A SAS® IML Library for the Metalog Distribution

Rick Wicklin, SAS Institute Inc.

March 31, 2023

ABSTRACT

The metalog family of distributions (Keelin 2016a) is a flexible family of distributions that can model a wide range of univariate data distributions. This paper shows how to use a library of SAS® IML functions to model data by using the metalog distributions.

THE METALOG DISTRIBUTION

You can use a family of probability distributions to model univariate data. This often requires domain-specific knowledge to decide which distribution models the data-generating process. Common continuous distributions that model data include the normal, lognormal, Weibull, gamma, and beta distributions.

Sometimes the usual collection of well-known distributions is not sufficiently flexible to model data. Therefore, researchers developed flexible *systems* of distributions that can model a wide range of distributional shapes. Many popular systems use the sample moments of the data to identify a family within the system, then estimate the parameters of the selected family by fitting the distribution to the data. Popular systems include the following:

- The Pearson system: Karl Pearson used the normal distribution and seven other families to construct a system that can match any sample skewness and kurtosis. The Pearson system is described in Johnson, Kotz, and Balakrishnan (1994, Chapter 12).
- The Johnson system: Norman Johnson used the normal distribution and three other families to construct a system that can match any sample skewness and kurtosis. The Johnson system is described in Johnson, Kotz, and Balakrishnan (1994, Chapter 12). It includes the following:
  - lognormal family, which models semibounded distributions on an interval $[L, \infty)$
  - S_B family, which models bounded distributions on an interval $[L, U]$
  - S_U family, which models unbounded distributions on $(-\infty, \infty)$

  You can use the negative of a lognormal random variable to model a semibounded distribution on an interval $(-\infty, U]$.
- The Fleishman system: The Fleishman system (Fleishman 1978; Headrick 2002, 2010) is a polynomial transformation of a normal distribution. It cannot model distributions that have extreme skewness and kurtosis values.

The metalog distribution is an alternative family of distributions that can model many different shapes for bounded, semibounded, and unbounded distributions. To use the metalog distribution, do the following:

- Choose the *type* of the distribution from the four available types: unbounded, semibounded with a lower bound, semibounded with an upper bound, or bounded. This choice is often based on domain-specific knowledge of the data-generating process.

- Choose the *number of terms*, k, to use in the metalog model. A small number of terms ($3 \leq k \leq 6$) results in a smooth model. A larger number of terms ($k \geq 7$) can fit data distributions for which the density appears to have multiple peaks.
- Fit the quantile function of the metalog distribution to the data. This estimates the k parameters in the model.

The metalog system is designed to model continuous distributions for which the quantile function is strictly increasing as a function of the probability on the open interval $p \in (0, 1)$. Equivalently, it models distributions for which the cumulative distribution function (CDF) is strictly increasing.

Keelin (2016a) uses ordinary least squares (OLS) to fit the data to the quantile function of the metalog distribution. This can lead to an unfortunate situation: the fitted metalog coefficients might not correspond to a valid distribution. An invalid model is called *infeasible*. If you plot the CDF of an infeasible model, the graph will not be increasing. Equivalently, the probability density function (PDF) of the model will contain nonpositive values.

Infeasibility is somewhat complicated, but some situations are easy to understand. The coefficients in a metalog model are related to the moments of the distribution. If the coefficients would lead to a "negative variance," those coefficients would be infeasible. Similarly, if the coefficients would produce a kurtosis value that is extremely large relative to the skewness value, those coefficients would be infeasible.

It is relatively easy to create examples for which a k-term model is infeasible. When fitting a metalog model to data, you should always verify that the resulting model is feasible. If it is not, reduce the number of terms in the model. Some data cannot be fit when $k > 2$. For example, data that have one or more extreme observations often lead to an infeasible set of parameter estimates.

DEFINITIONS OF THE METALOG DISTRIBUTION

Most common continuous probability distributions are parameterized on the data scale. That is, if X is a continuous random variable, then the CDF is a function that is defined on the set of outcomes for X. You can write the CDF as some increasing function $p = F(x) = P(X \leq x)$.

From the definition of a continuous CDF, you can obtain the quantile function and the PDF. The quantile function is defined as the inverse CDF: $x = F^{-1}(p)$ for $p \in (0, 1)$. The PDF is defined as the derivative with respect to x of the CDF.

Keelin (2016a) specifies the definitions of the metalog quantile function and PDF function. The metalog distribution reverses the usual convention by parameterizing the quantile function as $x = Q(p)$. (Keelin uses the notation $M(p)$ for the quantile function.) From the quantile function, you can obtain the inverse function, which is the CDF. From the CDF, you can obtain the PDF. From the quantile function, you can generate random variates. Thus, knowing the quantile function is the key to working with the metalog distribution.

The Unbounded Metalog Distribution

When you fit an unbounded metalog model, you estimate the parameters for which the model's predicted values $(p, Q(p))$ are close (in a least squares sense) to the data pairs (p, x). Here, p is a vector of cumulative probabilities, x is a vector of observed data values, and $Q(p)$ is the quantile function of a metalog distribution.

The predicted values are obtained by using a linear regression model. The unbounded model has a design matrix that uses the following basis functions:

- a vector of 1s
- a vector that contains the values logit$(p) = \log(p/(1-p))$
- a vector that contains the values $c(p) = p - 1/2$

Specifically, the columns of the design matrix involve powers of the vector $c(p)$ and interactions with the logit(p). If $M = M_1|M_2|\ldots|M_k$ is the design matrix, then the columns are as follows:

- M_1 is a column of 1s.
- M_2 is the column logit(p).

- M_3 is the interaction term $c(p)$logit(p).
- M_4 is the linear term $c(p)$.
- M_5 is the quadratic term $c(p)^2$.
- M_6 is the interaction term $c(p)^2$logit(p).
- Additional columns are of the form $c(p)^j$ or $c(p)^j$logit(p).

These formulas are given in vector notation, which means that the *i*th row of M_2 is logit(p_i), the *i*th row of M_3 is $c(p_i)$logit(p_i), and so forth.

Given the design matrix, M, the least squares parameter estimates are the elements of the vector, a, which solves the matrix equations $M'Ma = M'x$. An estimate of the quantile function is given by $Q(p) = Ma$. By differentiating this expression with respect to p, you can obtain a formula for the PDF for the model (Keelin 2016a, Eq. 6, p. 254).

From these equations, you can see that not every set of coefficients corresponds to an increasing quantile function. For example, when $k = 2$, $Q(p) = a_1 + a_2$logit(p). For $Q(p)$ to be an increasing function of p, it is necessary that a_2 be strictly positive.

The Semibounded and Bounded Metalog Distributions

The semibounded and bounded metalog distributions are obtained by transforming the unbounded metalog model.

The semibounded metalog distribution on the interval $[L, \infty)$ is obtained by fitting an unbounded k-term distribution to the transformed data $z = \log(x - L)$. This gives the quantile function, $Q(p)$, for z. The quantile function for the semibounded distribution on $[L, \infty)$ is then defined as $Q_L(p) = L + \exp(Q(p))$ for $0 < p < 1$ and $Q_L(0) = L$.

Similarly, the quantile function for the semibounded distribution on $(-\infty, U)$ is obtained by fitting an unbounded k-term distribution to the transformed data $z = \log(U - x)$. This gives the quantile function, $Q(p)$, for z. The quantile function for the semibounded distribution is then defined as $Q_U(p) = U - \exp(-Q(p))$ for $0 < p < 1$ and $Q_U(1) = U$.

Finally, the quantile function for the bounded metalog distribution on $[L, U]$ is obtained by fitting an unbounded k-term distribution to the transformed data $z = \log((x - L)/(U - x))$. This gives the quantile function, $Q(p)$, for z. The quantile function for the bounded distribution is then defined by $Q_B(p) = (L + U\exp(Q(p)))/(1 + \exp(Q(p)))$ for $0 < p < 1$ and the boundary conditions $Q_B(0) = L$ and $Q_B(1) = U$.

As mentioned previously, for every distribution, you can obtain the PDF function from the quantile function.

Random Variates from the Metalog Distribution

Recall that you can generate a random sample from any continuous probability distribution by using the inverse CDF method. You generate a random variate u from the uniform distribution $U(0, 1)$, then compute $x = F^{-1}(u)$, where F is the CDF function and F^{-1} is the inverse CDF, which is the quantile function. You can prove that x is a random variate that is distributed according to F.

For most distributions, the inverse CDF is not known explicitly, which means that you must solve for x as the root of the nonlinear equation $F(x) - u = 0$. However, the metalog distribution supports direct simulation because the quantile function is defined explicitly as a function of probability.

For the unbounded metalog distribution, you can generate random variates by generating $u \sim U(0, 1)$ and computing $x = Q(u)$, where Q is the quantile function. The semibounded and bounded cases are treated similarly.

OVERVIEW OF THE SAS IML METALOG FUNCTIONS

The metalog library contains many low-level computational routines and a small number of high-level routines. All high-level computational routines begin with the prefix ML_. The high-level routines are intended for public use and are documented in this paper.

All the examples in this paper assume that you have downloaded and installed the metalog functions according to the methods that are described in Appendix A. In addition, the examples assume that you have loaded the modules into your SAS IML session. For example, you can use the LOAD statement to load the modules, as follows:

```
load module= _all_;   /* load all metalog modules */
```

The modules are object-oriented, which means that you must first construct a "metalog object." The object contains information about the distribution, such as the order and whether it is bounded, semibounded, or unbounded. All subsequent operations are performed by passing the metalog object as an argument to functions. You can classify the functions into four categories by using the following functions:

- *Constructors* create a metalog object from data. By default, the constructors fit a five-term unbounded metalog model.
- *Query functions* enable you to query properties of the distribution.
- *Computational functions* enable you to compute the PDF and quantiles of the distribution and to generate random variates.
- *Plotting routines* create graphs of the CDF and PDF of the distribution. The plotting routines are not available in the **iml** action.

To use the SAS IML metalog functions, you must create a metalog object by using one of the following functions:

- ML_CreateFromData function, which creates a metalog object by using the empirical cumulative distribution function (ECDF) of univariate data. Use this function to model univariate data.
- ML_CreateFromCDF function, which creates a metalog object by using quantile-probability pairs. This specification is common in decision analysis applications in which experts estimate the quantiles of a set of probability values.
- ML_CreateFromCoef function, which creates a metalog object directly from a set of parameter coefficients. This method is used primarily for testing and simulation.

After creating a metalog object, you can ask for information about the model by using the following:

- The ML_Summary subroutine displays a summary of the model for a metalog object.
- The ML_Order function returns the order of the metalog model. The order is the number of coefficients in the model.
- The ML_Bounds function returns the bounds for the metalog model.
- The ML_BoundType function returns a character string that indicates whether the model is unbounded, semibounded, or bounded.
- The ML_Coef function returns the estimates of the coefficient parameters for the model.
- The ML_IsFeasible function returns 1 if the coefficients result in a valid distribution and 0 if the distribution is infeasible. A valid CDF is monotone-increasing. Equivalently, a valid PDF is nonnegative everywhere.

After creating a metalog object, you can compute several quantities of the distribution by using the following functions:

- The ML_PDF function returns the PDF (density) of the metalog model.

- The ML_Quantile function returns the quantiles that are associated with a specified set of cumulative probabilities.
- The ML_Rand function returns a random sample from the metalog distribution.

If you are running PROC IML, you can request graphs of the metalog model by using the following functions:

- The ML_PlotCDF function creates a series plot of the CDF of the model.
- The ML_PlotECDF subroutine creates a graph that overlays a scatter plot of the empirical distribution and the CDF of the model.
- The ML_PlotEQuantile subroutine creates a graph that overlays a scatter plot of the empirical quantiles and the quantile function of the model.
- The ML_PlotPDF function creates a series plot of the PDF of the model.
- The ML_PlotQuantile function creates a series plot of the quantile function of the model.

The following sections define the syntax for the high-level metalog functions.

SYNTAX OF THE SAS IML METALOG FUNCTIONS

ML_CreateFromCDF Function

ML_CreateFromCDF(*x, p < , order> < , bounds>* **);**

The ML_CreateFromCDF function returns a metalog object that represents a metalog model. The model is defined by using quantile-probability pairs. This specification is common in decision analysis applications in which experts estimate the quantiles of a set of probability values.

The required input arguments are as follows:

x	is a vector of quantile values.
p	is a vector of cumulative probabilities. The *x* and *p* vectors must be the same size.

The ordered pairs $\{(x_i, p_i)\} | i = 1, \ldots, n\}$ are an approximation of the empirical cumulative distribution function (ECDF), where n is the number of elements in the vectors.

The optional input arguments are as follows:

order is the number of terms in the metalog model. If the value is not specified, then $\min(n, 5)$ terms are used.

bounds is a 1×2 vector that specifies the bounds for the metalog model. If the value is not specified, the support of the model is unbounded. Valid values are as follows:

- A vector of missing values specifies an unbounded model.
- A vector of the form `{L, .}` specifies a semibounded model that is bounded below.
- A vector of the form `{., U}` specifies a semibounded model that is bounded above.
- A vector of the form `{L, U}` specifies a bounded model.

The following example creates an unbounded metalog model according to an expert's opinion about the length of time required to complete a project. The expert believes the following:

- There is a 10% chance that the project will be completed in 5 days or less.
- There is a 25% chance that the project will be completed in 8 days or less.
- There is a 50% chance that the project will be completed in 15 days or less.
- There is a 75% chance that the project will be completed in 20 days or less.

- There is a 90% chance that the project will be completed in 30 days or less.

By default, the metalog functions try to fit a five-term model to the data. For small data, a five-term model might not produce a valid distribution. Therefore, the following statements fit a three-term metalog model to the expert's beliefs. You can call the ML_Summary subroutine to print a summary of the model and to see the model coefficients. The output in Figure 1 shows that the three-term model is valid. If you want to visualize the model, you can use the ML_PlotECDF subroutine to display a graph that overlays the model's CDF and the expert's estimates. The graph is shown in Figure 2.

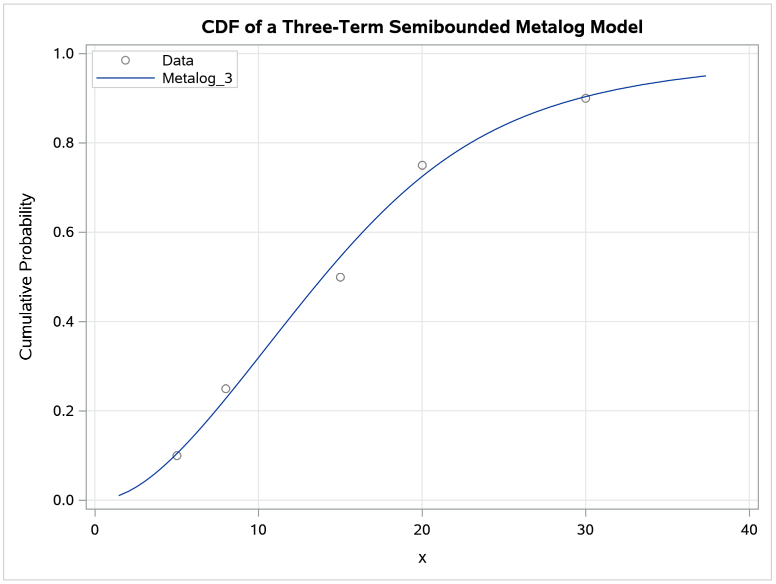
```
x = {  5,    8,  15,   20,  30};           /* number of days */
p = {0.1, 0.25, 0.5, 0.75, 0.9};           /* probability of completion */
MLObj = ML_CreateFromCDF(x, p, 3, {0,.});  /* three-term semibounded model */
call ML_Summary(MLObj);

title "CDF of a Three-Term Semibounded Metalog Model";
prob = do(0.01, 0.95, 0.01);
call ML_PlotECDF(MLObj, prob);
```

Figure 1 Summary of a Three-Term Semibounded Metalog Model

Model Summary	
Order	3
Type	SL
Bounds	[0,.]
Is Feasible	1

	Estimate
a1	2.6359613
a2	0.4095903
a3	-0.167288

Figure 2 CDF of a Three-Term Semibounded Metalog Model



ML_CreateFromData Function

ML_CreateFromData(*x, < , order> < , bounds> < , method>* **);**

The ML_CreateFromData function creates a metalog object by using the empirical cumulative distribution function (ECDF) of input data. Keelin (2016a) refers to this as the equally likely data (ELD) metalog distribution. Instead of using (x, y) pairs as the input parameters, as for the ML_CreateFromCDF function, you specify only the data (x) values, and the ECDF (y) is computed automatically.

The ML_CreateFromData function requires one input argument:

x	is a vector of data values.

The routine first estimates the empirical distribution of the data. The data and the associated cumulative probabilities are then used to fit the metalog model, as described for the ML_CreateFromCDF function.

The optional arguments are as follows:

order is the number of terms in the metalog model. If the value is not specified, then $\min(n, 5)$ terms are used.

bounds is a 1×2 vector that specifies the bounds for the metalog model. If the value is not specified, the support of the model is unbounded. Valid values are as follows:

- A vector of missing values specifies an unbounded model.
- A vector of the form **{L, .}** specifies a semibounded model that is bounded below.
- A vector of the form **{., U}** specifies a semibounded model that is bounded above.
- A vector of the form **{L, U}** specifies a bounded model.

method is a string that specifies the method of computing the ECDF from the data. Let $R_1, R_2, \ldots, R_n$ be the ranks of the n data values. First, all methods rank the data values. Tied values are ranked arbitrarily. Valid methods are as follows:

"Blom": Blom's method estimates the cumulative probability for the ith data value by using the score $p_i = (R_i - 3/8)/(n + 1/4)$.

"Haven": Haven's method uses the score $p_i = (R_i - 1/2)/n$.

"Tukey": Tukey's method uses the score $p_i = (R_i - 1/3)/(n + 1/3)$.

"VW": Van der Waerden's method uses the score $p_i = R_i/(n + 1)$. This is the default method.

The following statements specify data and metalog models for the ECDF of the data. By default, the ECDF is found by using van der Waerden's method. By default, the ML_CreateFromData function fits a five-term unbounded metalog distribution, as shown in the following example:

```
x = {14, 18, 22, 24, 26, 31, 32, 38};
MLObj = ML_CreateFromData(x);   /* default order and unbounded */
```

You can specify the following optional arguments to override the defaults:

```
order = 4;
bounds = {0 .};     /* semibounded model */
method="Blom";
MLObj2 = ML_CreateFromData(x, order, bounds, method);
```

If you need to obtain the values for the ECDF, you can get them from the object. For example, in the preceding statements, **MLObj$'cdf'** contains the ECDF values for van der Waerden's method, and **MLObj2$'cdf'** contains the ECDF values for Blom's method.

ML_CreateFromCoef Function

ML_CreateFromCoef(*a* < , *bounds* > **);**

The ML_CreateFromCoef function creates a metalog object directly from a set of parameter coefficients. This function is primarily used for simulation studies, not for data modeling. Recall that not every vector of coefficients corresponds to a valid distribution.

One input argument is required:

a is a vector of k coefficients, which determines a k-term metalog model.

The optional argument is as follows:

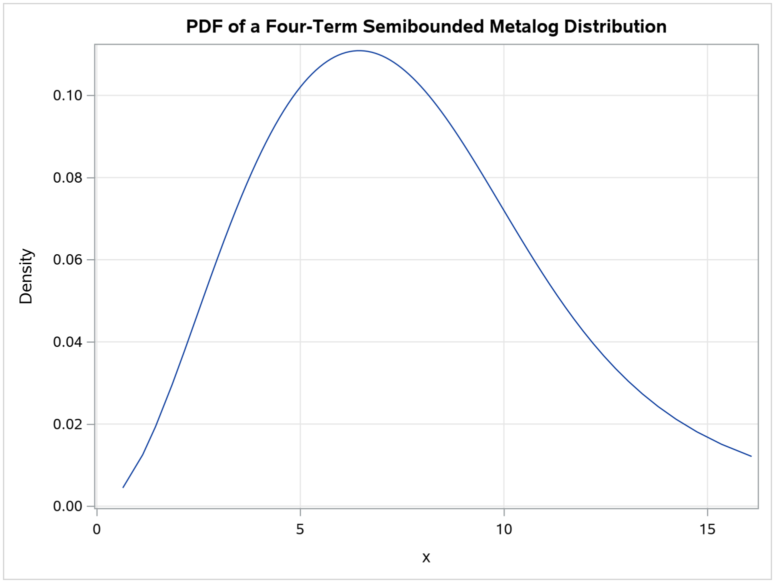
bounds is a 1×2 vector that specifies the bounds for the metalog model. If the value is not specified, the support of the model is unbounded. Valid values are as follows:

- A vector of missing values specifies an unbounded model.
- A vector of the form **{L, .}** specifies a semibounded model that is bounded below.
- A vector of the form **{., U}** specifies a semibounded model that is bounded above.
- A vector of the form **{L, U}** specifies a bounded model.

The following statements define a four-term bounded metalog distribution by specifying the coefficients of the model. The PDF of the distribution is shown in Figure 3.

```
a   = {2, 0.3, -0.1, 0.06};
bounds = {0 .};
MLObj = ML_CreateFromCoef(a, bounds);

title "PDF of a Four-Term Semibounded Metalog Distribution";
prob = {0.001 0.005} || do(0.01, 0.95, 0.01);
run ML_PlotPDF(MLObj, prob);
```

Figure 3 PDF of a Four-Term Semibounded Metalog Distribution



CALL ML_Summary

CALL ML_Summary(*MLObj***);**

The ML_Summary subroutine displays a summary of the model for a metalog object. The routine displays two tables:

- The “Summary” table displays the order, type, and bounds for the model. It also displays whether the fitted model is a feasible distribution.
- The “Estimate” table displays the parameter estimates for the fitted model.

The required input argument is a metalog object. You can create a metalog object by using the ML_CreateFromCDF function, the ML_CreateFromCoef function, or the ML_CreateFromData function.

See the ML_CreateFromCDF function for an example that uses the ML_Summary subroutine.

ML_Order Function

ML_Order(*MLObj***);**

The ML_Order function returns the order of the metalog model, which is an integer. The order is the number of coefficients in the model. The following example retrieves several bits of information about a metalog model. The information is shown in Figure 4.

```
x = {14, 18, 22, 24, 26, 31, 32, 38};
MLObj = ML_CreateFromData(x);   /* default order and unbounded */
/* query information about the model */
order = ML_Order(MLObj);
bounds = ML_Bounds(MLObj);
type = ML_BoundType(MLObj);
isFeas = ML_IsFeasible(MLObj);
print order bounds type isFeas;

coef = ML_Coef(MLObj);
print coef;
```

Figure 4 Information about a Five-Term Unbounded Metalog Model

order	bounds	type	isFeas
5	. .	U	1

coef
25.603708
5.3803684
4.675885
1.9483888
-22.83967

ML_Bounds Function

ML_Bounds(*MLObj*);

The ML_Bounds function returns the bounds for the metalog model. The return value is a 1×2 numerical vector. The form of the value is one of the following:

- A vector of missing values specifies an unbounded model.
- A vector of the form `{L, .}` specifies a semibounded model that is bounded below.
- A vector of the form `{., U}` specifies a semibounded model that is bounded above.
- A vector of the form `{L, U}` specifies a bounded model.

The required input argument is a metalog object. You can create a metalog object by using the ML_CreateFromCDF function, the ML_CreateFromCoef function, or the ML_CreateFromData function.

See the ML_Order Function for an example that uses the ML_Bounds function.

ML_BoundType Function

ML_BoundType(*MLObj*);

The ML_BoundType function returns a character string that indicates whether the model is unbounded, semibounded, or bounded. The valid values are "U" for an unbounded distribution, "SL" for a semibounded distribution that has a lower bound, "SU" for a semibounded distribution that has an upper bound, and "B" for a bounded distribution.

The required input argument is a metalog object. You can create a metalog object by using the ML_CreateFromCDF function, the ML_CreateFromCoef function, or the ML_CreateFromData function.

See the ML_Order Function for an example that uses the ML_BoundType function.

ML_Coef Function

ML_Coef(*MLObj*);

The ML_Coef function returns the estimates for the coefficient parameters for a metalog model.

The required input argument is a metalog object. You can create a metalog object by using the ML_CreateFromCDF function, the ML_CreateFromCoef function, or the ML_CreateFromData function.

See the ML_Order Function for an example that uses the ML_Coef function.

ML_IsFeasible Function

ML_IsFeasible(*MLObj*);

The ML_IsFeasible function returns 1 if the coefficients result in a valid distribution. A valid CDF must be monotone-increasing. Equivalently, a valid PDF must be nonnegative everywhere. If the distribution is not valid, the ML_IsFeasible function returns 0.

The required input argument is a metalog object. You can create a metalog object by using the ML_CreateFromCDF function, the ML_CreateFromCoef function, or the ML_CreateFromData function.

For two-term and three-term metalog models, feasibility is determined from the coefficients of the model. For a four-term model, Keelin (2016b) provides approximate formulas that determine feasibility. For higher-order models, the function determines feasibility by evaluating the PDF at a grid of points and verifying that the PDF at those points is not negative.

See the ML_Order Function for an example that uses the ML_IsFeasible function.

ML_PDF Function

ML_PDF(*MLObj, p***);**

The ML_PDF function returns the PDF (density) values that are associated with a specified vector of cumulative probabilities. Notice that metalog models are parameterized by using the cumulative probability, not quantiles. Accordingly, the second argument to this function contains probabilities, not values on the data scale.

The function requires two input arguments:

MLObj	is a metalog object. You can create a metalog object by using the ML_CreateFromCDF function, the ML_CreateFromCoef function, or the ML_CreateFromData function.
p	is a vector of cumulative probabilities.

```
x = {14, 18, 22, 24, 26, 31, 32, 38};
MLObj = ML_CreateFromData(x);   /* default order and unbounded */

p = {0.01, 0.1, 0.25, 0.5, 0.75, 0.90, 0.99}; /* cumulative probability values */
Q = ML_Quantile(MLObj, p);                    /* the corresponding quantiles */
PDF = ML_PDF(MLObj, p);                       /* the corresponding densities */
print p Q PDF;
```

Figure 5 Quantiles and Densities for a Set of Cumulative Probabilities

p	Q	PDF
0.01	4.9700194	0.0031758
0.1	13.457716	0.0204305
0.25	19.062439	0.0325818
0.5	25.603708	0.0426078
0.75	31.858511	0.0326847
0.9	38.660183	0.0134203
0.99	56.326316	0.0012887

ML_Quantile Function

ML_Quantile(*MLObj, p***);**

The ML_Quantile function returns the quantiles that are associated with a specified vector of cumulative probabilities. The quantile function is the inverse CDF function.

The function requires two input arguments:

MLObj	is a metalog object. You can create a metalog object by using the ML_CreateFromCDF function, the ML_CreateFromCoef function, or the ML_CreateFromData function.
p	is a vector of cumulative probabilities.

See the ML_PDF Function for an example that uses the ML_Quantile function.

ML_Rand Function

ML_Rand(*MLObj, n***);**

The ML_Rand function returns a random sample of size n from the metalog distribution. To obtain a reproducible sample, you should call the RANDSEED subroutine prior to calling the ML_Rand function.

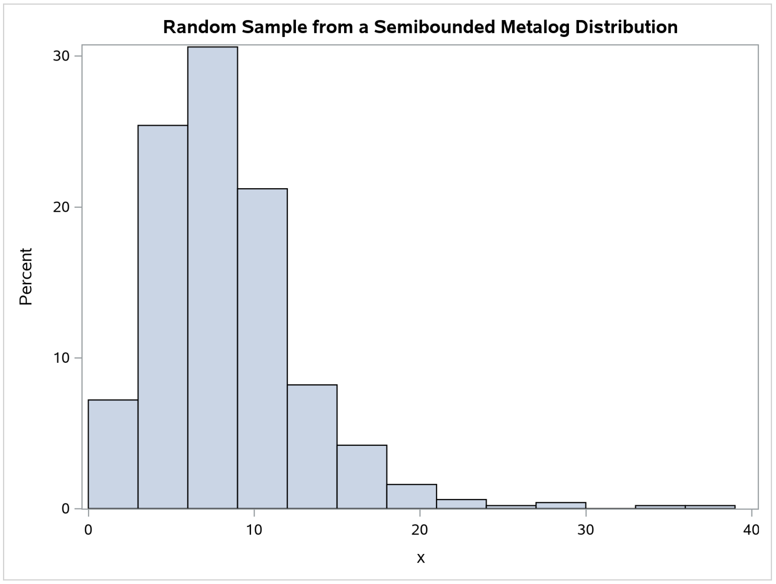
The function requires two input arguments:

MLObj	is a metalog object. You can create a metalog object by using the ML_CreateFromCDF function, the ML_CreateFromCoef function, or the ML_CreateFromData function.
n	is the sample size of the resulting vector of random variates.

The following statements generate a random sample of size 500 from a semibounded metalog distribution that has specified coefficients. Figure 6 shows the distribution of the random sample.

```
a = {2, 0.3, -0.1, 0.06};
bounds = {0 .};
MLObj = ML_CreateFromCoef(a, bounds);
call randseed(123);
x = ML_Rand(MLObj, 500);

title "Random Sample from a Semibounded Metalog Distribution";
run histogram(x);
```

Figure 6 Random Sample from a Semibounded Metalog Distribution



CALL ML_PlotCDF

ML_PlotCDF(*MLObj* < , *p*> **);**

The ML_PlotCDF subroutine creates a series plot of the CDF of the model.

The subroutine requires one input argument:

MLObj	is a metalog object. You can create a metalog object by using the ML_CreateFromCDF function, the ML_CreateFromCoef function, or the ML_CreateFromData function.

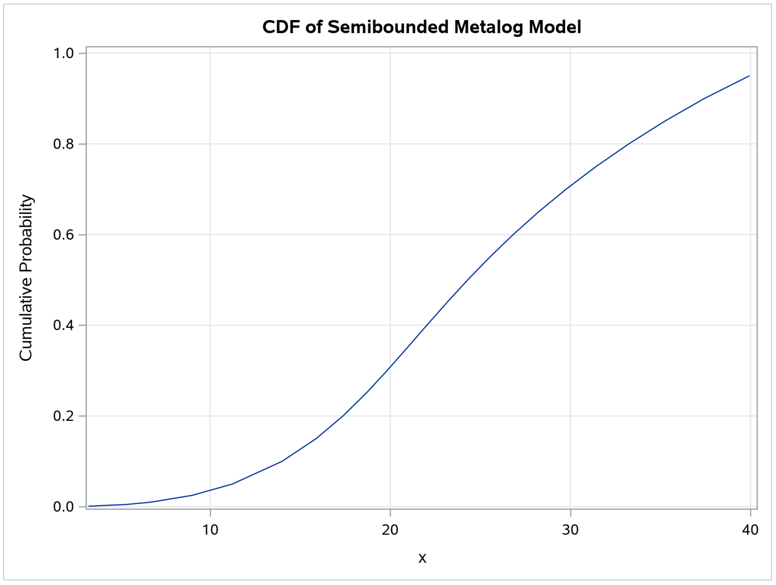
The subroutine supports one optional input argument:

p	is a vector of cumulative probabilities. If you omit this option, the graph is created by using percentiles and a few extreme probabilities.

The following statements fit a five-term bounded metalog model to nine data values. The call to the ML_CDF subroutine creates the graph shown in Figure 7. The CDF is shown for a custom set of cumulative probability values.

```
x   = {14, 17.3, 20, 22, 24, 27, 30, 33, 37.5};
bounds = {0 .};
MLObj = ML_CreateFromData(x, 5, bounds);
p = {0.001 0.005 0.01 0.025} || do(0.05, 0.95, 0.05);
title "CDF of Semibounded Metalog Model";
run ML_PlotCDF(MLObj, p);
```

Figure 7 CDF of a Semibounded Metalog Model



CALL ML_PlotECDF

ML_PlotECDF(*MLObj* < , *p*> **);**

The ML_PlotECDF subroutine creates a graph that overlays a scatter plot of the empirical distribution and the CDF of the model.

The subroutine requires one input argument:

MLObj	is a metalog object. You can create a metalog object by using the ML_CreateFromCDF function, the ML_CreateFromCoef function, or the ML_CreateFromData function.

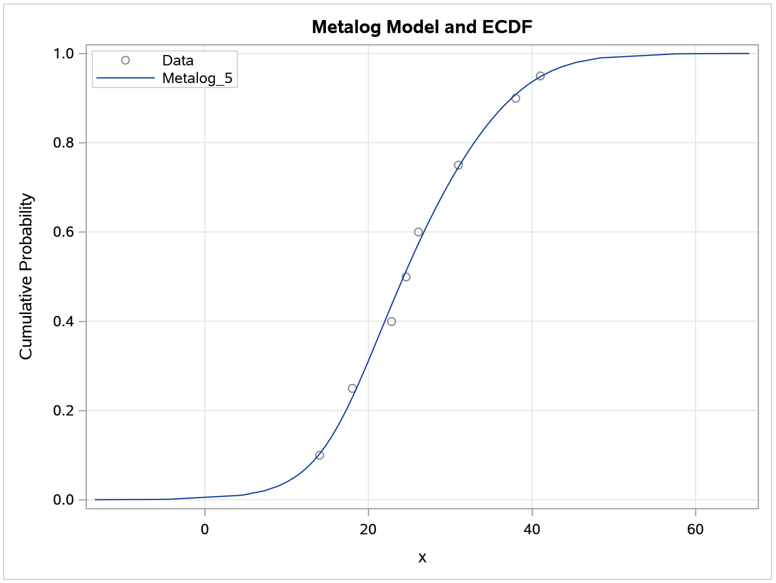
The subroutine supports one optional input argument:

p	is a vector of cumulative probabilities. If you omit this option, the graph is created by using percentiles and a few extreme probabilities.

The following statements fit a five-term unbounded metalog model to eight data points. The call to the ML_ECDF subroutine creates the graph shown in Figure 8. The graph overlays the ECDF of the data and the CDF of the model.

```
x   = { 14,   18, 22.8, 24.6, 26.1,   31,   38,   41};
cdf = {0.1, 0.25,  0.4,  0.5,  0.6, 0.75, 0.9, 0.95};
MLObj = ML_CreateFromCDF(x, cdf); /* default: order=5, unbounded */
title "Metalog Model and ECDF";
run ML_PlotECDF(MLObj);
```

Figure 8 Empirical CDF of Data and a Metalog Model



CALL ML_PlotEQuantile

ML_PlotEQuantile(*MLObj* < , *p*> **);**

The ML_PlotEQuantile subroutine creates a graph that overlays a scatter plot of the empirical quantiles of the data and the quantile function of the model. This graph is similar to the graph that is produced by the ML_PlotECDF subroutine, except that the positions of the axes are switched. The subroutine has the same syntax as the ML_PlotECDF function.

CALL ML_PlotPDF

ML_PlotPDF(*MLObj* < , *p*> **);**

The ML_PlotPDF subroutine creates a series plot of the PDF of the model.

The subroutine requires one input argument:

MLObj	is a metalog object. You can create a metalog object by using the ML_CreateFromCDF function, the ML_CreateFromCoef function, or the ML_CreateFromData function.

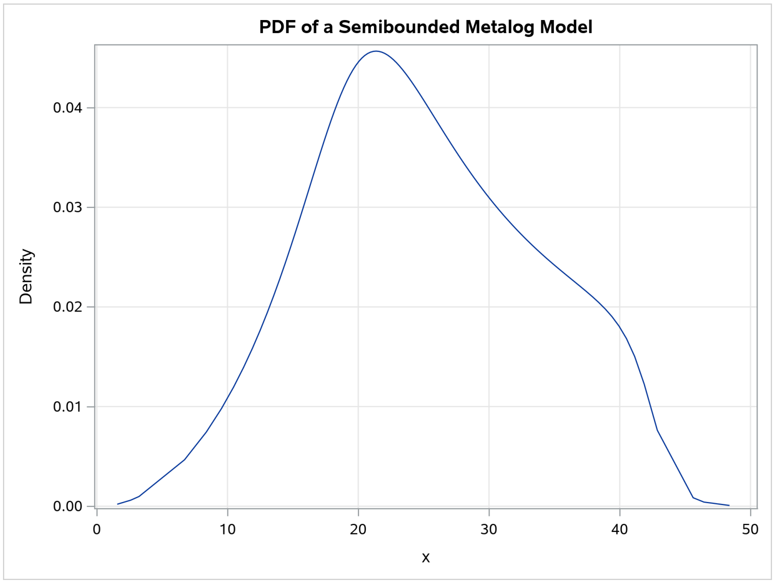
The subroutine supports one optional input argument:

p	is a vector of cumulative probabilities. If you omit this option, the graph is created by using percentiles and a few extreme probabilities.

The following statements fit a five-term bounded metalog model to nine data values. (The model is the same as the one shown in Figure 7.) The call to the ML_PDF subroutine creates the graph shown in Figure 9. The PDF is shown for the default set of cumulative probability values.

```
x    = {14, 17.3, 20, 22, 24, 27, 30, 33, 37.5};
bounds = {0 .};
MLObj = ML_CreateFromData(x, 5, bounds);
title "PDF of a Semibounded Metalog Model";
run ML_PlotPDF(MLObj);
```

Figure 9 PDF of a Semibounded Metalog Model



CALL ML_PlotQuantile

ML_PlotQuantile(*MLObj* < , *p*> **);**

The ML_PlotQuantile subroutine creates a series plot of the quantile function of the model. This graph is similar to the graph that is produced by the ML_PlotCDF subroutine, except that the positions of the axes are switched. The subroutine has the same syntax as the ML_PlotCDF function.

EXAMPLES OF USING THE METALOG FUNCTIONS

Symmetric-Percentile Triplets

Some fields rely on expert opinion to estimate the length of time required to complete a task. Specifically, an expert might provide estimates of a symmetric set of probabilities, such as the 10th, 50th, and 90th percentiles. (Another possible choice is the 10th, 25th, 50th, 75th, and 90th percentiles.) The expert might provide the following estimates:

- 10% of the time, we can complete this work in 7 days or less.
- 50% of the time, we can do it in 10 days or less.
- 90% of the time, we finish the work by the 14th day.

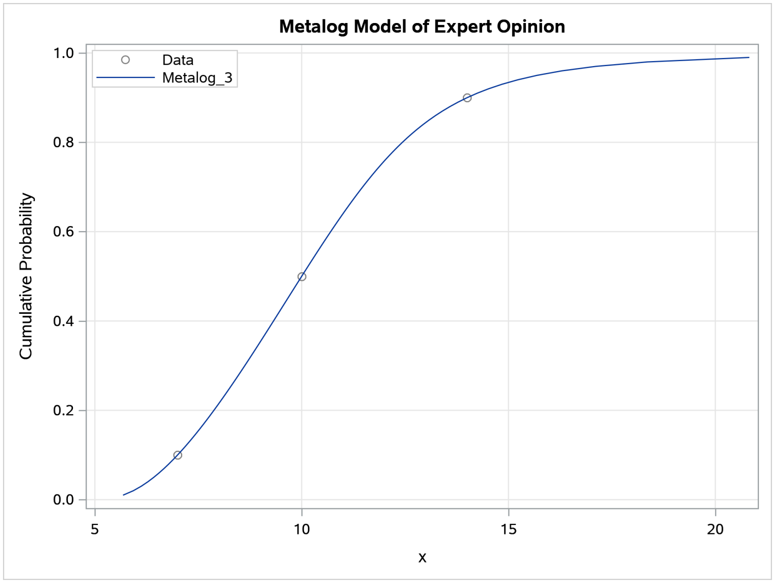
This is an example of a symmetric-percentile triplet (SPT). According to this expert, you can model the distribution of times by using a cumulative distribution that passes through (or near) the probability-time pairs (0.1, 7), (0.5, 10), and (0.9,14). Of course, infinitely many distributions could fit those three quantiles, but the following statements fit a three-term semibounded metalog distribution. To determine the lower bound for the distribution, you should go back to the expert and ask a second question: What is the minimum length of time required to complete the task? Assume that the expert replies that the task always requires at least five days. Then you can fit a metalog distribution on the interval $[5, \infty)$, as follows:

```
proc iml;
load module=_all_;

prob = {0.1, 0.5, 0.90};   /* cumulative probabilities */
x    = {7,    10,   14};   /* quantiles */
order = 3;                 /* three terms */
bounds = {5 .};            /* support on [5, infinity) */
```

```
SPT = ML_CreateFromCDF(x, prob, order, bounds );
title "Metalog Model of Expert Opinion";
p = do(0.01, 0.99, 0.01);
run ML_PlotECDF(SPT, p);
```

The metalog model is visualized in Figure 10. For these data, a three-term model fits the expert's estimates exactly.

Figure 10 CDF of a Semibounded Metalog Model



You can use the model to estimate completion times for other probability values. For example, the following statements estimate quantiles for the 5th, 25th, 75th, and 90th percentiles. The results are shown in Figure 11.

```
/* estimates for other percentiles */
p = {0.05, 0.25, 0.75, 0.95};
estTime = ML_Quantile(SPT, p);
print p[F=PERCENT9.] estTime[F=6.2];
```

Figure 11 Quantiles for a Set of Cumulative Probabilities

p	estTime
5%	6.42
25%	8.26
75%	11.92
95%	15.69

According to the expert's model, there is only a 5% chance that the task will be finished in 6.42 days. But the expert claims that 95% of the time, the task will be completed within 15.69 days.

A Model of the Average Expert Opinion

This example is a continuation of the example in the previous section.

One of the properties of the metalog distribution is that you can use it to construct a model that is the average of several expert opinions. Suppose you ask four experts to estimate the time that a task requires with 10%, 50%, and 90% confidence. Suppose you get the following responses:

- Expert 1 thinks the quantiles are 7, 10, and 14 days.
- Expert 2 thinks the quantiles are 8, 11, and 15 days.
- Expert 3 thinks the quantiles are 8, 12, and 18 days.
- Expert 4 thinks the quantiles are 7, 12, and 21 days.

You can use a metalog distribution to model the average opinion. There are two ways to form the "average model." One way is to form four symmetric-percentile triplet (SPT) models, as described in the previous section, and average the models. An equivalent, but simpler, way is to incorporate all of the experts' estimates in a single model.

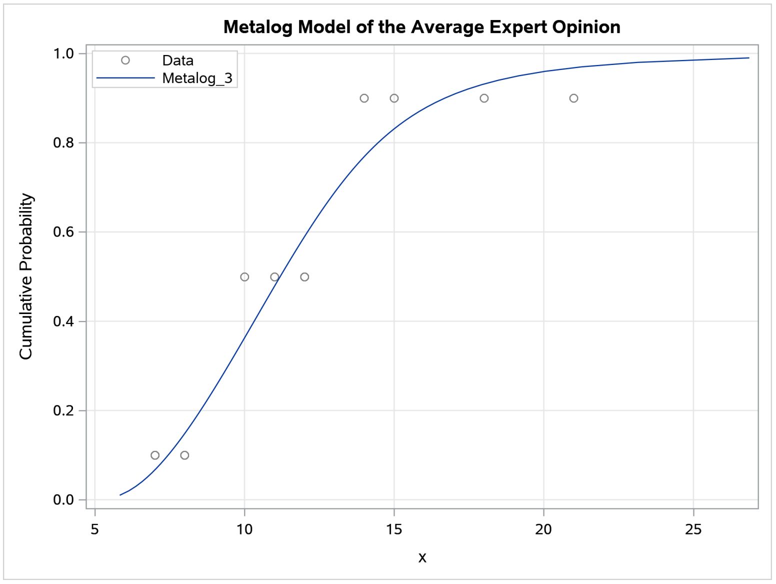
The following statements define the estimates from the experts. To fit the estimates, convert the data from wide format to long format as follows:

- Use the REPEAT function to duplicate the vector of probabilities four times.
- Use the COLVEC function to reshape the matrix of estimates into a column vector.

You can then use the ML_CreateFromCDF function to fit a bounded metalog model to the data. Figure 12 shows the model overlaid on a scatter plot of the experts' estimates.

```
proc iml;
load module=_all_;
order = 3;                 /* three terms */
bounds = {5 .};            /* support on [5, infinity) */

/* average the opinions of four experts */
prob = {0.1, 0.5, 0.90};   /* cumulative probabilities */
x    = {7   10   14,       /* quantiles from expert 1 */
        8   11   15,       /* expert 2 */
        8   12   18,       /* expert 3 */
        7   12   21};      /* expert 4 */

cdf = repeat(prob,4);      /* duplicate these values */
q = colvec(x);             /* convert from wide to long */
ML = ML_CreateFromCDF(q, cdf, order, bounds);
p = do(0.01, 0.99, 0.01);
title "Metalog Model of the Average Expert Opinion";
run ML_PlotECDF(ML, p);
```

Figure 12 Metalog Model of the Average Expert Opinion



Model Data

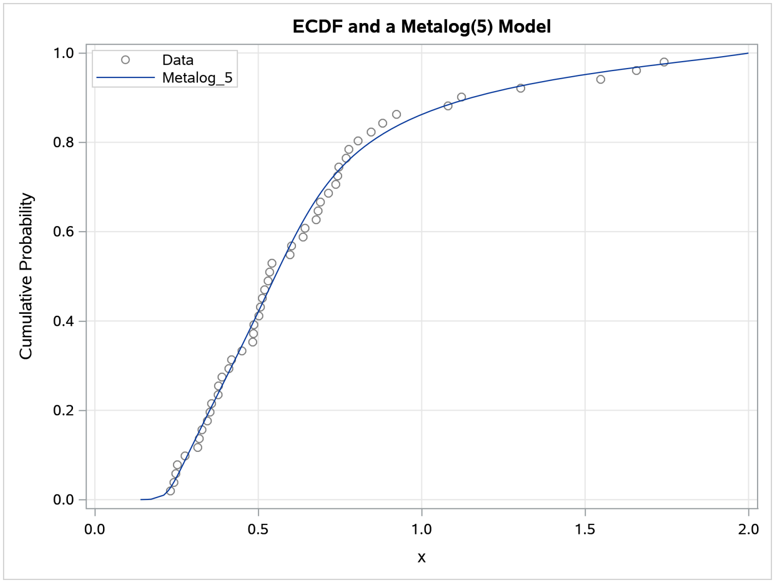
You can use the metalog distribution to model data for which the data-generating process is not known or cannot be modeled by a common “named” distribution, such as the lognormal, gamma, or Weibull distribution. For example, the documentation for the UNIVARIATE procedure includes an example that fits those three semibounded distributions to the width of gaps between two welded plates. In the example, two of the distributions (lognormal and gamma) are good models for the data, whereas the data do not support a Weibull model.

The following DATA step creates the data from the PROC UNIVARIATE example and fits a five-term bounded metalog distribution to the data. Because the measurements are lengths, the data are clearly bounded below by zero. You could decide to model the gaps by using a semibounded metalog distribution, but the following model assumes that the distribution is supported on the interval $[0, 2]$. This could be true if the manufacturer discards plates whose gaps are larger than 2 centimeters.

```
/* Example from PROC UNIVARIATE documentation: Gaps in 50 welded plates */
data Plates;
   label Gap = 'Plate Gap in cm';
   input Gap @@;
datalines;
0.746  0.357  0.376  0.327  0.485 1.741  0.241  0.777  0.768  0.409
0.252  0.512  0.534  1.656  0.742 0.378  0.714  1.121  0.597  0.231
0.541  0.805  0.682  0.418  0.506 0.501  0.247  0.922  0.880  0.344
0.519  1.302  0.275  0.601  0.388 0.450  0.845  0.319  0.486  0.529
1.547  0.690  0.676  0.314  0.736 0.643  0.483  0.352  0.636  1.080
;

proc iml;
load module=_all_;
/* read numerical data */
use Plates;  read all var "Gap" into X; close;

/* create a BOUNDED metalog object */
order = 5; bounds = {0 2};
SBL = ML_CreateFromData(X, order, bounds);

title "ECDF and a Metalog(5) Model";
run ML_PlotECDF(SBL);
title "PDF of a Metalog(5) Model";
run ML_PlotPDF(SBL, {0.0001 0.001}||do(0.01, 0.99, 0.01));
```

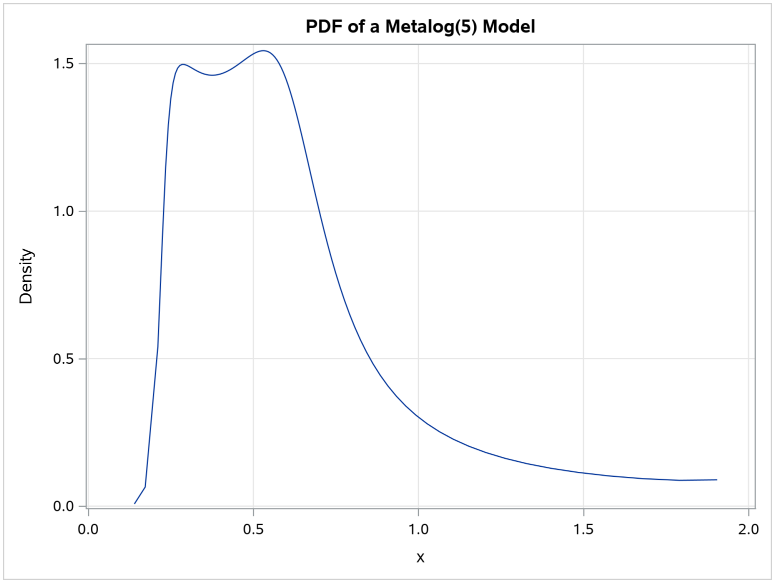
The graph in Figure 13 shows the cumulative distribution of the metalog distribution and the empirical distribution of the data. The graph shows that the cumulative distribution reaches the value 1 at x=2.

Figure 13 Empirical CDF of Data and a Metalog Model



The graph in Figure 14 shows the density of the metalog distribution. The density of the distribution is highest in the interval $[0.25, 0.6]$, which contains about 60% of the data.

Figure 14 PDF of a Bounded Metalog Model



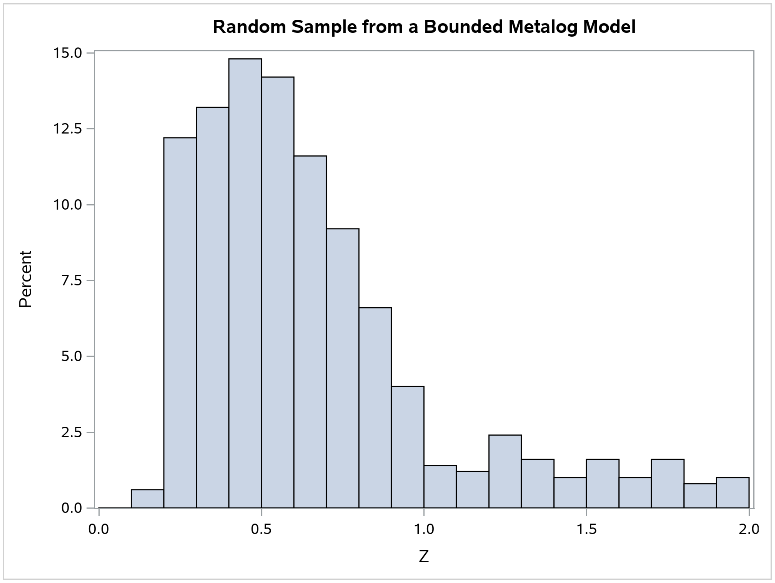
Simulation from a Metalog Model

An advantage of the metalog model is that it has an explicit quantile function. This makes it easy and fast to simulate from a metalog distribution.

The inverse CDF method is implemented in the ML_Rand function. The following program models the data from the previous section. The data are the width of the gap between two welded plates. The call to the ML_Rand function simulates 500 values from the model. The histogram of the simulated data is shown in Figure 15.

```
proc iml;
load module=_all_;
/* read numerical data */
use Plates;  read all var "Gap" into X; close;

/* create a BOUNDED metalog object */
order = 5; bounds = {0 2};
SBL = ML_CreateFromData(X, order, bounds);

call randseed(12345);
Z = ML_Rand(SBL, 500);
title "Random Sample from a Bounded Metalog Model";
call histogram(Z) rebin={0.05 0.1};
```

Figure 15 Random Sample from a Bounded Metalog Model



APPENDIX A: DOWNLOAD THE METALOG FUNCTIONS IN SAS

You can run the functions for the metalog distribution in the following SAS releases:

- PROC IML in SAS 9.4. This requires a license for SAS/IML.
- PROC IML in SAS Viya. This requires a license for SAS IML.
- The **iml** action in SAS Viya. This requires a license for SAS IML. You can call the **iml** action from many programming languages, including SAS, R, and Python.

To download the metalog package, you can clone the GitHub repository at https://github.com/sassoftware/sas-iml-packages/. There are several ways to do this. In SAS, you can use the GITFN_CLONE function to clone the repository to a local directory or to a libref. (In SAS Viya, the GITFN_CLONE function has been renamed; it is now the GIT_CLONE function.) For example, the following DATA step clones the repository to the WORK libref:

```
%let gitURL = https://github.com/sassoftware/sas-iml-packages/;
options dlcreatedir;
%let repoPath = %sysfunc(getoption(WORK))/sas-iml-packages;

/* clone repository; if repository exists, skip download */
data _null_;
if fileexist("&repoPath.") then
      put 'Repository already exists; skipping the clone operation';
else do;
      put 'Cloning repository sas-iml-packages';
      rc = gitfn_clone("&gitURL", "&repoPath." );
end;
run;
```

If you clone the repository to the WORK libref, it will disappear at the end of the SAS session. You can clone the repository to a permanent libref if you want it to persist. For example, you can clone the repository to a local directory by modifying the definition of the REPOPATH macro, as follows:

```
%let repoPath = C:/sas-iml-packages;   /* clone repository to this directory */
```

If you clone the repository to a permanent location, you should reclone (or fetch) the repository when you want a new copy.

After you clone the repository, you can use the Metalog package by including the function definitions in your program. There are two files that you can include:

- ML_proc.sas: Include this file *only* if you are using PROC IML.
- ML_define.sas: Always include this file.

For example, in PROC IML, you can use the %INCLUDE macro statement to include both files, as follows:

```
proc iml;
%include "&repoPath./Metalog/ML_proc.sas";
%include "&repoPath./Metalog/ML_define.sas";
x = {14, 18, 22, 24, 26, 31, 32, 38};
MLObj = ML_CreateFromData(x);
run ML_Summary(MLObj);
quit;
```

Each file ends with a STORE statement that saves the functions to the active storage library. Therefore, if you want to access the functions later in the same SAS session, you can use the LOAD statement, as follows:

```
proc iml;
load module= _all_;  /* assumes the files were read in this session */
x = {14, 18, 22, 24, 26, 31, 32, 38};
MLObj = ML_CreateFromData(x);
run ML_Summary(MLObj);
```

The following statements show how to use the Metalog package in the **iml** action from the SAS language. It assumes that you have downloaded the repository and that the REPOPATH macro points to the location of the repository.

```
proc cas;
action loadactionset / actionset='iml';
source pgm;
   %include "&repoPath./Metalog/ML_define.sas";
   x = {14, 18, 22, 24, 26, 31, 32, 38};
   MLObj = ML_CreateFromData(x);
   run ML_Summary(MLObj);
endsource;
iml / code = pgm;
quit;
```

APPENDIX B: DOWNLOAD THE METALOG FUNCTIONS IN PYTHON

An alternative to cloning the GitHub repository is to download the files directly from the internet. The following Python program converts the URL for the Metalog package into a "raw" URL, then reads the complete contents of the file into a string. The string is appended to a program that calls a metalog function. The program is sent to the **iml** action on a CAS server. You can use a similar technique in other programming languages.

```
import requests
import swat
# GitHub URL for file
url="https://github.com/sassoftware/sas-iml-packages/blob/master/Metalog/ML_define.sas"

# read the file directly from GitHub
raw_url = url.replace("github.com", "raw.githubusercontent.com").replace("/blob/", "/")
define_funcs = requests.get(raw_url).content.decode()

s = swat.CAS('myhost', 12345)  # server='myhost'; port=12345
                               # assumes login and password in ~/,authinfo
s.loadactionset('iml')
# concatenate definitions and SAS IML programming statements
pgm = define_funcs + """
      x = {14, 18, 22, 24, 26, 31, 32, 38};
      MLObj = ML_CreateFromData(x);
      run ML_Summary(MLObj);
      """
# submit the program; retrieve any results
c = s.iml(code=pgm)
print(c)
```

REFERENCES

Fleishman, A. (1978). "A Method for Simulating Non-normal Distributions." *Psychometrika* 43:521–532.

Headrick, T. C. (2002). "Fast Fifth-Order Polynomial Transforms for Generating Univariate and Multivariate Nonnormal Distributions." *Computational Statistics and Data Analysis* 40:685–711.

Headrick, T. C. (2010). *Statistical Simulation: Power Method Polynomials and Other Transformations*. Boca Raton, FL: Chapman & Hall/CRC.

Johnson, N. L., Kotz, S., and Balakrishnan, N. (1994). *Continuous Univariate Distributions.* 2nd ed. Vol. 1. New York: John Wiley & Sons.

Keelin, T. (2016a). "The Metalog Distributions." *Decision Analysis* 13:243–277. https://doi.org/10.1287/deca.2016.0338.

Keelin, T. (2016b). "The Metalog Distributions." http://www.metalogdistributions.com.

ACKNOWLEDGMENTS

Thanks to Kelly Miller and Jack Berry for testing the metalog functions. Thanks to Arash Dehghan Banadaki and Dave Kessler for improving the documentation.