College High School

The regression modeling results for the lognormal distribution are shown in Output 8.10.2. The "Initial Parameter Values and Bounds" table is important especially because the preceding PROC HPSEVERITY 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 8.10.2 Initial Values for the Scale Regression Model with Class and Interaction Effects

Initial Parameter Values and Bounds			
Parameter	**Initial Value**	**Lower Bound**	**Upper Bound**
Mu	4.88526	-709.78271	709.78271
Sigma	0.51283	1.05367E-8	Infty
carType SUV	0.56953	-709.78271	709.78271
carType Sedan	Redundant		
gender Female	0.41154	-709.78271	709.78271
gender Male	Redundant		
carSafety	-0.72742	-709.78271	709.78271
income	-0.33216	-709.78271	709.78271
carType*education SUV AdvancedDegree	0.31686	-709.78271	709.78271
carType*education SUV College	0.66361	-709.78271	709.78271
carType*education SUV High School	Redundant		
carType*education Sedan AdvancedDegree	-0.47841	-709.78271	709.78271
carType*education Sedan College	-0.25968	-709.78271	709.78271
carType*education Sedan High School	Redundant		
income*gender Female	-0.02112	-709.78271	709.78271
income*gender Male	Redundant		
carSafety*income	0.13084	-709.78271	709.78271

The convergence and optimization summary information in Output 8.10.3 indicates that the scale regression model for the lognormal distribution has converged with the default optimization technique in five iterations.

Output 8.10.3 Optimization Summary for the Scale Regression Model with Class and Interaction Effects

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

Optimization Summary	
Optimization Technique	Trust Region
Iterations	5
Function Calls	14
Log Likelihood	-8286.8

The "Parameter Estimates" table in Output 8.10.4 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.

Output 8.10.4 Parameter Estimates for the Scale Regression with Class and Interaction Effects

Parameter Estimates					
Parameter	**DF**	**Estimate**	**Standard Error**	**t Value**	**Approx Pr > |t|**
Mu	1	5.08874	0.05768	88.23	<.0001
Sigma	1	0.55774	0.01119	49.86	<.0001
carType SUV	1	0.62459	0.04452	14.03	<.0001
carType Sedan	0	0	.	.	.
gender Female	1	0.44145	0.04885	9.04	<.0001
gender Male	0	0	.	.	.
carSafety	1	-0.82942	0.08371	-9.91	<.0001
income	1	-0.35212	0.07657	-4.60	<.0001
carType*education SUV AdvancedDegree	1	0.39393	0.07351	5.36	<.0001
carType*education SUV College	1	0.76532	0.05723	13.37	<.0001
carType*education SUV High School	0	0	.	.	.
carType*education Sedan AdvancedDegree	1	-0.61064	0.05387	-11.34	<.0001
carType*education Sedan College	1	-0.35210	0.03942	-8.93	<.0001
carType*education Sedan High School	0	0	.	.	.
income*gender Female	1	-0.01486	0.06629	-0.22	0.8226
income*gender Male	0	0	.	.	.
carSafety*income	1	0.07045	0.11447	0.62	0.5383

If you want to update the model when new claims data arrive, then you can potentially speed up the estimation process by specifying the OUTSTORE= item store that is created by the preceding PROC HPSEVERITY step as an INSTORE= item store in a new PROC HPSEVERITY step as follows:

```
/* Refit scale regression model on new data different types of regression effects */
proc hpseverity data=withNewLosses instore=eststore print=all plots=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;
```

PROC HPSEVERITY uses the parameter estimates in the INSTORE= item store to initialize the distribution and regression parameters.

References

Burr, I. W. (1942). "Cumulative Frequency Functions." *Annals of Mathematical Statistics* 13:215–232.

D'Agostino, R. B., and Stephens, M., eds. (1986). *Goodness-of-Fit Techniques*. New York: Marcel Dekker.

Danielsson, J., de Haan, L., Peng, L., and de Vries, C. G. (2001). "Using a Bootstrap Method to Choose the Sample Fraction in Tail Index Estimation." *Journal of Multivariate Analysis* 76:226–248.

Dunn, P. K., and Smyth, G. K. (2005). "Series Evaluation of Tweedie Exponential Dispersion Model Densities." *Statistics and Computing* 15:267–280.

Frydman, H. (1994). "A Note on Nonparametric Estimation of the Distribution Function from Interval-Censored and Truncated Observations." *Journal of the Royal Statistical Society, Series B* 56:71–74.

Gentleman, R., and Geyer, C. J. (1994). "Maximum Likelihood for Interval Censored Data: Consistency and Computation." *Biometrika* 81:618–623.

Greenwood, M. (1926). "The Natural Duration of Cancer." In *Reports of Public Health and Related Subjects*, vol. 33, 1–26. London: Her Majesty's Stationery Office.

Hill, B. M. (1975). "A Simple General Approach to Inference about the Tail of a Distribution." *Annals of Statistics* 3:1163–1173.

Jørgensen, B. (1987). "Exponential Dispersion Models." *Journal of the Royal Statistical Society, Series B* 49:127–162. With discussion.

Kaplan, E. L., and Meier, P. (1958). "Nonparametric Estimation from Incomplete Observations." *Journal of the American Statistical Association* 53:457–481.

Klein, J. P., and Moeschberger, M. L. (1997). *Survival Analysis: Techniques for Censored and Truncated Data*. New York: Springer-Verlag.

Klugman, S. A., Panjer, H. H., and Willmot, G. E. (1998). *Loss Models: From Data to Decisions*. New York: John Wiley & Sons.

Koziol, J. A., and Green, S. B. (1976). "A Cramér–von Mises Statistic for Randomly Censored Data." *Biometrika* 63:466–474.

Lai, T. L., and Ying, Z. (1991). "Estimating a Distribution Function with Truncated and Censored Data." *Annals of Statistics* 19:417–442.

Lynden-Bell, D. (1971). "A Method of Allowing for Known Observational Selection in Small Samples Applied to 3CR Quasars." *Monthly Notices of the Royal Astronomical Society* 155:95–118.

Rodriguez, R. N. (2006). "Burr Distributions." In *Encyclopedia of Statistical Sciences*, 2nd ed., vol. 1, edited by S. Kotz, N. Balakrishnan, C. B. Read, B. Vidakovic, and N. L. Johnson. New York: John Wiley & Sons.

Searle, S. R. (1971). *Linear Models*. New York: John Wiley & Sons.

Turnbull, B. W. (1976). "The Empirical Distribution Function with Arbitrarily Grouped, Censored, and Truncated Data." *Journal of the Royal Statistical Society, Series B* 38:290–295.

Tweedie, M. C. K. (1984). "An Index Which Distinguishes between Some Important Exponential Families." In *Statistics: Applications and New Directions—Proceedings of the Indian Statistical Institute Golden Jubilee International Conference*, edited by J. K. Ghosh and J. Roy, 579–604. Calcutta: Indian Statistical Institute.

Subject Index

Syntax Index