SAS/ETS® 13.2 User's Guide

The SEVERITY Procedure

This document is an individual chapter from *SAS/ETS® 13.2 User's Guide*.

The correct bibliographic citation for the complete manual is as follows: SAS Institute Inc. 2014. *SAS/ETS® 13.2 User's Guide*. Cary, NC: SAS Institute Inc.

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

August 2014

Chapter 23
The SEVERITY Procedure

Contents

Overview: SEVERITY Procedure

The SEVERITY 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 SEVERITY 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 SEVERITY provides a default set of probability distribution models that includes the Burr, exponential, gamma, generalized Pareto, inverse Gaussian (Wald), lognormal, Pareto, Tweedie, and Weibull distributions. In the simplest form, you can estimate the parameters of any of these distributions by using a list of severity values that are recorded in a SAS data set. You can optionally group the values by a set of BY variables. PROC SEVERITY 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 SEVERITY 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).

A high-performance version of PROC SEVERITY is available as the HPSEVERITY procedure in the SAS High-Performance Econometrics product. The following key features make PROC SEVERITY and PROC HPSEVERITY unique among SAS procedures that can estimate continuous probability distributions:

- Both procedures enable 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 SEVERITY 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.

Both procedures use an appropriate estimator of the empirical distribution function (EDF). EDF is required to compute the KS, AD, and CvM statistics-of-fit. The procedures also provide 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. Both procedures compute the standard errors for all EDF estimators.

- Both procedures enable 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 SEVERITY and HPSEVERITY procedures can fit the distribution model defined by them.

- Both procedures 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.

- Both procedures enable 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.

- Both procedures enable 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.

- Both procedures use multithreading to significantly reduce the time it takes to fit a distribution model.

Getting Started: SEVERITY Procedure

This section outlines the use of the SEVERITY 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 SEVERITY 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 severity data=test_sev1 crit=aicc;
   loss y;
   dist _predefined_;
run;
```

The PROC SEVERITY 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 23.1 through Figure 23.5. First, information about the input data set is displayed followed by the "Model Selection" table, as shown in Figure 23.1. 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 23.1 Data Set Information and Model Selection Table

The SEVERITY Procedure

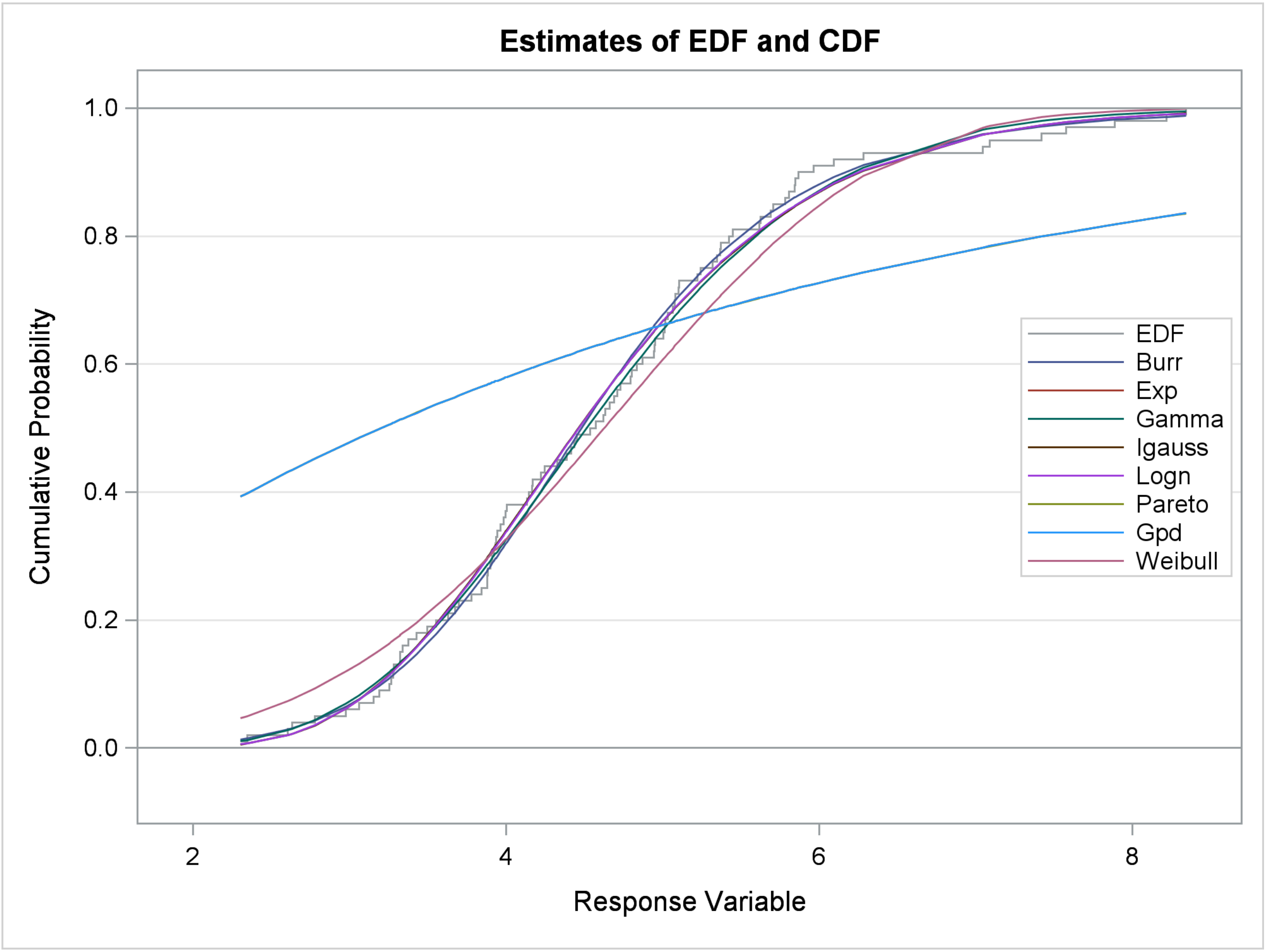
Input Data Set	
Name	WORK.TEST_SEV1
Label	Simple Lognormal Sample

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

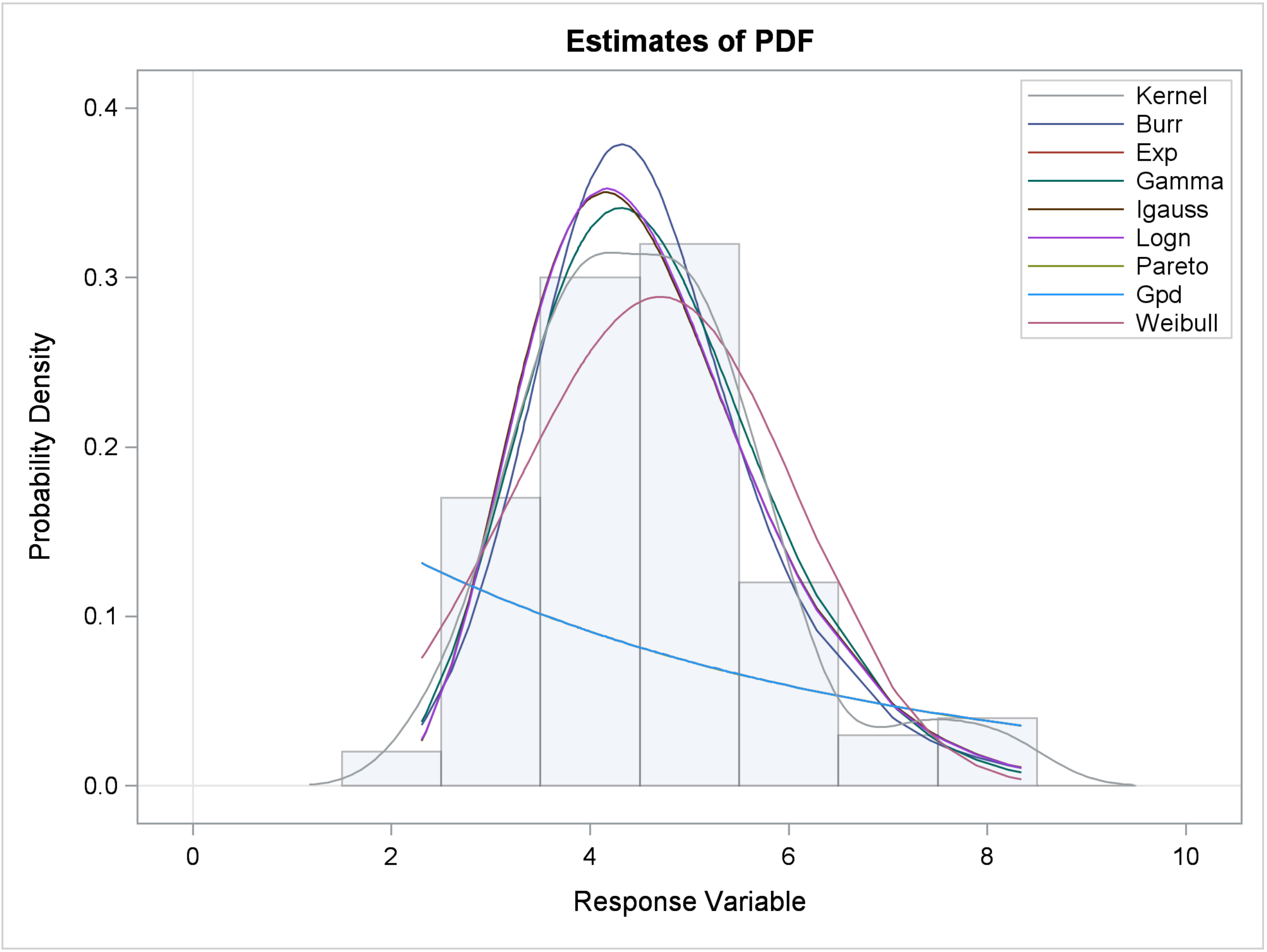
Next, two comparative plots are prepared. These plots enable you to visually verify how the models differ from each other and from the nonparametric estimates. The plot in Figure 23.2 displays the cumulative distribution function (CDF) estimates of all the models and the estimates of the empirical distribution function (EDF). The CDF plot indicates that the Exp (exponential), Pareto, and Gpd (generalized Pareto) distributions are a poor fit as compared to the EDF estimate. The Weibull distribution is also a poor fit, although not as poor as exponential, Pareto, and Gpd. The other four distributions seem to be quite close to each other and to the EDF estimate.

Figure 23.2 Comparison of EDF and CDF Estimates of the Fitted Models



The plot in Figure 23.3 displays the probability density function (PDF) estimates of all the models and the nonparametric kernel and histogram estimates. The PDF plot enables better visual comparison between the Burr, Gamma, Igauss (inverse Gaussian), and Logn (lognormal) models. The Burr and Gamma differ significantly from the Igauss and Logn distributions in the central portion of the range of Y values, while the latter two fit the data almost identically. This provides a visual confirmation of the information in the “Model Selection” table of Figure 23.1, which indicates that the AICC values of Igauss and Logn distributions are very close.

Figure 23.3 Comparison of PDF Estimates of the Fitted Models



The comparative plots are followed by the estimation information for each of the candidate models. The information for the lognormal model, which is the best fitting model, is shown in Figure 23.4. 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 SEVERITY 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 23.4 Estimation Details for the Lognormal Model

The SEVERITY Procedure
Logn Distribution

Distribution Information	
Name	Logn
Description	Lognormal Distribution
Distribution Parameters	2

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	Estimate	Standard Error	t Value	Approx Pr > |t|
Mu	1.49605	0.02651	56.43	<.0001
Sigma	0.26243	0.01874	14.00	<.0001

The parameter estimates of the Burr distribution are shown in Figure 23.5. These estimates are used in the next example.

Figure 23.5 Parameter Estimates for the Burr Model

Parameter Estimates

Parameter	Estimate	Standard Error	t Value	Approx Pr > |t|
Theta	4.62348	0.46181	10.01	<.0001
Alpha	1.15706	0.47493	2.44	0.0167
Gamma	6.41227	0.99039	6.47	<.0001

An Example with Left-Truncation and Right-Censoring

PROC SEVERITY 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 severity data=test_sev2 crit=aicc
       print=all plots=(cdfperdist pp qq);
   loss y / lt=threshold rc=limit;

   dist logn burr gamma weibull;
run;
```

The LOSS statement specifies the left-truncation and right-censoring variables. The DIST statement specifies the candidate distributions. The PRINT= option in the PROC SEVERITY statement requests that all the displayed output be prepared. The PLOTS= option in the PROC SEVERITY statement requests that the CDF plot, P-P plot, and Q-Q plot be prepared for each candidate distribution in addition to the default plots.

Some of the key results prepared by PROC SEVERITY are shown in Figure 23.6 through Figure 23.13. In addition to the estimates of the range, mean, and standard deviation of Y, the “Descriptive Statistics for y” table shown in Figure 23.6 also indicates the number of observations that are left-truncated or right-censored. The “Model Selection” table in Figure 23.6 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 23.6 Summary Results for the Truncated and Censored Data

The SEVERITY Procedure

Input Data Set	
Name	WORK.TEST_SEV2
Label	A Lognormal Sample With Censoring and Truncation

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 SEVERITY 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 23.7. It indicates that the lognormal model is chosen by all the criteria.

Figure 23.7 Comparing All Statistics of Fit for the Truncated and Censored Data

All Fit Statistics

Distribution	-2 Log Likelihood	AIC	AICC	BIC	KS	AD
Logn	294.80301 *	298.80301 *	298.92672 *	304.01335 *	0.51824 *	0.34736 *
Burr	296.41229	302.41229	302.66229	310.22780	0.66984	0.36712
Gamma	295.32921	299.32921	299.45293	304.53955	0.62511	0.42921
Weibull	305.14408	309.14408	309.26779	314.35442	0.93307	1.40699

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

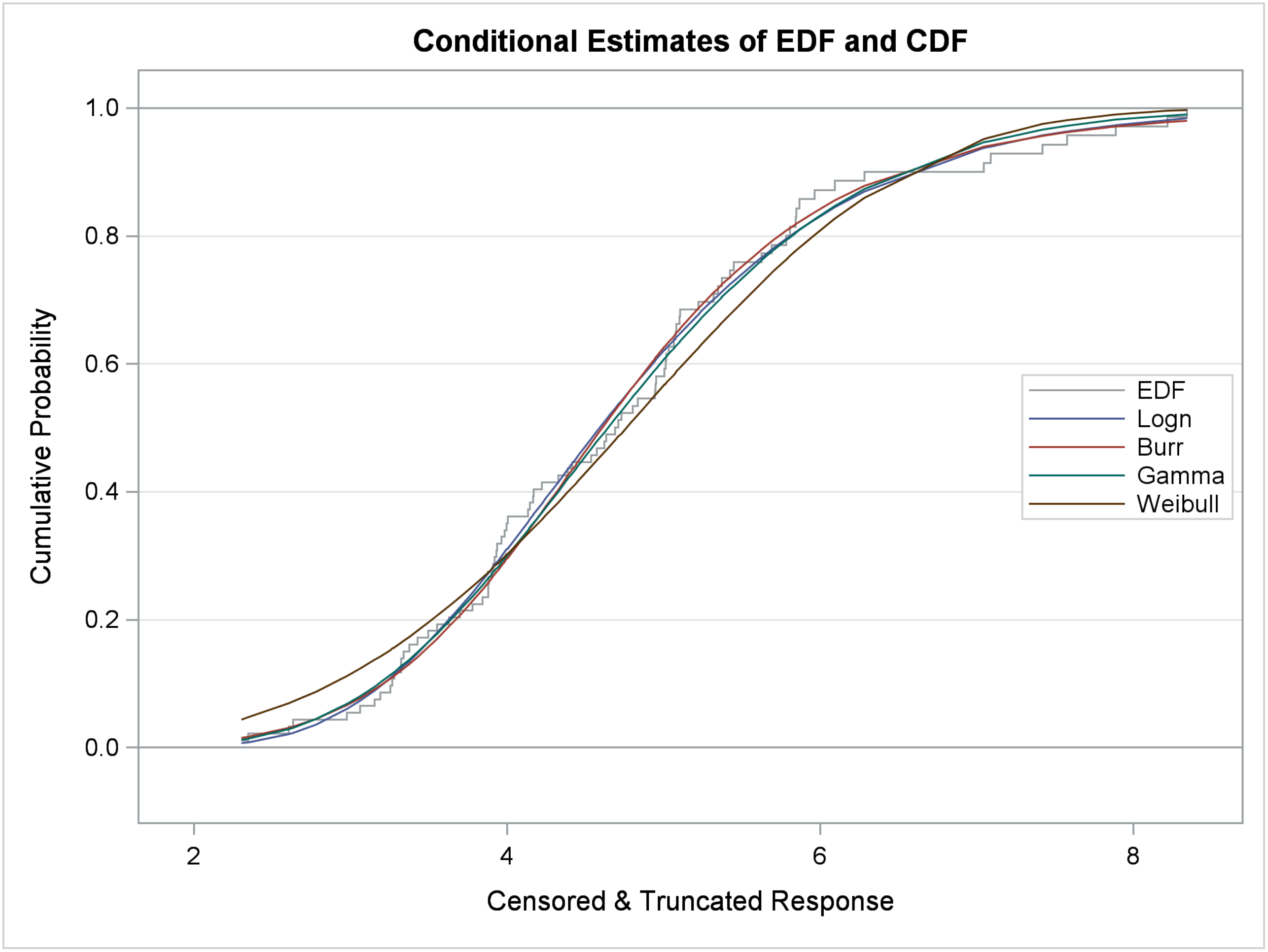
All Fit Statistics

Distribution	CvM
Logn	0.05159 *
Burr	0.05726
Gamma	0.05526
Weibull	0.17465

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

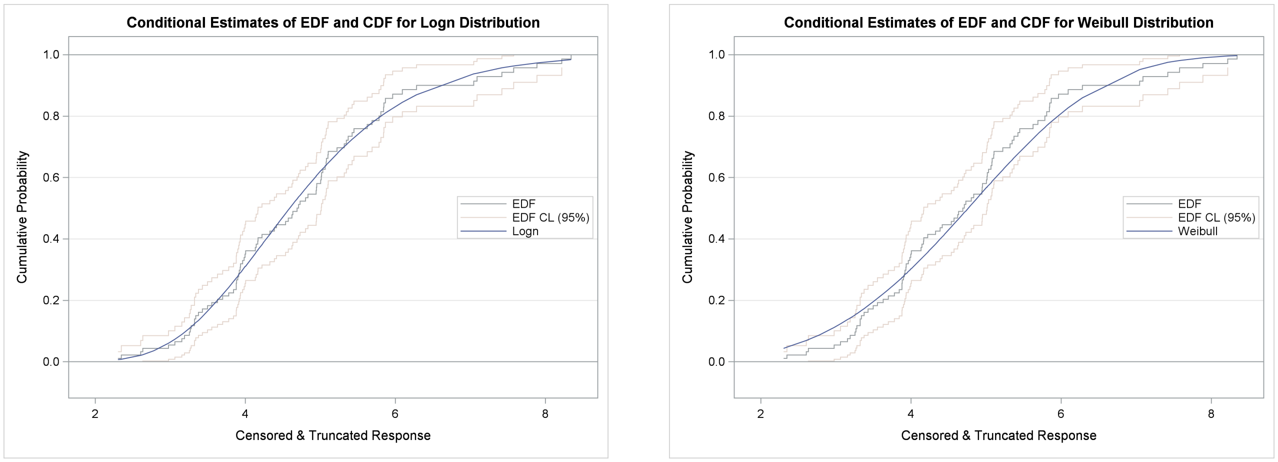
The plot that compares EDF and CDF estimates is shown in Figure 23.8. When you specify left-truncation, both the EDF and CDF estimates are conditional on the response variable being greater than the smallest left-truncation threshold in the sample.

Figure 23.8 EDF and CDF Estimates for the Truncated and Censored Data



When you specify the PLOTS=CDFPERDIST option, PROC SEVERITY prepares a plot that compares the nonparametric EDF estimates with the parametric CDF estimates for each distribution. These plots for lognormal and Weibull distributions are shown in Figure 23.9. These plots also contain the lower and upper confidence limits of EDF for the specified confidence level. Because no confidence level is specified in the EDFALPHA= option in the PROC SEVERITY statement, a default confidence level of 95% is used, which is equivalent to specifying EDFALPHA=0.05. If the CDF estimates lie entirely within the EDF confidence interval, then you can be 95% confident that the parametric and nonparametric estimates are in agreement.

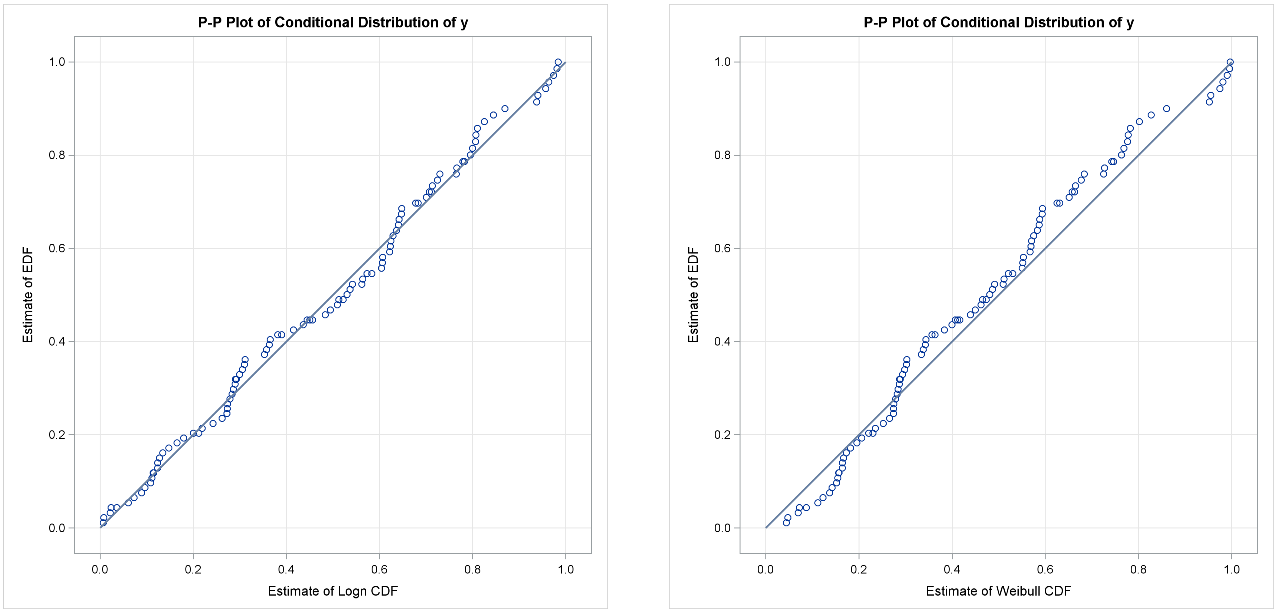
Figure 23.9 Comparing EDF and CDF Estimates for Lognormal and Weibull Models Fitted to Truncated and Censored Data



There are two additional ways to compare nonparametric (empirical) and parametric estimates for each model that has not failed to converge:

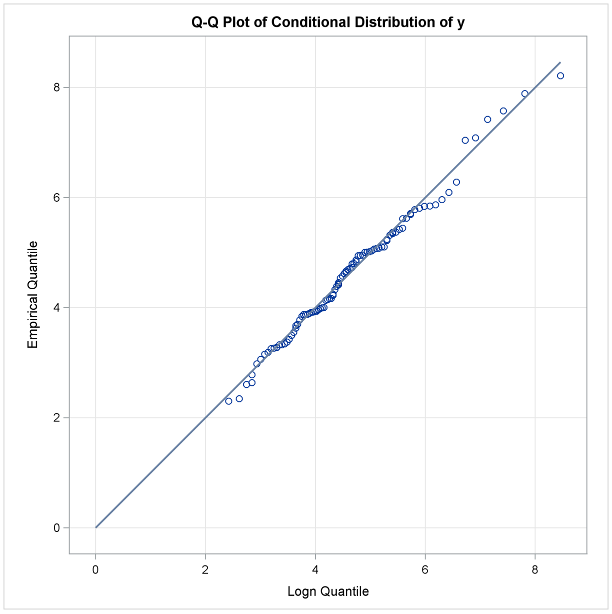
- A P-P plot is a scatter plot of the EDF and the CDF estimates. The model for which the points are scattered closer to the unit-slope reference line is a better fit. The P-P plot for the lognormal distribution is shown in Figure 23.10. It indicates that the EDF and the CDF match very closely. In contrast, the P-P plot for the Weibull distribution, also shown in Figure 23.10, indicates a poor fit.

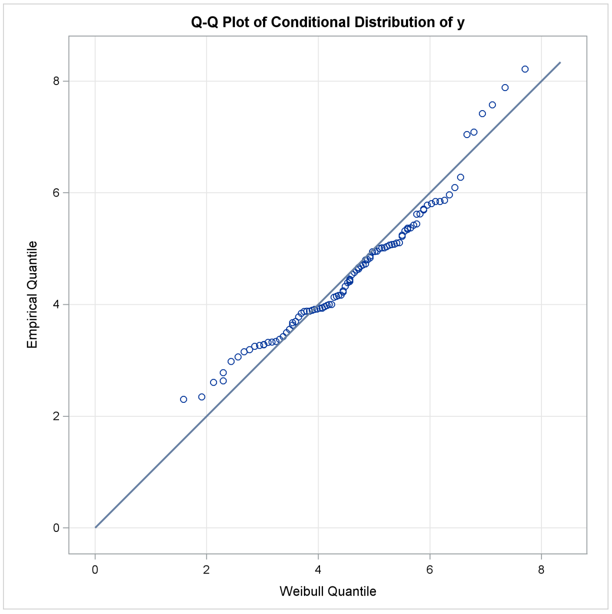
Figure 23.10 P-P Plots for Lognormal and Weibull Models Fitted to Truncated and Censored Data



- A Q-Q plot is a scatter plot of empirical quantiles and the quantiles of a parametric distribution. Like the P-P plot, points scattered closer to the unit-slope reference line indicate a better fit. The Q-Q plots of lognormal and Weibull distributions are shown in Figure 23.11, which confirm the conclusions arrived at by comparing the P-P plots.

Figure 23.11 Q-Q Plots for Lognormal and Weibull Models Fitted to Truncated and Censored Data





Specifying Initial Values for Parameters

All the predefined distributions have parameter initialization functions built into them. For the current example, Figure 23.12 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 23.12 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

Optimization Summary

Optimization Technique	Trust Region
Iterations	8
Function Calls	23
Log Likelihood	-148.20614

Parameter Estimates

Parameter	Estimate	Standard Error	t Value	Approx Pr > |t|
Theta	4.76980	0.62492	7.63	<.0001
Alpha	1.16363	0.58859	1.98	0.0509
Gamma	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 23.5). One of the ways in which you can specify the initial values is as follows:

```
/*------ Specifying initial values using INIT= option -------*/
proc severity data=test_sev2 crit=aicc print=all plots=none;
   loss y / lt=threshold rc=limit;

   dist burr(init=(theta=4.62348 alpha=1.15706 gamma=6.41227));
run;
```

The names of the parameters that are specified in the INIT option must match the parameter names in the definition of the distribution. The results obtained with these initial values are shown in Figure 23.13. 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 23.13 Burr Model Optimization Summary for the Truncated and Censored Data

The SEVERITY Procedure
Burr Distribution

Optimization Summary	
Optimization Technique	Trust Region
Iterations	5
Function Calls	16
Log Likelihood	-148.20614

Parameter Estimates

Parameter	Estimate	Standard Error	t Value	Approx Pr > |t|
Theta	4.76980	0.62492	7.63	<.0001
Alpha	1.16363	0.58859	1.98	0.0509
Gamma	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 SEVERITY 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 SEVERITY 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 SEVERITY step fits the lognormal, Burr, and gamma distribution models to this data. The regressors are specified in the SCALEMODEL statement. The DFMIXTURE= option in the SCALEMODEL statement specifies the method of computing the CDF estimates that are used to compute the EDF-based statistics of fit.

```
proc severity data=test_sev3 crit=aicc print=all;
   loss y;
   scalemodel x1-x3 / dfmixture=full;

   dist logn burr gamma;
run;
```

Some of the key results prepared by PROC SEVERITY are shown in Figure 23.14 through Figure 23.18. The descriptive statistics of all the variables are shown in Figure 23.14.

Figure 23.14 Summary Results for the Regression Example

The SEVERITY 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

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 23.15. 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 23.15 Comparison of Statistics of Fit for the Regression Example

All Fit Statistics

Distribution	-2 Log Likelihood		AIC		AICC		BIC		KS		AD
Logn	187.49609	*	197.49609	*	198.13439	*	210.52194	*	0.68991	*	0.74299
Burr	190.69154		202.69154		203.59476		218.32256		0.72348		0.73064
Gamma	188.91483		198.91483		199.55313		211.94069		0.69101		0.72219 *

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

All Fit Statistics

Distribution	CvM	
Logn	0.11044	
Burr	0.11332	
Gamma	0.10546	*

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 23.16. 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 23.16 Convergence Results for the Lognormal Model with Regressors

The SEVERITY 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	-1.421E-13	3.2667E-12

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 23.17. 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 23.17 Parameter Estimates for the Lognormal Model with Regressors

Parameter Estimates

Parameter	Estimate	Standard Error	t Value	Approx Pr > |t|
Mu	1.04047	0.07614	13.66	<.0001
Sigma	0.22177	0.01609	13.78	<.0001
x1	0.65221	0.08167	7.99	<.0001
x2	-0.91116	0.07946	-11.47	<.0001
x3	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 23.18. 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 23.18 Parameter Estimates for the Gamma Model with Regressors

Parameter Estimates

Parameter	Estimate	Standard Error	t Value	Approx Pr > |t|
Theta	0.14293	0.02329	6.14	<.0001
Alpha	20.37726	2.93277	6.95	<.0001
x1	0.64562	0.08224	7.85	<.0001
x2	-0.89831	0.07962	-11.28	<.0001
x3	0.14901	0.07870	1.89	0.0613

Syntax: SEVERITY Procedure

The following statements are available in the SEVERITY procedure:

> **PROC SEVERITY** *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* > **;**
> **OUTSCORELIB** < *OUTLIB=* > *fcmp-library-name options* **;**
> **NLOPTIONS** *options* **;**
> **Programming statements ;**

Functional Summary

Table 23.1 summarizes the statements and options that control the SEVERITY procedure.

Table 23.1 SEVERITY 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 library to write scoring functions to	OUTSCORELIB	
Specifies optimization options	NLOPTIONS	
Specifies programming statements that define an objective function	Programming statements	
Input and Output Options		
Specifies that the OUTEST= data set contain covariance estimates	PROC SEVERITY	COVOUT
Specifies the input data set	PROC SEVERITY	DATA=
Specifies the input data set for parameter estimates	PROC SEVERITY	INEST=
Specifies the input item store for parameter initialization	PROC SEVERITY	INSTORE=
Limits the length of effect names	PROC SEVERITY	NAMELEN=
Specifies the output data set for CDF estimates	PROC SEVERITY	OUTCDF=
Specifies the output data set for parameter estimates	PROC SEVERITY	OUTEST=

Table 23.1 *continued*

Description	Statement	Option
Specifies the output data set for model information	PROC SEVERITY	OUTMODELINFO=
Specifies the output data set for statistics of fit	PROC SEVERITY	OUTSTAT=
Specifies the output item store for context and estimation results	PROC SEVERITY	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 SEVERITY	CRITERION=
Specifies the method for computing mixture distribution	SCALEMODEL	DFMIXTURE=
Specifies initial values for model parameters	DIST	INIT=
Specifies the objective function symbol	PROC SEVERITY	OBJECTIVE=
Specifies the offset variable in the scale regression model	SCALEMODEL	OFFSET=
Specifies the denominator for computing covariance estimates	PROC SEVERITY	VARDEF=
Empirical Distribution Function (EDF) Estimation Options		
Specifies the confidence level for reporting the confidence interval for EDF estimates	PROC SEVERITY	EDFALPHA=
Specifies the nonparametric method of CDF estimation	PROC SEVERITY	EMPIRICALCDF=
Specifies the sample to be used for computing the EDF estimates	PROC SEVERITY	INITSAMPLE
EMPIRICALCDF=MODIFIEDKM Options		
Specifies the α value for the lower bound on risk set size	PROC SEVERITY	ALPHA=
Specifies the c value for the lower bound on risk set size	PROC SEVERITY	C=
Specifies the absolute lower bound on risk set size	PROC SEVERITY	RSLB=
EMPIRICALCDF=TURNBULL Options		
Specifies that the final EDF estimates be maximum likelihood estimates	PROC SEVERITY	ENSUREMLE
Specifies the relative convergence criterion	PROC SEVERITY	EPS=
Specifies the maximum number of iterations	PROC SEVERITY	MAXITER=

Table 23.1 *continued*

Description	**Statement**	**Option**
Specifies the threshold below which an EDF estimate is deemed to be 0	PROC SEVERITY	ZEROPROB=
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 SEVERITY	NOCLPRINT
Suppresses all displayed and graphical output	PROC SEVERITY	NOPRINT
Specifies which graphical output to prepare	PROC SEVERITY	PLOTS=
Specifies which output to display	PROC SEVERITY	PRINT=
Specifies that distributions be validated without estimating any models that use them	DIST	VALIDATEONLY

PROC SEVERITY Statement

PROC SEVERITY *options* **;**

The PROC SEVERITY statement invokes the procedure. You can specify two types of *options* in the PROC SEVERITY 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 SEVERITY and various forms of output generated by PROC SEVERITY. 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 1745.

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 SEVERITY 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 1743.

If you specify the SCALEMODEL statement, then PROC SEVERITY 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 SEVERITY 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 SEVERITY uses the uniform random sampling method to select the sample, the size and randomness of which is 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 SEVERITY, which can be useful for replicating the results across different runs. If you do not specify the seed value, PROC SEVERITY 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 PROC SEVERITY computes the EDF estimates by using all valid observations in the DATA= data set, or by using all valid observations in the current BY group if you specify a BY statement.

INSTORE=*store-name* (Experimental)

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 SEVERITY 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 SEVERITY reads the item store from the WORK library. If you specify a two-level name of the form *libname.membername*, then PROC SEVERITY 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 1688 and "Parameter Initialization for Regression Models" on page 1690.

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. 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 1744.

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 1745.

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 SEVERITY 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 SEVERITY 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 1746.

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 1747.

OUTSTORE=*store-name* (Experimental)

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 SEVERITY 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 SEVERITY 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 SEVERITY 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. 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.

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.

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 1687.

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 Craḿer-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 1710.

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. This option is the default.

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 α = 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.

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. 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 SEVERITY 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 SEVERITY 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 SEVERITY 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 1704.

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 SEVERITY 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 1739.

BY Statement

BY *variable-list* **;**

A BY statement can be used in the SEVERITY procedure to process the input data set in groups of observations defined by the BY variables.

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

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 1694.

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 SEVERITY 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 SEVERITY 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 SEVERITY 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 severity;
   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 severity;
   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 1697.

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 SEVERITY procedure. You can specify multiple DIST statements, and each statement can contain one or more distribution specifications.

For your convenience, PROC SEVERITY provides the following 10 different predefined distributions (the name in the 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 1674.

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 SEVERITY 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 1714.

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 SEVERITY 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 a M_PDF function with 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

SEVERITY 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 1689.

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 1688. If none of the initialization methods is used, then PROC SEVERITY 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 SEVERITY 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 SEVERITY'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 SEVERITY 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 SEVERITY 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 SEVERITY 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 SEVERITY 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 SEVERITY 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 1684.

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 SEVERITY 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 1684.

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 SEVERITY 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 1685.

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 SEVERITY 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 SEVERITY 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 SEVERITY 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 1684.

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 SEVERITY 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 1684.

NLOPTIONS Statement

NLOPTIONS *options* ;

The SEVERITY 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 SEVERITY 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.” The SEVERITY 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 SEVERITY statement to request the optimization summary and iteration history. If you specify more than one NLOPTIONS statement, then the first statement is used.

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 SEVERITY.

If you specify the SCALEMODEL statement and if you specify interaction or classification effects, then PROC SEVERITY ignores the OUTSCORELIB statement and does not generate scoring functions. In other words, if you specify the SCALEMODEL statement, then PROC SEVERITY 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 SEVERITY 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 SEVERITY 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 SEVERITY 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 SEVERITY 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 SEVERITY 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 SEVERITY.

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 SEVERITY 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 SEVERITY 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 1731.

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 1697.

All the regressor variables must be present in the input data set that you specify by using the DATA= option in the PROC SEVERITY 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 1689.

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 SEVERITY 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 1692.

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 SEVERITY 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 SEVERITY 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 SEVERITY 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 1690.

WEIGHT Statement

WEIGHT *variable-name* **;**

The WEIGHT statement specifies the name of a variable whose values represent the weight of each observation. PROC SEVERITY 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.

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 SEVERITY 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 SEVERITY statement. If you do not specify the OBJECTIVE= option in the PROC SEVERITY 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 1739.

Details: SEVERITY Procedure

Predefined Distributions

PROC SEVERITY assumes the following model for the response variable Y

$$Y \sim \mathcal{F}(\Theta)$$

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 SEVERITY 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 SEVERITY procedure. A summary of the distributions is provided in Table 23.2. 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 1714.

Table 23.2 Predefined SEVERITY Distributions

Name	Distribution	Parameters	PDF (*f*) and CDF (*F*)
BURR	Burr	$\theta > 0, \alpha > 0, \gamma > 0$	$f(x) = \frac{\alpha\gamma z^{\gamma}}{x(1+z^{\gamma})^{(\alpha+1)}}$ $F(x) = 1 - \left(\frac{1}{1+z^{\gamma}}\right)^{\alpha}$
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	$\theta > 0, \alpha > 0$	$f(x) = \frac{\alpha\theta^{\alpha}}{(x+\theta)^{\alpha+1}}$ $F(x) = 1 - \left(\frac{\theta}{x+\theta}\right)^{\alpha}$
TWEEDIE	Tweedie[6]	$p > 1, \mu > 0, \phi > 0$	$f(x) = a(x,\phi)\exp\left[\frac{1}{\phi}\left(\frac{x\mu^{1-p}}{1-p} - \kappa(\mu,p)\right)\right]$ $F(x) = \int_0^x f(t)dt$
STWEEDIE	Scaled Tweedie[6]	$\theta > 0, \lambda > 0, 1 < p < 2$	$f(x) = a(x,\theta,\lambda,p)\exp\left(-\frac{x}{\theta} - \lambda\right)$ $F(x) = \int_0^x f(t)dt$
WEIBULL	Weibull	$\theta > 0, \tau > 0$	$f(x) = \frac{1}{x}\tau z^{\tau}e^{-z^{\tau}}$ $F(x) = 1 - e^{-z^{\tau}}$

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. $\Phi(y) = \frac{1}{2}\left(1 + \operatorname{erf}\left(\frac{y}{\sqrt{2}}\right)\right)$ is the standard normal CDF.
6. For more information, see the section "Tweedie Distributions" on page 1676.

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

$$\begin{aligned}
\lambda &= \frac{\mu^{2-p}}{\phi(2-p)} \\
\alpha &= \frac{2-p}{p-1} \\
\theta &= \phi(p-1)\mu^{p-1}
\end{aligned}$$

The mean of a Tweedie distribution is positive for $p > 1$.

Two predefined versions of the Tweedie distribution are provided with the SEVERITY 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

$$\begin{aligned}
\mu &= \lambda\theta\alpha \\
\phi &= \frac{(\lambda\theta\alpha)^{2-p}}{\lambda(2-p)} = \frac{\theta}{(p-1)(\lambda\theta\alpha)^{p-1}}
\end{aligned}$$

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 SEVERITY 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 1689. 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 100pth 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 SEVERITY 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 The parameters are initialized by using the method of moments. The kth raw moment of the Burr distribution 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:

$$\hat{\theta} = \sqrt{\frac{m_2 m_3}{2m_3 - 3m_1m_2}}, \quad \hat{\alpha} = 1 + \frac{m_3}{2m_3 - 3m_1m_2}, \quad \hat{\gamma} = 2$$

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 $Var[X] = \mu^3/\lambda$, which yields the second raw moment as $E[X^2] = \mu^2(1+\mu/\lambda)$ (computed by using $E[X^2] = Var[X] + (E[X])^2$).

The predefined IGAUSS distribution in PROC SEVERITY uses the following alternate parameterization to allow the distribution to have a scale parameter, θ:

$$f(x) = \sqrt{\frac{\alpha\theta}{2\pi x^3}} \exp\left(\frac{-\alpha(x-\theta)^2}{2x\theta}\right)$$
$$F(x) = \Phi\left(\left(\frac{x}{\theta}-1\right)\sqrt{\frac{\alpha\theta}{x}}\right) + \Phi\left(-\left(\frac{x}{\theta}+1\right)\sqrt{\frac{\alpha\theta}{x}}\right) \exp(2\alpha)$$

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 kth 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 parameters are initialized by using the method of moments. The kth 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 1676, 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 23.3. 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 23.3 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:
- m_k denotes the kth raw moment
- $d = \log(m_1) - (\sum \log(y_i))/n$
- $q1$ and $q3$ denote the 25th and 75th percentiles, respectively
- $r = \log(\log(4))/\log(\log(4/3))$

Censoring and Truncation

One of the key features of PROC SEVERITY 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 SEVERITY 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 the 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 SEVERITY checks these relationships and write a warning to the SAS log if any is violated.

If you specify the response variable in the LOSS statement, then PROC SEVERITY also checks whether each observation satisfies the definitions of the specified censoring and truncation effects. If you specify left-truncation, then PROC SEVERITY ignores observations where $Y \leq T^l$, because such observations are not observable by definition. Similarly, if you specify right-truncation, then PROC SEVERITY ignores observations where $Y > T^r$. If you specify left-censoring, then PROC SEVERITY 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 SEVERITY 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 SEVERITY 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 1686) and how the empirical distribution function (EDF) is estimated (see the section "Empirical Distribution Function Estimation Methods" on page 1704).

Probability of Observability

For left-truncated data, PROC SEVERITY 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 SEVERITY can use the information about the number of those events.

In particular, for each left-truncated observation, PROC SEVERITY 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 1687) and an unconditional estimate of the empirical distribution function (see the section "EDF Estimates and Truncation" on page 1708).

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. See the section "EDF Estimates and Truncation" on page 1708 for more information. In such cases, PROC SEVERITY uses conditional estimates of the CDF whenever they are used for computational or visual comparison with the EDF estimates.

Let $t^l_{\text{min}} = \min_i\{t^l_i\}$ be the smallest value of the left-truncation threshold (t^l_i is the left-truncation threshold for observation i) and $t^r_{\text{max}} = \max_i\{t^r_i\}$ 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

  $$\hat{F}^c(y) = \frac{\hat{F}(y) - \hat{F}(t^l_{\text{min}})}{\hat{F}(t^r_{\text{max}}) - \hat{F}(t^l_{\text{min}})}$$

  If an observation is left-truncated but not right-truncated, then

  $$\hat{F}^c(y) = \frac{\hat{F}(y) - \hat{F}(t^l_{\text{min}})}{1 - \hat{F}(t^l_{\text{min}})}$$

  If an observation is right-truncated but not left-truncated, then

  $$\hat{F}^c(y) = \frac{\hat{F}(y)}{\hat{F}(t^r_{\text{max}})}$$

- 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

$$\hat{F}^c(y) = \frac{\hat{F}(y)}{\hat{F}(t^r_{\text{max}})}$$

If you specify regression effects, then $\hat{F}(y)$, $\hat{F}(t^l_{\text{min}})$, and $\hat{F}(t^r_{\text{max}})$ are all computed from a mixture distribution, as described in the section "CDF and PDF Estimates with Regression Effects" on page 1692.

Parameter Estimation Method

If you do not specify a custom objective function by specifying programming statements and the OBJECTIVE= option in the PROC SEVERITY statement, then PROC SEVERITY 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 SEVERITY 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 1739.

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

$$l_{C_t} = \Pr(c^r < Y \le c^l | t^l < Y \le t^r) = \frac{F_\Theta(c^l) - F_\Theta(c^r)}{F_\Theta(t^r) - F_\Theta(t^l)}$$

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 SEVERITY 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 1714. 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 SEVERITY 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 $\Pr(Y \le t^l) = F_\Theta(t^l)$. The right-truncation and censoring information does not apply to these added observations. Thus, following the notation of the section "Likelihood Function" on page 1686, 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 SEVERITY 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 SEVERITY 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 $N \to \infty$, 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 $\mathbf{I}(\boldsymbol{\theta}) \cdot \hat{\mathbf{C}} \to 1$, 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 SEVERITY 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 1714.

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 1689.

- 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 SEVERITY 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 1689.

Estimating Regression Effects

The SEVERITY 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 SEVERITY 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 SEVERITY 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 SEVERITY 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 SEVERITY 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 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 1714. If that subroutine is defined, then s_0 is initialized as follows: Each input value y_i of the response variable is transformed to its scale-normalized version w_i as

   $$w_i = \frac{y_i}{\exp(\beta_0 + \sum_{j=1}^{k} \beta_j x_{ij})}$$

   where x_{ij} denotes the value of *j*th regression effect in the *i*th input observation. These w_i values are used to compute the input arguments for the *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 SEVERITY computes and reports the CDF and PDF estimates from a representative distribution. The *representative distribution* is a mixture of a certain number of distributions, where each distribution differs only in the value of the scale parameter. You can specify the number of distributions in the mixture and how their scale values are chosen by using the DFMIXTURE= option in the SCALEMODEL statement.

Let N denote the number of observations that are used for estimation, K denote the number of components in the mixture distribution, s_k denote the scale parameter of the *k*th mixture component, and d_k denote the weight associated with *k*th mixture component.

Let $f(y; s_k, \hat{\Omega})$ and $F(y; s_k, \hat{\Omega})$ denote the PDF and CDF, respectively, of the *k*th component distribution, where $\hat{\Omega}$ denotes the set of estimates of all parameters of the distribution other than the scale parameter. Then, the PDF and CDF estimates, $f^*(y)$ and $F^*(y)$, respectively, of the mixture distribution at *y* are computed as

$$f^*(y) = \frac{1}{D} \sum_{k=1}^{K} d_k f(y; s_k, \hat{\Omega})$$

$$F^*(y) = \frac{1}{D} \sum_{k=1}^{K} d_k F(y; s_k, \hat{\Omega})$$

where *D* is the normalization factor $(D = \sum_{k=1}^{K} d_k)$.

PROC SEVERITY 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 is plots in 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 *j*th regression effect in observation *i*.

Let w_i denote the weight of observation *i*. If you specify the WEIGHT statement, then the weight is equal to the value of the specified weight variable for the corresponding observation in the DATA= data 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 ($k = 1, \dots, K$) is computed as $s_k = \exp(\hat{\lambda}_k)$, where $\hat{\lambda}_k$ is the kth q-quantile from the set $\{\hat{\lambda}_i\}$ ($i = 1, \dots, N$). 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 SEVERITY 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 ($j = 1, \dots, r$), such that $1 \le l_j \le N$, 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 23.4. 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 23.4 apply two different formats to the variable X.

Table 23.4 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 23.4 leads to the level assignment in Table 23.5.

Table 23.5 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 23.6 shows how values of the ORDER= option are interpreted.

Table 23.6 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 23.7 displays the results of levelizing the values in Table 23.4 when the MISSING option is in effect.

Table 23.7 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 SEVERITY 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 SEVERITY 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 SEVERITY 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 1694. 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 1697.

If you specify the CLASS statement, then PROC SEVERITY 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 1700 and "Reference Parameterization" on page 1703.

Effect Operators

Table 23.8 summarizes the operators that are available for selecting and constructing effects. These operators are discussed in the following sections.

Table 23.8 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 23.9 shows the types of effects that are available in the SEVERITY 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 23.9 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 23.10 is an example where β_0 denotes the intercept and A and B are classification variables that have two and three levels, respectively.

Table 23.10 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 23.11.

Table 23.11 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 23.11, 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 SEVERITY 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 23.12.

Table 23.12 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

Table 23.12 *continued*

Data		I	A		B(A)					
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 23.13.

Table 23.13 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 23.14 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 23.13.

Table 23.14 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 SEVERITY 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 23.15.

Table 23.15 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 SEVERITY 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, then PROC SEVERITY computes the EDF estimates by using all valid observations in the DATA= data set, or by using all valid observations in the current BY group if you specify a BY statement.

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, then PROC SEVERITY 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 1706.

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, $\delta_i = I[Y \leq C^r]$ and $y_i = \min(y_i, c_i^r)$.

Given this notation, the EDF is estimated as

$$F_n(y) = \begin{cases} 0 & \text{if } y < y^{(1)} \\ \hat{F}_n(y^{(k)}) & \text{if } y^{(k)} \leq y < y^{(k+1)}, k = 1, \ldots, N-1 \\ \hat{F}_n(y^{(N)}) & \text{if } y^{(N)} \leq y \end{cases}$$

where $y^{(k)}$ denotes the 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 SEVERITY choose an appropriate method by using the EMPIRICALCDF= option in the PROC SEVERITY statement. Each method computes $\hat{F}_n$ as follows:

STANDARD

This method is the standard way of computing EDF. The EDF estimate at observation i is computed as follows:

$$\hat{F}_n(y_i) = \frac{1}{N}\sum_{j=1}^{N} w_j \cdot I[y_j \leq y_i]$$

If you do not specify any censoring or truncation information, then this method is chosen. If you explicitly specify this method, then PROC SEVERITY 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 SEVERITY uses the definition that combines both censoring and truncation information (Klein and Moeschberger 1997; Lai and Ying 1991).

The EDF estimate at observation i is computed as

$$\hat{F}_n(y_i) = 1 - \prod_{\tau \leq y_i} \left(1 - \frac{n(\tau)}{R_n(\tau)}\right)$$

where $n(\tau)$ and $R_n(\tau)$ are defined as follows:

- $n(\tau) = \sum_{k=1}^{N} w_k \cdot I[y_k = \tau \text{ and } \tau \leq t_k^r \text{ and } \delta_k = 1]$, which is the number of uncensored observations ($\delta_k = 1$) for which the response variable value is equal to τ and τ is observable according to the right-truncation threshold of that observation ($\tau \leq t_k^r$).

- $R_n(\tau) = \sum_{k=1}^{N} w_k \cdot I[y_k \geq \tau > t_k^l]$, which is the size (cardinality) of the risk set at 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 SEVERITY, *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 KAPLAN-MEIER 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, then the SEVERITY 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 1704. 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 SEVERITY 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 SEVERITY 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 SEVERITY 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 SEVERITY 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 SEVERITY statement, then PROC SEVERITY 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 SEVERITY 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 SEVERITY 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 SEVERITY 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 $\hat{\mathrm{C}}(\boldsymbol{s}) = \mathbf{G}^{-1}(\boldsymbol{s})$. 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 SEVERITY estimates τ_G and τ_H by t^l_{min} and t^r_{max}, respectively, defined as

$$t^l_{\text{min}} = \min\{t^l_k : 1 \le k \le N\}$$
$$t^r_{\text{max}} = \max\{t^r_k : 1 \le k \le N\}$$

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 1685.

If you specify left-truncation *with* the probability of observability p, then PROC SEVERITY 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, 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 1714 and "Predefined Utility Functions" on page 1726.

PROC SEVERITY 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 SEVERITY 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. The following subsections provide definitions of each.

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 1686. 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 $r_i = \sum_{j=1}^{N} I[y_j \leq y_i]$ denote the number of observations with a value less than or equal to y_i, where I is an indicator function. 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. See the section "CDF and PDF Estimates with Regression Effects" on page 1692 for more information.
- 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 1685.

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:

$$D^+ = \max_i \left(\frac{r_i}{N} - Z_i\right)$$
$$D^- = \max_i \left(Z_i - \frac{r_{i-1}}{N}\right)$$
$$\text{KS} = \sqrt{N}\,\max(D^+, D^-) + \frac{0.19}{\sqrt{N}}$$

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:

$$D^+ = \max_i (F_n(Z_i) - Z_i), \text{ if } F_n(Z_i) \geq Z_i$$
$$D^- = \max_i (Z_i - F_n(Z_i)), \text{ if } F_n(Z_i) < Z_i$$
$$\text{KS} = \sqrt{N}\,\max(D^+, D^-) + \frac{0.19}{\sqrt{N}}$$

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:

$$\text{AD} = -N - \frac{1}{N} \sum_{i=1}^{N} [(2r_i - 1)\log(Z_i) + (2N + 1 - 2r_i)\log(1 - Z_i)]$$

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 following 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 following 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:

$$\text{CvM} = \frac{1}{12N} + \sum_{i=1}^{N} \left(Z_i - \frac{(2r_i - 1)}{2N} \right)^2$$

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).

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 23.16.

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 SEVERITY 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 provided by PROC SEVERITY 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 SEVERITY 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 SEVERITY 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 SEVERITY, 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 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 1674 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 23.2. You do not need to use the CMPLIB= option in order to use the predefined distributions with PROC SEVERITY. However, if you need to use the functions or subroutines of the predefined distributions in SAS statements other than the PROC SEVERITY 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 23.16 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 23.16 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 SEVERITY. 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 SEVERITY. You can estimate them inside the *dist*_PARMINIT subroutine. Once initialized, the parameters remain constant in the context of PROC SEVERITY; that is, they retain their initial value. PROC SEVERITY estimates only the nonconstant parameters.

- *Type*: Subroutine
- *Required*: NO
- *Number of arguments*: k, where k is the number of constant parameters
- *Sequence and type of arguments*:

constant parameter 1	Name of the first constant parameter
...	
constant parameter k	Name of the kth constant parameter

- *Return value*: None

Here is a sample structure of the subroutine for a distribution named 'FOO' that has P3 and P5 as its constant parameters, assuming that distribution has at least three parameters:

```
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 SEVERITY 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 SEVERITY procedure assumes that $l_i < p_i$ (strict inequality). If a missing value is returned for some parameter, then the SEVERITY 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 SEVERITY 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. See the section "Supplying EDF Estimates to Functions and Subroutines" on page 1709 for definition of types.
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
	...
pm	Numeric value of the mth 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 SEVERITY 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 SEVERITY procedure assumes that $p_i < u_i$ (strict inequality). If a missing value is returned for some parameter, then the SEVERITY 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
	...
pm	Numeric value of the mth 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
	. . .
pm	Numeric value of the mth 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 SEVERITY 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. See the section "Supplying EDF Estimates to Functions and Subroutines" on page 1709 for definition of types.

- *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 1709.

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. See the section "Supplying EDF Estimates to Functions and Subroutines" on page 1709 for definition of types.

- *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 1709 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 ($m_1 < n$).

$x_{(i)}^{j,m}$	*i*th order statistic of *j*th bootstrap sample of size m $(1 \le i \le m, 1 \le j \le B)$.
$x_{(i)}$	*i*th order statistic of the original sample $(1 \le i \le n)$.

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:

$$k_1 = \arg \min_{1 \le k < m_1} Q(m_1, k)$$

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:

$$k_2 = \arg \min_{1 \le k < m_2} Q(m_2, k)$$

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 *j*th 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,

$$M_j(m,k) = \frac{1}{k} \sum_{i=1}^{k} \left(\log(x_{(m-i+1)}^{j,m}) - \log(x_{(m-k)}^{j,m}) \right)^2$$

and $\gamma_j(m,k)$ is the Hill's estimator of the tail index (Hill 1975),

$$\gamma_j(m,k) = \frac{1}{k} \sum_{i=1}^{k} \log(x_{(m-i+1)}^{j,m}) - \log(x_{(m-k)}^{j,m})$$

This algorithm has two tuning parameters, B and m_1. The number of bootstrap samples B is chosen based on the availability of computational resources. The optimal value of m_1 is chosen such that the following ratio, $R(m_1)$, is minimized:

$$R(m_1) = \frac{(Q(m_1, k_1))^2}{Q(m_2, k_2)}$$

The SVRTUTIL_HILLCUTOFF utility function implements the preceding algorithm. It uses the grid search method to compute the optimal value of m_1.

- *Type*: Function
- *Signature*: SVRTUTIL_HILLCUTOFF(n, x{*}, b, s, status)

- *Argument Description*:

n	Dimension of the array *x*.
x{*}	Input numeric array of dimension *n* that contains the sample.
b	Number of bootstrap samples used to estimate the mean squared error. If *b* is less than 10, then a default value of 50 is used.
s	Approximate number of steps used to search the optimal value of m_1 in the range $[n^{0.75}, n-1]$. If *s* is less than or equal to 1, then a default value of 10 is used.
status	Output argument that contains the status of the algorithm. If the algorithm succeeds in computing a valid cutoff point, then *status* is set to 0. If the algorithm fails, then *status* is set to 1.

- *Return value*: The cutoff value where the right tail is estimated to start. If the size of the input sample is inadequate ($n \leq 5$), then a missing value is returned and *status* is set to a missing value. If the algorithm fails to estimate a valid cutoff value (*status* = 1), then the fifth largest value in the input sample is returned.

SVRTUTIL_PERCENTILE:

This function computes the specified empirical percentile given the EDF estimates.

- *Type*: Function
- *Signature*: SVRTUTIL_PERCENTILE(p, n, x{*}, F{*}, Ftype)
- *Argument Description*:

p	Desired percentile. The value must be in the interval (0,1). The function returns the 100*p*th percentile.
n	Dimension of the *x* and *F* input arrays.
x{*}	Input numeric array of dimension *n* that contains values of the random variable observed in the sample. These values are sorted in nondecreasing order.
F{*}	Input numeric array of dimension *n* in which each F[*i*] contains the EDF estimate for x[*i*]. These values must be sorted in nondecreasing order.
Ftype	Type of the empirical estimate that is stored in the *x* and *F* arrays. See the section "Supplying EDF Estimates to Functions and Subroutines" on page 1709 for definition of types.

- *Return value*: The 100*p*th percentile of the input sample.

The method used to compute the percentile depends on the type of the EDF estimate (Ftype argument).

Ftype = 1 Smoothed empirical estimates are computed using the method described in Klugman, Panjer, and Willmot (1998). Let $\lfloor x \rfloor$ 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

$$\begin{aligned} E[(X \wedge u)^k] &= \int_0^u x^k f(x)dx + \int_u^\infty u^k f(x)dx \\ &= \int_0^u x^k f(x)dx + u^k(1 - F(u)) \end{aligned}$$

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 SEVERITY. However, that approach requires some cumbersome programming. In order to simplify the scoring process, you can specify that PROC SEVERITY create scoring functions for each fitted distribution.

As an example, assume that you have fitted the Pareto distribution by using PROC SEVERITY 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 SEVERITY step that fits your distributions, you create the scoring functions library by specifying the OUTSCORELIB statement as illustrated in the following steps:

```
proc severity 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 SEVERITY 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 SEVERITY. 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 severity 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 SEVERITY 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 1714.

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 SEVERITY 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 SEVERITY 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 SEVERITY step to create the scoring library Sasuser.Scalescore:

```
proc severity 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 SEVERITY 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 SEVERITY 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 SEVERITY 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 SEVERITY 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 SEVERITY 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 1689.

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 SEVERITY 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 SEVERITY 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 SEVERITY 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 SEVERITY 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 SEVERITY 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 SEVERITY 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 SEVERITY computes the estimate using the SVRTUTIL_EDF function as described in the section "Predefined Utility Functions" on page 1726. 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 SEVERITY computes the moment using the SVRTUTIL_EMPLIMMOMENT function as described in the section "Predefined Utility Functions" on page 1726. 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 SEVERITY computes the moment using the LIMMOMENT function as described in the section "Predefined Utility Functions" on page 1726.

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 severity 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 SEVERITY 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 SEVERITY 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 SEVERITY 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 SEVERITY. 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 SEVERITY 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. See the section "PROC FCMP and DATA Step Differences" in *Base SAS Procedures Guide* for more information. In addition to the differences listed in that section, the following differences exist:
  - Only numeric-valued variables can be used in PROC SEVERITY 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 SEVERITY programming statements. If you need lagged versions, then you can use a DATA step prior to the PROC SEVERITY 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 SEVERITY.

Custom Objective Functions and Regression Effects

If you specify regression effects by using the SCALEMODEL statement, then PROC SEVERITY 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 SEVERITY prepends your program with a statement that is equivalent to the following statement:

```
S = _SEVTHETA0 * exp(_SEVBETA1 * x1 + _SEVBETA2 * x2 + _SEVBETA3 * x3);
```

If a model contains a distribution D1 with a log-transformed scale parameter M, PROC SEVERITY prepends your program with a statement that is equivalent to the following statement:

```
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 SEVERITY writes an error message to the SAS log and stops processing.

Multithreaded Computation

PROC SEVERITY attempts to use all the computational resources of the machine where SAS is running in order to complete the estimation tasks as fast as possible. This section describes the options that control the use of multithreading by PROC SEVERITY.

Threading refers to the organization of computational work into multiple tasks (processing units that can be scheduled by the operating system). A task is associated with a thread. Multithreading refers to the concurrent execution of threads. When multithreading is possible, substantial performance gains can be realized compared to sequential (single-threaded) execution.

The number of threads spawned by the SEVERITY procedure is determined by the number of CPUs on a machine. You can control the number of threads by specifying either the CPUCOUNT= or the NOTHREADS SAS system option.

- You can specify the CPU count with the CPUCOUNT= SAS system option. For example, if you specify the following statement, then PROC SEVERITY schedules threads as if it executed on a system with four CPUs, regardless of the actual CPU count:

  ```
  options cpucount=4;
  ```

  On most systems, the default value of the CPUCOUNT= system option is set to the number of actual CPU cores available for processing.

- If you do not want PROC SEVERITY to use multithreading, then you can turn off the THREADS SAS system option by specifying the following statement:

  ```
  options nothreads;
  ```

  On most systems, the THREADS option is turned on by default.

You can examine the current settings of these system options in the SAS log by submitting the following PROC OPTIONS step:

```
proc options option=(threads cpucount);
run;
```

Input Data Sets

PROC SEVERITY 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 SCALE-MODEL 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 1689.

<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 SEVERITY writes the OUTCDF=, OUTEST=, OUTMODELINFO=, and OUTSTAT= data sets when requested by their respective options. The data sets and their contents are described in the following sections.

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).

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 SEVERITY 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 SEVERITY 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 SEVERITY statement (default is $\alpha = 0.05$). For an EDF estimate F_n that has standard error σ_n, it is computed as 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 SEVERITY 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 regression effects, then the reported estimates are from a mixture distribution. For more information, see the section "CDF and PDF Estimates with Regression Effects" on page 1692.

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 1685.

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 1748.

<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 1689.

<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 SEVERITY 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 SEVERITY 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 1748.
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 SEVERITY procedure optionally produces displayed output by using the Output Delivery System (ODS). All output is controlled by the PRINT= option in the PROC SEVERITY statement. Table 23.17 relates the ODS tables to PRINT= options.

Table 23.17 ODS Tables Produced in PROC SEVERITY

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

Table 23.17 *continued*

ODS Table Name	Description	Option
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
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	PRINT=ALL
TurnbullSummary	Turnbull EDF estimation summary	PRINT=ALL

If you do not specify the PRINT= option, then by default PROC SEVERITY produces ModelSelection, ConvergenceStatus, OptimizationSummary, StatisticsOfFit, and ParameterEstimates ODS tables.

The following describes the content that is displayed in each table:

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.

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.

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 (PRINT=ALL)
displays elapsed times (absolute and relative) for the main tasks of the procedure.

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, 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 SEVERITY procedure.

ODS Graph Names

PROC SEVERITY 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 23.18.

Table 23.18 ODS Graphics Produced by PROC SEVERITY

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 1685.

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 1692.

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 MAX$(0, F_n - z_{(1-\alpha/2)}\sigma_n)$ and 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 1685.

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 1692.

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 1692.

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 1692.

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 1711, 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 1708. 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 1685.

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 1692.

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 1722, 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 SEVERITY 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 1708. 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

$$F_n^u = F_n \cdot (\hat{F}(t_{\text{max}}^r) - \hat{F}(t_{\text{min}}^l)) + \hat{F}(t_{\text{min}}^l)$$

where $\hat{F}(t_{\text{max}}^r)$ and $\hat{F}(t_{\text{min}}^l)$ are as defined in the section "Truncation and Conditional CDF Estimates" on page 1685.

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 1692.

Examples: SEVERITY Procedure

Example 23.1: 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):

$$f(x; \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$
$$F(x; \mu, \sigma) = \frac{1}{2}\left(1 + \text{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right)$$

For PROC SEVERITY, 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 1714.

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 SEVERITY 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 1726.

```
/*-------- 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 1714. 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 SEVERITY 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 SEVERITY 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 SEVERITY. 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 SEVERITY, 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 SEVERITY 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 SEVERITY. The following statements fit the model to the values of Y in the Work.Testnorm data set:

```
/*--- Fit models with PROC SEVERITY ---*/
proc severity 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 SEVERITY statement, and the INIT= option is not specified in the DIST statement. So PROC SEVERITY initializes the parameters by invoking the NORMAL_PARMINIT subroutine.

Some of the results prepared by the preceding PROC SEVERITY step are shown in Output 23.1.1 and Output 23.1.2. The descriptive statistics of variable Y and the “Model Selection” table, which includes just the normal distribution, are shown in Output 23.1.1.

Output 23.1.1 Summary of Results for Fitting the Normal Distribution

The SEVERITY 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

Output 23.1.1 *continued*

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 23.1.2. 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 23.1.2 Details of the Fitted Normal Distribution Model

The SEVERITY 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	Estimate	Standard Error	t Value	Approx Pr > |t|
Mu	10.02059	0.23894	41.94	<.0001
Sigma	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 23.2: 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 SEVERITY 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 SEVERITY, so that it can identify which is the scale parameter. When you specify regression effects, PROC SEVERITY 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:

$$\sigma = \exp(1 + 0.5\ \text{X1} + 0.75\ \text{X3} - 2\ \text{X4} + \text{X5})$$

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 and x3 */
      x(2) = x(1) + 5 * x(3);

      /* 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 SEVERITY 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 SEVERITY --------*/
proc severity data=testnorm_reg print=all plots=none;
   loss y;
   scalemodel x1-x5;
   dist Normal_s burr logn pareto weibull;
run;
```

The "Model Selection" table in Output 23.2.1 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 23.2.1 Summary of Results for Fitting the Normal Distribution with Regressors

The SEVERITY 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

All Fit Statistics

Distribution	-2 Log Likelihood	AIC	AICC	BIC	KS	AD
Normal_s	603.95786 *	615.95786 *	616.86108 *	631.58888 *	1.52388	4.00152
Burr	612.81685	626.81685	628.03424	645.05304	1.50448 *	3.90072 *
Logn	749.20125	761.20125	762.10448	776.83227	2.88110	16.20558
Pareto	841.07022	853.07022	853.97345	868.70124	4.83810	31.60568
Weibull	612.77496	624.77496	625.67819	640.40598	1.50490	3.90559

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

All Fit Statistics

Distribution	CvM
Normal_s	0.70769
Burr	0.63399 *
Logn	3.04825
Pareto	6.84046
Weibull	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 23.2.2. 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 SEVERITY 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 23.2.2 Details of the Fitted Burr Distribution Model

The SEVERITY Procedure
Burr Distribution

Distribution Information	
Name	Burr
Description	Burr Distribution
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 SEVERITY 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 severity data=testnorm_reg print=all plots=pp;
   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 23.2.3 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 23.2.3 Summary of Results after Changing Maximum Number of Iterations

The SEVERITY 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
Normal_s	603.95786 *	615.95786 *	616.86108 *	631.58888 *	1.52388	4.00152
Burr	612.79276	626.79276	628.01015	645.02895	1.50472 *	3.90351 *
Weibull	612.77496	624.77496	625.67819	640.40598	1.50490	3.90559

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

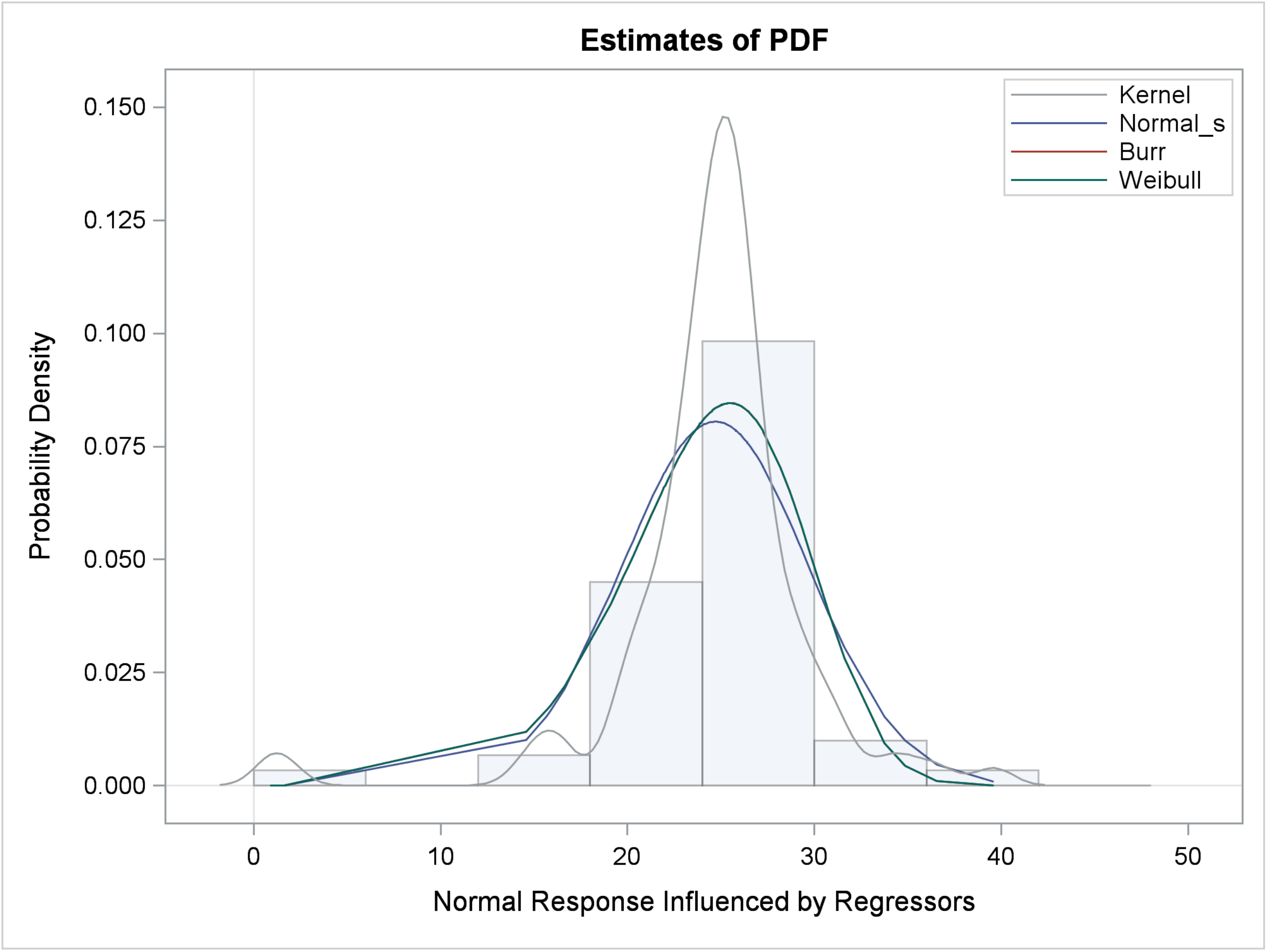
All Fit Statistics

Distribution	CvM
Normal_s	0.70769
Burr	0.63433 *
Weibull	0.63458

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

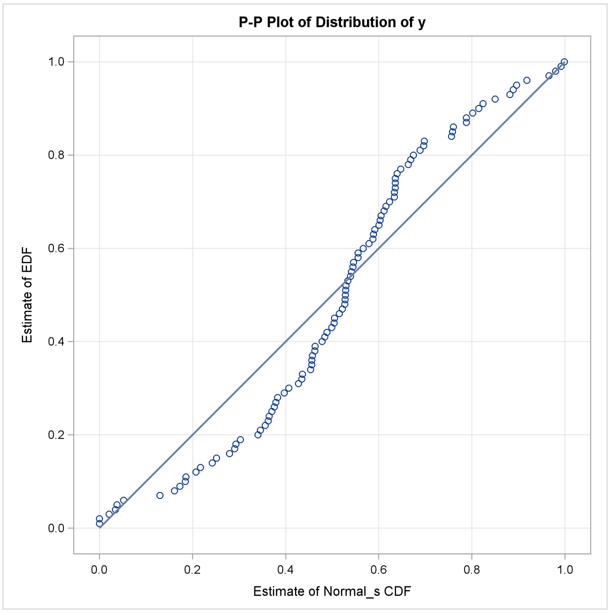
The comparison of the PDF estimates of all the candidates is shown in Output 23.2.4. Each plotted PDF estimate is an average computed over the *N* PDF estimates that are obtained with the scale parameter determined by each of the *N* observations in the input data set. The PDF plot shows that the Burr and Weibull models result in almost identical estimates. All the estimates have a slight left skew with the mode closer to Y=25, which is the mean of the simulated sample.

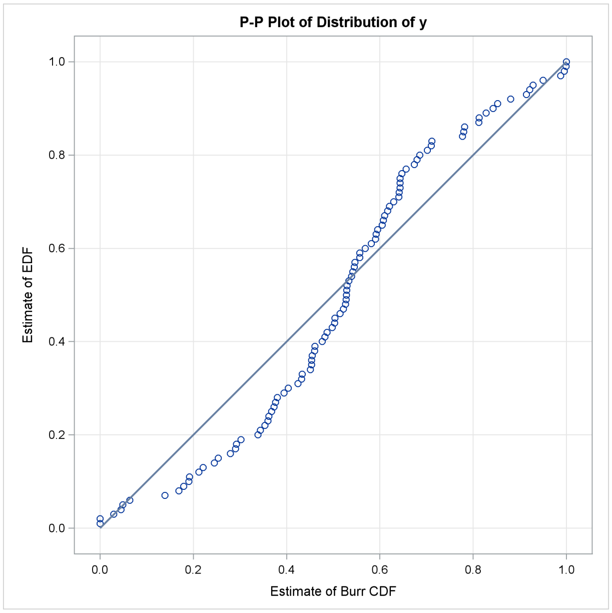
Output 23.2.4 Comparison of EDF and CDF Estimates of the Fitted Models



The P-P plots for the Normal_s and Burr distributions are shown in Output 23.2.5. These plots show how the EDF estimates compare against the CDF estimates. Each plotted CDF estimate is an average computed over the *N* CDF estimates that are obtained with the scale parameter determined by each of the *N* observations in the input data set. Comparing the P-P plots of Normal_s and Burr distributions indicates that both fit the data almost similarly, but the Burr distribution fits the right tail slightly better, which explains why the EDF-based statistics prefer it over the Normal_s distribution.

Output 23.2.5 Comparison of EDF and CDF Estimates of NORMAL_S and BURR Models





Example 23.3: 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 the 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 1726. 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);

   /* prepare arrays for excess values */
   i = 1;
   do while (i <= dim and x[i] < Xb+eps);
      i = i + 1;
   end;
   dime = dim-i+1;
   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 lognormal parameters */
         call logn_parminit(dim, x, nx, F, Ftype, Mu, Sigma);
         if (not(missing(Mu))) then
            Xr = Xb/exp(Mu);
         else
            Xr = .;

         /* Initialize GPD's shape parameter using excess values */
         call gpd_parminit(dime, xe, nxe, F, Ftype, theta_gpd, Xi);
      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 SEVERITY, so you need to specify them as *constant* parameters by defining the *dist*_CONSTANTPARM subroutine. The signature of 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 SEVERITY interprets as having no lower bound. You need not 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 SEVERITY 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 SEVERITY --------*/
proc severity data=testmixdist print=all plots(histogram kernel)=all
           outest=parmest;
   loss y;
   dist logngpd burr logn gpd;
run;
```

Some of the results prepared by PROC SEVERITY are shown in Output 23.3.1 through Output 23.3.4. The "Model Selection" table in Output 23.3.1 indicates that all models converged. The last table in Output 23.3.1 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 23.3.1 Summary of Fitting Mixed-Tail Distribution

The SEVERITY 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
logngpd	418.78232 *	428.78232 *	429.42062 *	441.80817	0.62140 *	0.31670 *
Burr	424.93728	430.93728	431.18728	438.75280 *	0.71373	0.57649
Logn	459.43471	463.43471	463.55842	468.64505	1.55267	3.27122
Gpd	558.13444	562.13444	562.25815	567.34478	3.43470	16.74156

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

All Fit Statistics

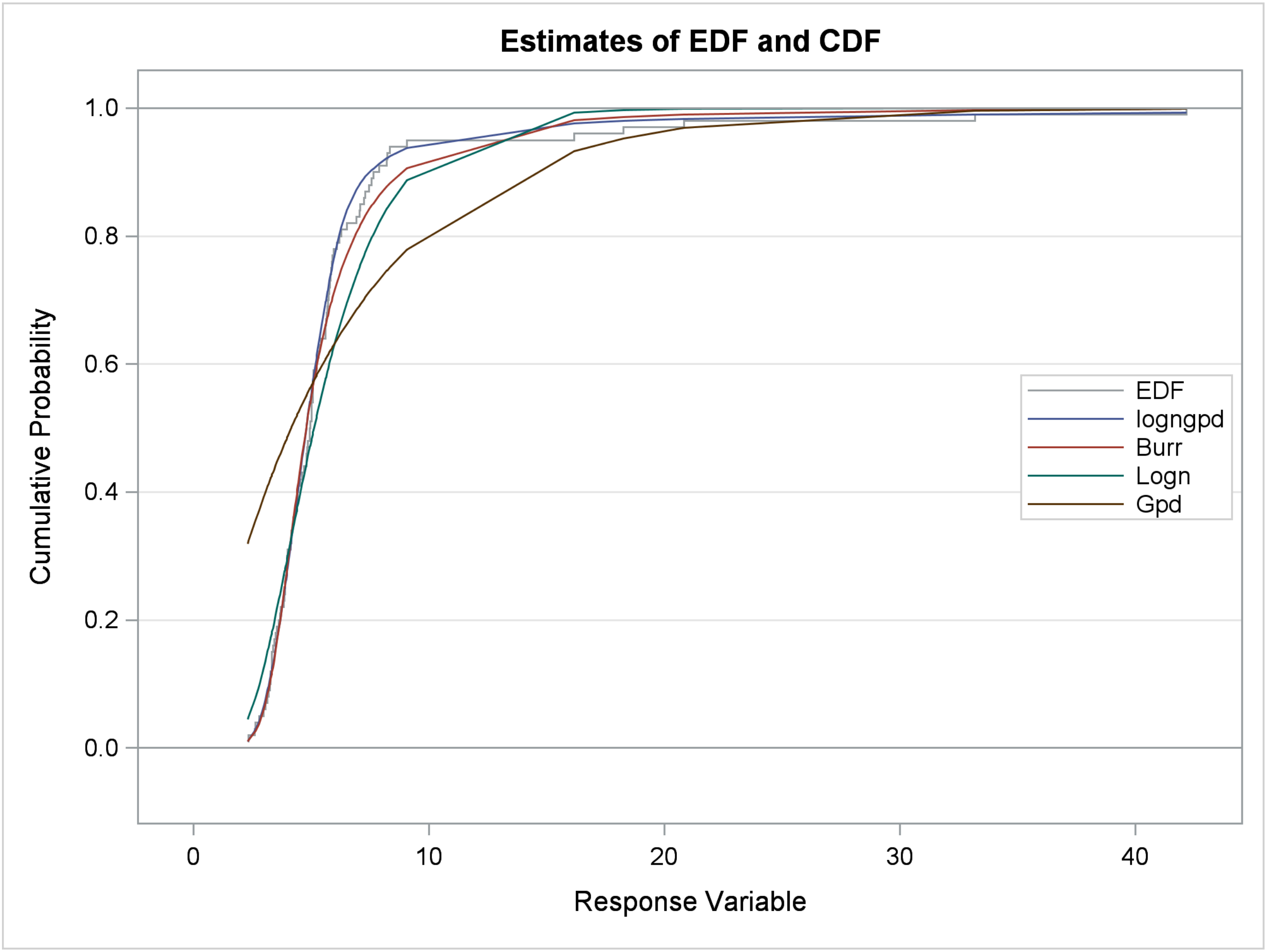
Distribution	CvM
logngpd	0.04972 *
Burr	0.07860
Logn	0.48448
Gpd	3.31860

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

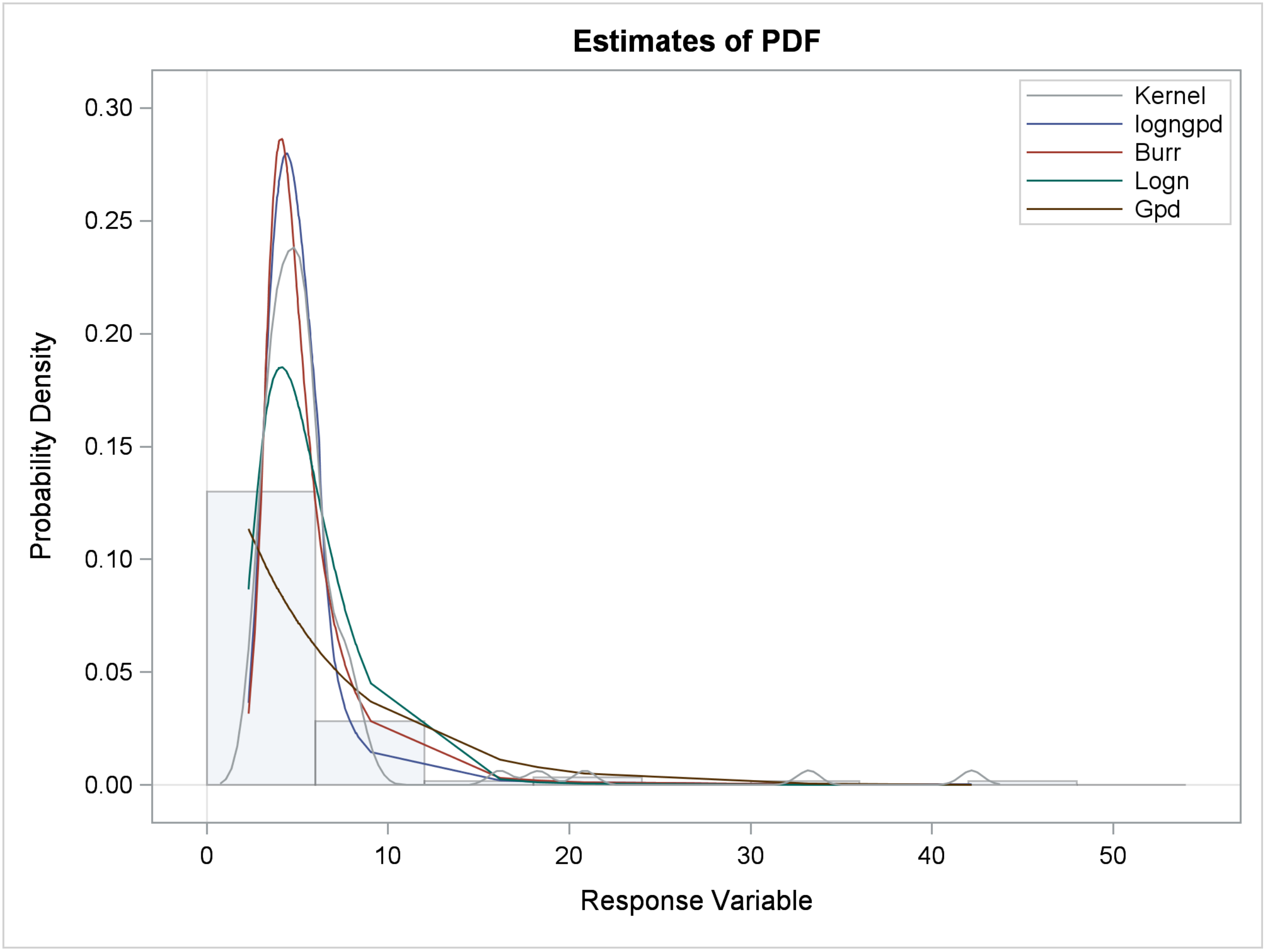
The plots in Output 23.3.2 show that both the lognormal and GPD distributions fit the data poorly, GPD being the worst. The Burr distribution fits the data as well as the LOGNGPD distribution in the body region, but has a poorer fit in the tail region than the LOGNGPD distribution.

The P-P plots of Output 23.3.3 provide a better visual confirmation that the LOGNGPD distribution fits the tail region better than the Burr distribution.

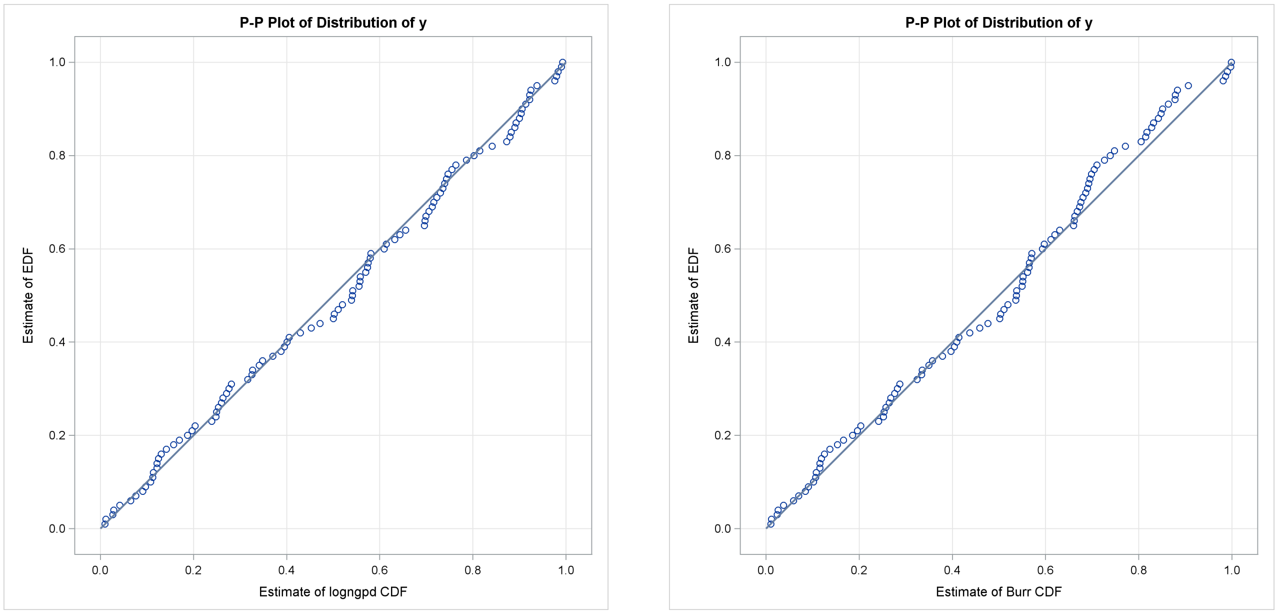
Output 23.3.2 Comparison of the CDF and PDF Estimates of the Fitted Models



Output 23.3.2 *continued*



Output 23.3.3 P-P Plots for the LOGNGPD and BURR Distribution Models



The detailed results for the LOGNGPD distribution are shown in Output 23.3.4. 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 23.3.4 Detailed Results for the LOGNGPD Distribution

The SEVERITY 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	Estimate	Standard Error	t Value	Approx Pr > |t|
Mu	1.57921	0.06426	24.57	<.0001
Sigma	0.31868	0.04459	7.15	<.0001
Xi	1.03771	0.38205	2.72	0.0078
Xr	1.27395	Constant	.	.
Pn	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 23.3.5 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 23.3.5. 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 gets optimized by virtue of optimizing the scale of the body region ($\theta = e^{\mu}$).
- 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 used for simulating the sample.
- The values $y_t = y - 6.18$ ($y_t > 0$) follow the GPD distribution with scale $\theta_t \approx 1.28$ and shape $\xi \approx 1.04$.

Example 23.4: Estimating Parameters Using Cramér-von Mises Estimator

PROC SEVERITY enables you to estimate model parameters by minimizing your own objective function. This example illustrates how you can use PROC SEVERITY 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 SEVERITY 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 23.3: Defining a Model for Mixed-Tail Distributions" on page 1764. 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 SEVERITY by using -------
  -------- the Cramer-von Mises minimum distance estimator -------*/
proc severity data=testmixdist obj=cvmobj print=all plots=pp;
   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 SEVERITY 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 SEVERITY are shown in Output 23.4.1. 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 23.4.1 with those of Output 23.3.1 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 23.4.1 Summary of Cramér-von Mises Estimation

The SEVERITY Procedure

Input Data Set	
Name	WORK.TESTMIXDIST
Label	Lognormal Body-GPD Tail Sample

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

Output 23.4.1 *continued*

All Fit Statistics

Distribution	cvmobj		-2 Log Likelihood		AIC		AICC		BIC		KS	
logngpd	0.02694	*	419.49635	*	429.49635	*	430.13464	*	442.52220	*	0.51332	*
Burr	0.03325		436.58823		442.58823		442.83823		450.40374		0.53084	
Logn	0.03633		491.88659		495.88659		496.01030		501.09693		0.52469	
Gpd	2.96090		560.35409		564.35409		564.47780		569.56443		2.99095	

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

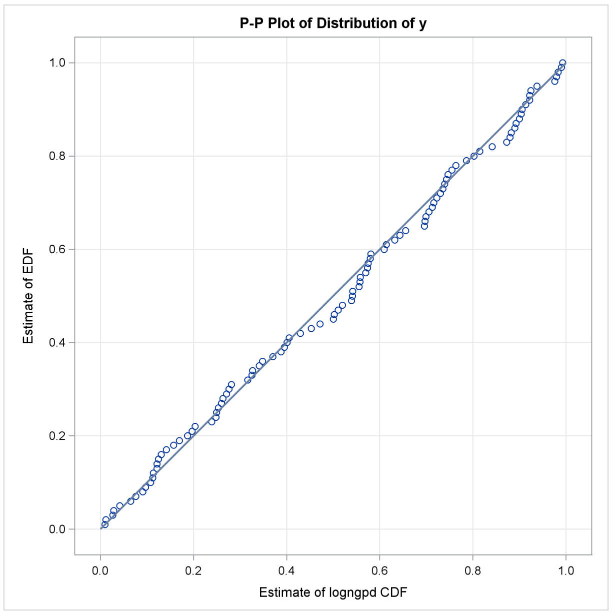
All Fit Statistics

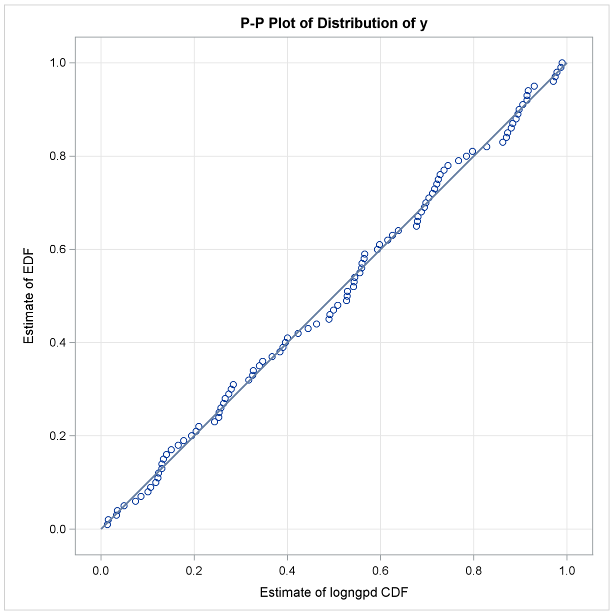
Distribution	AD		CvM	
logngpd	0.21563	*	0.03030	*
Burr	0.82875		0.03807	
Logn	2.08312		0.04173	
Gpd	15.51378		2.97806	

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

The P-P plots in Output 23.4.2 provide a visual confirmation that the CDF estimates match the EDF estimates more closely when compared to the estimates that are obtained with the maximum likelihood estimator.

Output 23.4.2 P-P Plots for LOGNGPD Model with Maximum Likelihood (Left) and Cramér-von Mises (Right) Estimators





Example 23.5: 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 SEVERITY 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 SEVERITY 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 1676, 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:

$$\mu = 5\ \exp(0.25\ \text{x1} - \text{x2} + 3\ \text{x3})$$

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 SEVERITY sample program *sevex05.sas*.

The following PROC SEVERITY 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 severity data=test_sevtw outest=estw covout print=all plots=none;
   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 SEVERITY are shown in Output 23.5.1 and Output 23.5.2. The distribution information and the convergence results are shown in Output 23.5.1.

Output 23.5.1 Convergence Results for the STWEEDIE Model with Regressors

The SEVERITY Procedure
stweedie Distribution

Distribution Information	
Name	stweedie
Description	Tweedie Distribution with Scale Parameter
Distribution Parameters	3
Regression Parameters	3

Convergence Status
Convergence criterion (FCONV=2.220446E-16) satisfied.

Optimization Summary	
Optimization Technique	Dual Quasi-Newton
Iterations	42
Function Calls	243
Log Likelihood	-1044.3

The final parameter estimates of the STWEEDIE regression model are shown in Output 23.5.2. 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 23.5.2 Parameter Estimates for the STWEEDIE Model with Regressors

Parameter Estimates

Parameter	Estimate	Standard Error	t Value	Approx Pr > |t|
Theta	0.82345	0.29281	2.81	0.0053
Lambda	16.24203	14.02459	1.16	0.2478
P	1.74930	0.22497	7.78	<.0001
x1	0.27930	0.09879	2.83	0.0050
x2	-0.76730	0.10318	-7.44	<.0001
x3	3.03183	0.10143	29.89	<.0001

If your goal is to explain the influence of regression effects on the scale parameter, then the output displayed in Output 23.5.2 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 SEVERITY step. The DATA step program is available in the sample program *sevex05.sas*. The estimates of μ_0 prepared by that program are shown in Output 23.5.3. These estimates and the estimates of β_j as shown in Output 23.5.2 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 23.5.3 Estimate of the Base Value Mu0 of the Mean Parameter

Parameter	Estimate	Standard Error	t Value	Approx Pr > |t|
Mu0	4.47472	0.42271	10.5858	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 1676, 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 SEVERITY enables you to estimate regression effects on various parameters and statistics of the Tweedie model.

Example 23.6: 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 SEVERITY 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):

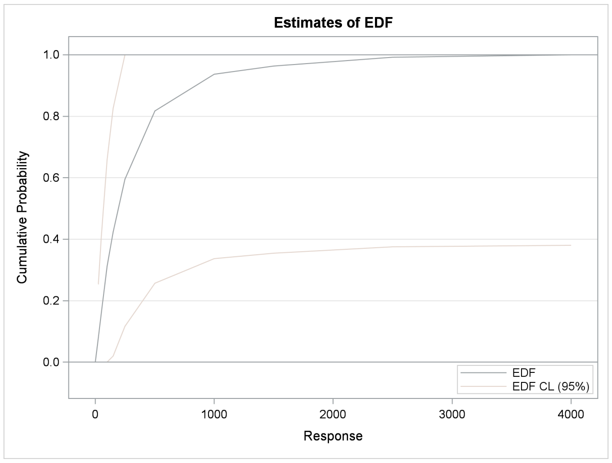
```
/* Grouped dental insurance claims data
     (Klugman, Panjer, and Willmot, 1998) */
data gdental;
   input lowerbd upperbd count @@;
   datalines;
0 25 30  25 50 31  50 100 57  100 150 42  150 250 65  250 500 84
500 1000 45  1000 1500 10  1500 2500 11  2500 4000 3
;
run;
```

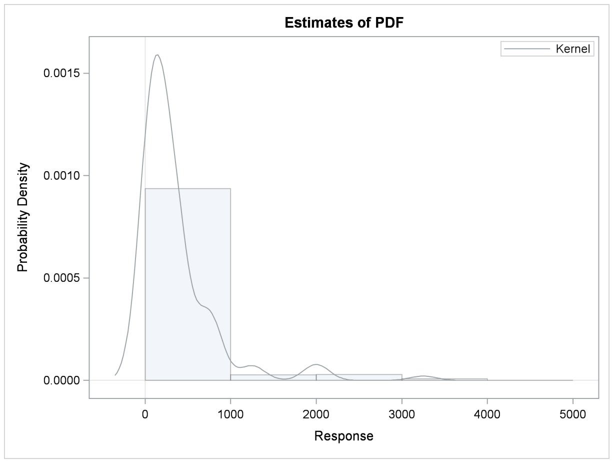
Often, when you do not know the nature of the data, it is recommended that you first explore the nature of the sample distribution by examining the nonparametric estimates of PDF and CDF. The following PROC SEVERITY step prepares the nonparametric estimates, but it does not fit any distribution because there is no DIST statement specified:

```
/* Prepare nonparametric estimates */
proc severity data=gdental print=all plots(histogram kernel)=all;
   loss / rc=lowerbd lc=upperbd;
   weight count;
run;
```

The LOSS statement specifies the left and right boundary 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 there is no response or loss variable specified in the LOSS statement, which is allowed as long as each observation in the input data set is censored. The nonparametric estimates prepared by this step are shown in Output 23.6.1. The histogram, kernel density, and EDF plots all indicate that the data is heavy-tailed. For interval-censored data, PROC SEVERITY uses Turnbull's algorithm to compute the EDF estimates. The plot of Turnbull's EDF estimates is shown to be linear between the endpoints of a censored group. The linear relationship is chosen for convenient visualization and ease of computation of EDF-based statistics, but you should note that theoretically the behavior of Turnbull's EDF estimates is undefined within a group.

Output 23.6.1 Nonparametric Distribution Estimates for Interval-Censored Data





With the PRINT=ALL option, PROC SEVERITY prints the summary of the Turnbull EDF estimation process as shown in Output 23.6.2. It indicates that the final EDF estimates have converged and are in fact maximum likelihood (ML) estimates. If they were not ML estimates, then you could have used the ENSUREMLE option to force the algorithm to search for ML estimates.

Output 23.6.2 Turnbull EDF Estimation Summary for Interval-Censored Data

Turnbull EDF Estimation Summary	
Technique	EM with Maximum Likelihood Check
Convergence Status	Converged
Iterations	2
Maximum Absolute Relative Error	1.8406E-16
Maximum Absolute Reduced Gradient	1.7764E-15
Estimates	Maximum Likelihood

After exploring the nature of the data, you can now fit a set of heavy-tailed distributions to this data. The following PROC SEVERITY step fits all the predefined distributions to the data in Work.Gdental data set:

```
/* Fit all predefined distributions */
proc severity data=gdental print=all plots(histogram kernel)=all
              criterion=ad;
   loss / rc=lowerbd lc=upperbd;
   weight count;
   dist _predef_;
run;
```

Some of the key results prepared by PROC SEVERITY are shown in Output 23.6.3 through Output 23.6.4. According to the “Model Selection” table in Output 23.6.3, all distribution models have converged. The “All Fit Statistics” table in Output 23.6.3 indicates that the generalized Pareto distribution (GPD) 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 23.6.3 Statistics of Fit for Interval-Censored Data

The SEVERITY Procedure

Input Data Set	
Name	WORK.GDENTAL

Model Selection

Distribution	Converged	AD	Selected
Burr	Yes	0.00103	Yes
Exp	Yes	0.09936	No
Gamma	Yes	0.04608	No
Igauss	Yes	0.12301	No
Logn	Yes	0.01884	No
Pareto	Yes	0.00739	No
Gpd	Yes	0.00739	No
Weibull	Yes	0.03293	No

All Fit Statistics

Distribution	-2 Log Likelihood	AIC	AICC	BIC	KS	AD
Burr	41.41112 *	47.41112	51.41112	48.31888	0.08974 *	0.00103 *
Exp	42.14768	44.14768 *	44.64768 *	44.45026 *	0.26412	0.09936
Gamma	41.92541	45.92541	47.63969	46.53058	0.19569	0.04608
Igauss	42.34445	46.34445	48.05874	46.94962	0.34514	0.12301
Logn	41.62598	45.62598	47.34027	46.23115	0.16853	0.01884
Pareto	41.45480	45.45480	47.16908	46.05997	0.11423	0.00739
Gpd	41.45480	45.45480	47.16908	46.05997	0.11423	0.00739
Weibull	41.76272	45.76272	47.47700	46.36789	0.17238	0.03293

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

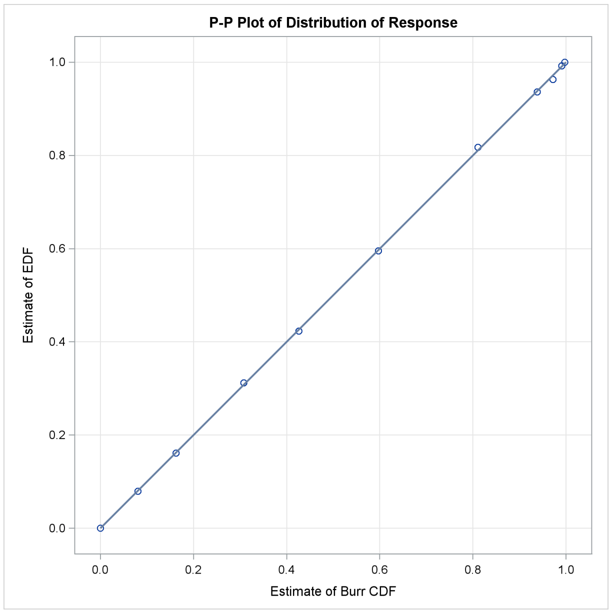
All Fit Statistics

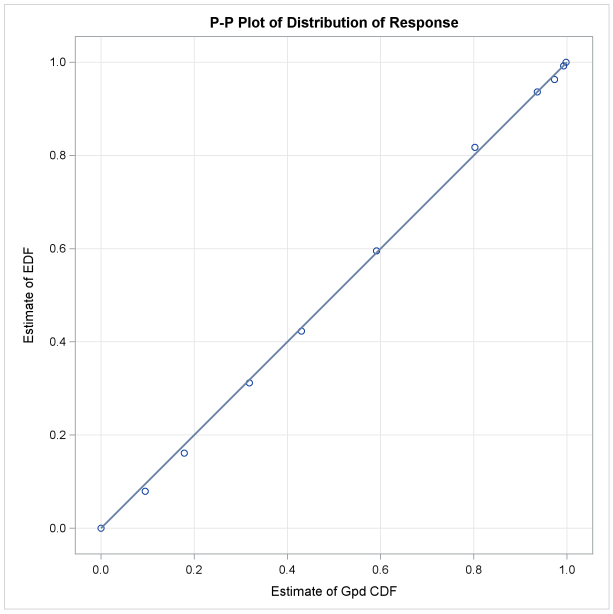
Distribution	CvM
Burr	0.0000816 *
Exp	0.01866
Gamma	0.00759
Igauss	0.02562
Logn	0.00333
Pareto	0.0009084
Gpd	0.0009084
Weibull	0.00472

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

The P-P plots of Output 23.6.4 show that both GPD and BURR have a close fit between EDF and CDF estimates, although BURR has slightly better fit, which is also indicated by the EDF-based statistics. Given that BURR is a generalization of the GPD and that the plots do not offer strong evidence in support of the more complex distribution, GPD seems like a good choice for this data.

Output 23.6.4 P-P Plots of Burr and GPD for Interval-Censored Data





Example 23.7: 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:

$$\begin{aligned} f(x; \theta, \rho, \Phi, p) &= \frac{1}{\theta} \left(p f_1(\frac{x}{\theta}; \Phi_1) + (1-p) f_2(\frac{x}{\theta}; \rho, \Phi_2) \right) \\ F(x; \theta, \rho, \Phi, p) &= p F_1(\frac{x}{\theta}; \Phi_1) + (1-p) F_2(\frac{x}{\theta}; \rho, \Phi_2) \end{aligned}$$

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:

$$\begin{aligned} \theta_1 &= \theta \\ \theta_i &= \rho_i \theta; \; i \in [2, K] \end{aligned}$$

In order to illustrate this approach, define a mixture of two lognormal distributions by using the following PDF function:

$$\begin{aligned} f(x; \mu, \sigma_1, p_2, \rho_2, \sigma_2) = & \frac{(1-p_2)}{\sigma_1 x \sqrt{2\pi}} \exp\left(\frac{-(\log(x) - \mu)^2}{2\sigma_1^2} \right) + \\ & \frac{p_2}{\sigma_2 x \sqrt{2\pi}} \exp\left(\frac{-(\log(x) - \mu - \log(\rho_2))^2}{2\sigma_2^2} \right) \end{aligned}$$

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 SEVERITY:

```
/*- 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 SEVERITY 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 SEVERITY 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

  $$\log(m_1) = \mu + \frac{\sigma_1^2}{2} \qquad \log(m_1) = \mu + \log(\rho_2) + \frac{\sigma_2^2}{2}$$

  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:

  $$\sigma_1^2 = 2(\log(m_1) - \mu) \qquad \sigma_2^2 = 2(\log(m_1) - \mu - \log(\rho_2))$$

  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 severity data=testlognmix print=all plots(histogram kernel)=all;
   loss y;
   dist slognmix2 logn;
run;
```

The comparison of the fit statistics of SLOGNMIX2 and LOGN, as shown in Output 23.7.1, confirms that the two-component mixture is certainly a better fit to these data than the single lognormal distribution.

Output 23.7.1 Comparison of Fitting One versus Two Lognormal Components to Mixture Data

All Fit Statistics

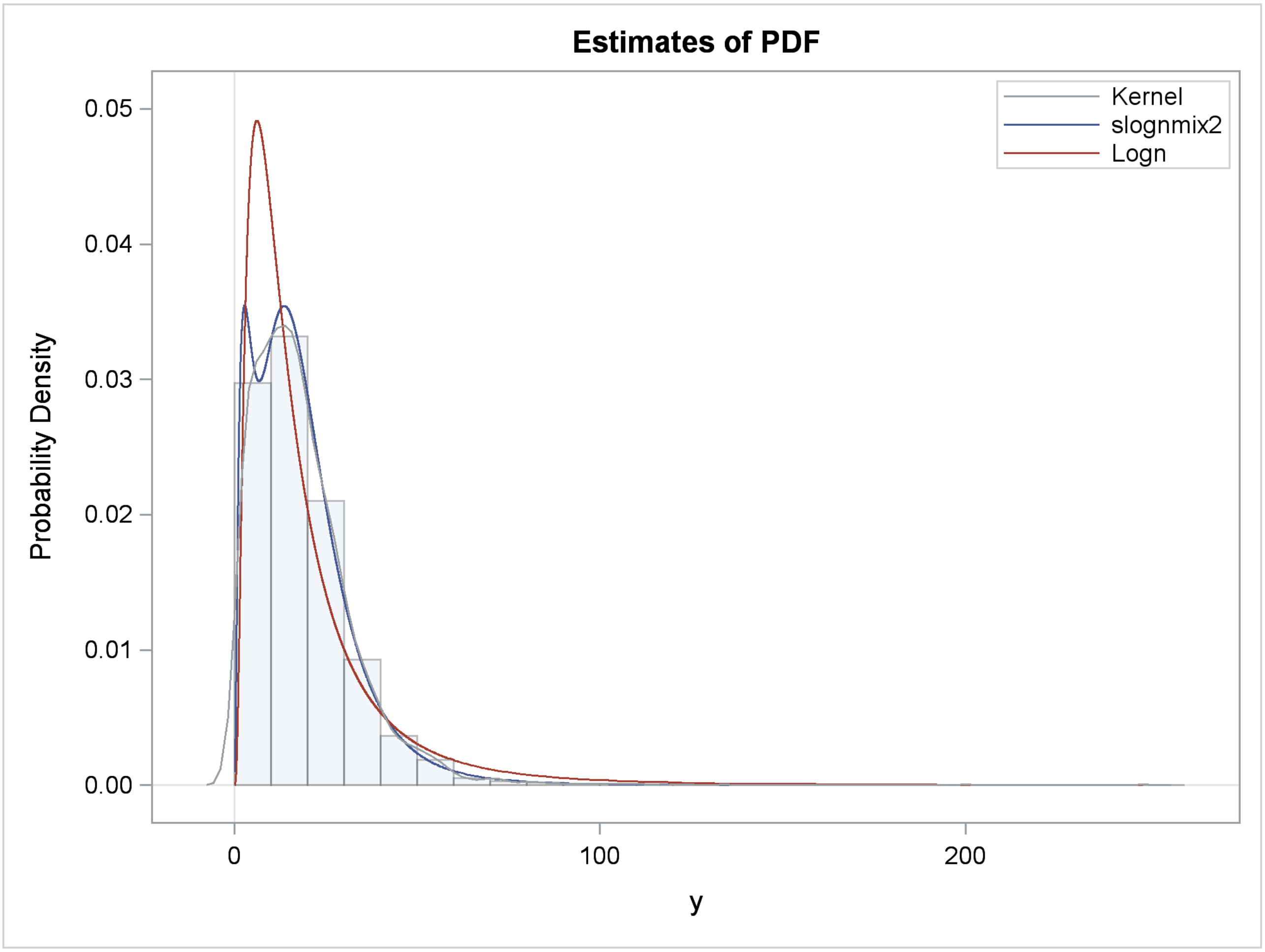
Distribution	-2 Log Likelihood	AIC	AICC	BIC	KS	AD	CvM
slognmix2	38343 *	38353 *	38353 *	38386 *	0.52221 *	0.19843 *	0.02728 *
Logn	39073	39077	39077	39090	5.86522	66.93414	11.72703

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

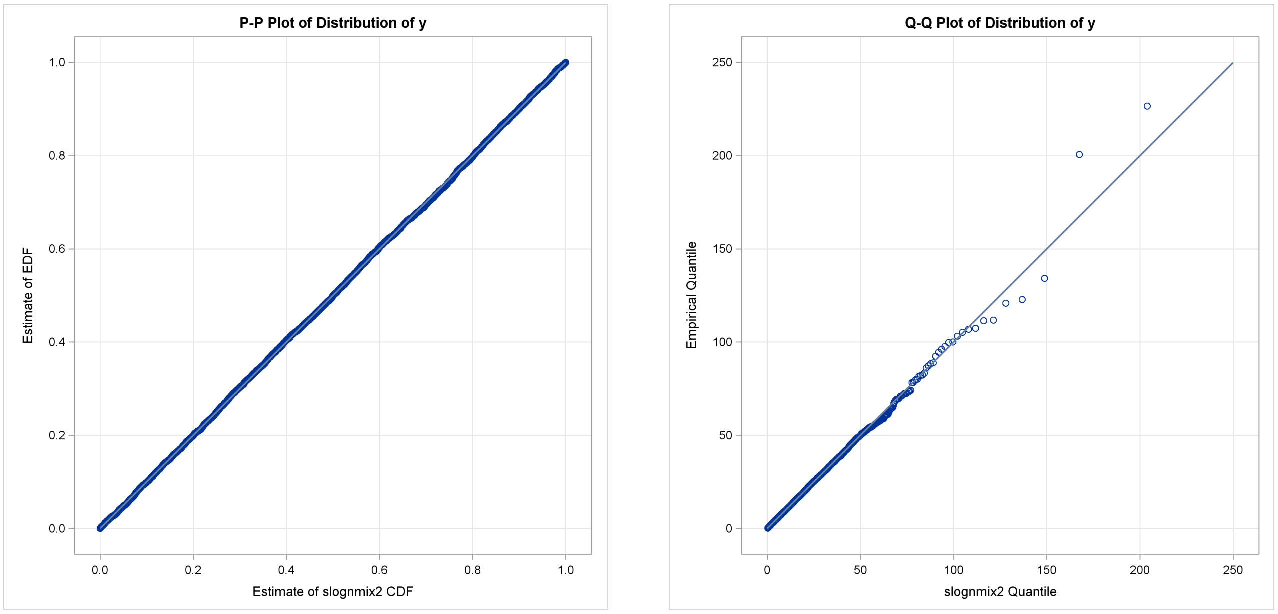
The comparative plot of probability densities in Output 23.7.2 shows that the density function of the mixture distribution is bimodal. In fact, one of the key motivations for using mixture distributions is to find better-fitting models for multimodal data.

The P-P and Q-Q plots in Output 23.7.3 visually confirm that SLOGNMIX2 fits the data very well.

Output 23.7.2 Comparison of PDF Estimates of the Fitted Models



Output 23.7.3 P-P and Q-Q Plots to Evaluate SLOGNMIX2 Fit



The detailed results for the SLOGNMIX2 distribution are shown in Output 23.7.4. 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 23.7.4 Detailed Estimation Results for the SLOGNMIX2 Distribution

The SEVERITY 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	Estimate	Standard Error	t Value	Approx Pr > |t|
Mu	3.00922	0.01554	193.68	<.0001
Sigma1	0.49516	0.01451	34.13	<.0001
P2	0.40619	0.02600	15.62	<.0001
Rho2	0.37212	0.02038	18.26	<.0001
Sigma2	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 23.8: 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), such that the probability of experiencing a loss value greater than the VaR is very low. 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 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.

First, the following PROC FCMP steps define the functions to compute the mean for each of the 10 predefined distributions that are available in the Sashelp.Svrtdist library:

```
/*--------- Define 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;
```

The following statements include the Work.Means library in the CMPLIB= system option and submit a PROC SEVERITY step to estimate the scale regression models for various distributions by using a lognormal sample in the Work.Test_sev8 data set:

```
   /*----- Fit all distributions and generate scoring functions ------*/
   options cmplib=work.means;
   proc severity data=test_sev8 outest=est print=all plots=none;
      loss y;
      scalemodel x1-x5;
      dist _predefined_ stweedie;
      outscorelib outlib=scorefuncs commonpackage;
   run;
```

The SAS statements that simulate the sample in the Work.Test_sev8 data set are available in the PROC SEVERITY sample program *sevex08.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 SEVERITY sets the CMPLIB system option to the following value:

```
    (work.means sashelp.svrtdist work.scorefuncs)
```

The "All Fit Statistics" table in Output 23.8.1 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.

Output 23.8.1 Comparison of Fitted Scale Models for Mean and VaR Illustration

The SEVERITY Procedure

All Fit Statistics

Distribution	-2 Log Likelihood	AIC	AICC	BIC	KS	AD
stweedie	460.65755	476.65755	476.95083	510.37441	10.44548	64571
Burr	451.42238	467.42238	467.71565	501.13924	10.32782	42254
Exp	1515	1527	1527	1552	8.85827	29917
Gamma	448.28222	462.28222	462.50986	491.78448	10.42272	63712
Igauss	444.44512	458.44512	458.67276	487.94738	10.33028	83195
Logn	444.43670 *	458.43670 *	458.66434 *	487.93895 *	10.37035	68631
Pareto	1515	1529	1529	1559	8.85775 *	29916 *
Gpd	1515	1529	1529	1559	8.85827	29917
Weibull	527.28676	541.28676	541.51440	570.78902	10.48084	72814

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

All Fit Statistics

Distribution	CvM
stweedie	37.07705
Burr	37.19808
Exp	23.98267
Gamma	37.19450
Igauss	37.30880
Logn	37.18553
Pareto	23.98149 *
Gpd	23.98267
Weibull	36.36039

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

You can examine the scoring functions that are written to the Work.Scorefuncs library by using the FCMP Function Editor that is available in the Display Manager session of Base SAS when you select **Solutions**→**Analysis** from the main menu. For example, PROC SEVERITY automatically generates and submits the following PROC FCMP statements to define the scoring functions 'SEV_MEAN_LOGN' and 'SEV_QUANTILE_IGAUSS':

```
proc fcmp library=(work.means 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;
```

An important point to note is that the *dist*_MEAN distribution functions are not available in the original definition of any of the distributions in the Sashelp.Svrtdist library. PROC SEVERITY detects the availability of those functions in the Work.Means library that is included in the value of the CMPLIB= system option just before submitting the PROC SEVERITY step. Each *dist*_MEAN distribution function has a signature that matches the signature of a distribution function of the respective distribution, so PROC SEVERITY creates the corresponding scoring functions. Specifying the COMMONPACKAGE option in the OUTSCORELIB statement causes the name of the scoring function to take the form SEV_MEAN_*dist*. 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.

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:

```
   /*--- 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 preceding steps use a VaR level of 97.5%.

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 SEVERITY 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 23.8.2 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 SEVERITY. 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 23.8.2 Comparison of Mean and VaR Estimates of Two Scoring Methods

```
                         Observation Summary

                    Observation      Base  Compare

                    First Obs           1        1
                    Last  Obs          10       10

Number of Observations in Common: 10.
Total Number of Observations Read from WORK.SCORESWITHOUTEST: 10.
Total Number of Observations Read from WORK.SCORES: 10.

Number of Observations with Some Compared Variables Unequal: 0.
Number of Observations with All Compared Variables Equal: 10.


                      Values Comparison Summary

 Number of Variables Compared with All Observations Equal: 9.
 Number of Variables Compared with Some Observations Unequal: 0.
 Total Number of Values which Compare Unequal: 0.
 Total Number of Values not EXACTLY Equal: 40.
 Maximum Difference Criterion Value: 2.0963E-13.
```

Example 23.9: 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 SEVERITY sample program *sevex09.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 1697. 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 SEVERITY 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 severity 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 SEVERITY 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 23.9, "GLM Parameterization of Classification Variables and Effects," on page 1700.

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 23.9.1. 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 23.9.1 Class Level Information Table

The SEVERITY 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 23.9.2. The "Initial Parameter Values and Bounds" table is important especially because the preceding PROC SEVERITY 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 23.9.2 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 23.9.3 indicates that the scale regression model for the lognormal distribution has converged with the default optimization technique in five iterations.

Output 23.9.3 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 23.9.4 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 23.9.4 Parameter Estimates for the Scale Regression with Class and Interaction Effects

Parameter Estimates

Parameter	Estimate	Standard Error	t Value	Approx Pr > |t|
Mu	5.08874	0.05768	88.23	<.0001
Sigma	0.55774	0.01119	49.86	<.0001
carType SUV	0.62459	0.04452	14.03	<.0001
gender Female	0.44145	0.04885	9.04	<.0001
carSafety	-0.82942	0.08371	-9.91	<.0001
income	-0.35212	0.07657	-4.60	<.0001
carType*education SUV AdvancedDegree	0.39393	0.07351	5.36	<.0001
carType*education SUV College	0.76532	0.05723	13.37	<.0001
carType*education Sedan AdvancedDegree	-0.61064	0.05387	-11.34	<.0001
carType*education Sedan College	-0.35210	0.03942	-8.93	<.0001
income*gender Female	-0.01486	0.06629	-0.22	0.8226
carSafety*income	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 SEVERITY step as an INSTORE= item store in a new PROC SEVERITY step as follows:

```
/* Refit scale regression model on new data different types of regression effects */
proc severity 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 SEVERITY uses the parameter estimates in the INSTORE= item store to initialize the distribution and regression parameters.

References

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*, volume 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 (with discussion)," *Journal of the Royal Statistical Society, Series B*, 49, 127–162.

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.

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 J. K. Ghosh and J. Roy, eds., *Statistics: Applications and New Directions—Proceedings of the Indian Statistical Institute Golden Jubilee International Conference*, 579–604.

Subject Index

Syntax Index