Paper SAS5245-2016

Custom Risk Metrics with SAS® High-Performance Risk

Katherine Taylor and Steven Miles, SAS Institute Inc.

ABSTRACT

There are standard risk metrics financial institutions use to assess the risk of a portfolio. These include well-known measures like value at risk and expected shortfall and many more. While there are industry-standard approaches for calculating these measures, it is often the case that financial institutions have their own methodologies. Further, financial institutions sometimes write their own measures. SAS® High-Performance Risk comes equipped with over twenty risk measures that use standard methodology, but the product also allows customers to define their own risk measures. This paper leads the user through the creation of custom risk metrics using the HPRISK procedure.

INTRODUCTION

What are risk metrics? The starting point of risk management is risk measurement. Risk metrics capture the expectation that a financial portfolio will experience loss. A risk measure can tell us the average expected loss, or the anticipated frequency of a loss. It can tell us how sure we are that our losses will be less than a certain amount. We select and define risk metrics based on the type of risk to manage and our risk preferences.

Common risk metrics have common definitions; value at risk (VaR) is defined the same in most textbooks. Other risk metrics can be completely custom to an organization. However, even when computing common risk metrics like VaR, analysts often use different methods in the details. SAS® High-Performance Risk comes equipped with over twenty risk measures that use standard methodology, but it also provides the option to modify these common risk measures. In addition, the product provides full support for customers to define their own risk measures.

COMPUTING RISK METRICS IN A SIMULATION

Because we do not know how much money our portfolio will lose in the future, we must guess how much it could lose. We can either make assumptions about losses, modeling them parametrically, or we can simulate losses. Simulation is the more common approach when leveraging high-performance technology. In simulation, we compute the value of a portfolio using many different sets of theoretical market conditions. The result is a vector with each value representing a different portfolio loss. From this vector of losses, we can calculate risk metrics. For example, an average of these values would be the expected loss and an average of the worst losses would be the expected shortfall.

In the case of portfolio simulation, risk measures are computed on the resulting set of simulated losses. There are two ways to think about risk measure definition. One is as a function directly applied to the vector of simulated losses. Another way to think about it is as a function of the distribution of losses. HPRISK supports both perspectives and we cover each in this paper. We start with the first, which uses a feature called user-defined statistics. Then we come to the second, which uses a feature called distortion risk measures.

USER-DEFINED STASTISTICS EXAMPLE 1: WEIGHTED VAR

Given a vector of losses, I can write any function to capture the risk characteristics of my portfolio. I might want to write something custom, like the average of the ten worst losses. Or perhaps I want to use a common metric such as VaR, but I want to apply a more conservative weighting around the quantile.

The syntax for registering your own risk metric in HPRISK is simple. All that is needed is to register the metric in the HPEXPORT procedure using the USERSTAT statement. You specify a function and the distribution type, which is usually set to standard:

```
USERSTAT my_metric_name FUNC=my_func DIST_TYPE=standard;
```

The function associated with the user-defined statistic contains the logic to compute the risk metric and is created using either the FCMP or the COMPILE procedure. The first argument to this function is the vector of simulated losses. This argument is required:

```
function my_func (losses[*]);
```

Here is an example of a function for a user-defined statistic in which I define my own VaR measure. I write logic to do a weighted average of the values around the confidence level. The call to the state_info routine to fetch alpha allows me to compute my measure at the user's current confidence level (this example, and the others that follow, are for value type output variables. You need additional logic to account for PL type output variables.):

```
/* Function that weights around the five nearest points */
function weighted_var(OutvarDistribution[*]);
   array weights[5] / nosym (0.1 0.2 0.4 0.2 0.1);

   /* Get alpha for VaR from the environment */
   call state_info('alpha', varAlpha);

   /* Get index of VaR */
   n_percentile = round(dim(OutvarDistribution) * varAlpha);

   /* Compute weighted VaR as a weighted sum of the points around VaR */
   i = 0;
   returnval = 0;
   do n = n_percentile-2 to n_percentile+2;
      i = i + 1;
      weightedVar = sum(weightedVar, weights[i] * OutvarDistribution[n]);
   end;
   return (weightedVar);
endfunc;
```

The following line is used to register the weighted_var function on HPEXPORT:

```
userstat WeightedVAR func=weighted_var dist_type=standard;
```

This statistic function will compute and return the custom statistic for every subportfolio requested in the query, for every horizon, and for every output variable.

For more complex custom metrics, you can use some additional optional arguments in your function. Additional arguments include the base case value, the current output variable name, and the current horizon number:

```
function my_func (losses[*], BasecaseValue, OutvarName $, HorizonNumber);
```

You can also write statistics on distributions other than the standard distribution by specifying the distribution type in the USERSTAT statement. Other options include the WITHOUT distribution, which is the distribution of the aggregate portfolio's loss if the current subportfolio is removed. You can also select the RAW distribution to include missing values.

USER-DEFINED STASTISTICS EXAMPLE 2: CONTRIBUTION VAR

Another useful risk metric is contribution value at risk. This measure calculates the contribution of the current subportfolio to the value at risk of the aggregate portfolio. VaR itself is not summable, but the contribution VaR measure is. The SAS® risk engine provides this measure out of the box, but there are several methods for computing it (Hallerbach 2003, pp. 1-18).

In this example, we compute contribution value at risk using two different methods, a state matching approach (or nearest neighbor) and a linear regression approach.

These functions use the get_ranked_values_top function to retrieve the vector of losses for the aggregate portfolio:

```
/* Function to find the nearest neighbor contribution VaR */
function cvar_nearest_neighbor(OutvarDistribution[*]);
   array topPortValues[1] / nosym;
   array index[1] / nosym;

   call state_info('alpha', varAlpha);

   call get_ranked_values_top(., ., topPortValues, index);

   /* Compute the nearest neighbor CVar */
   var_index = round(dim(OutvarDistribution) * varAlpha);
   cVar = OutvarDistribution[index[var_index]];
   return(cVar);
endfunc;
```

The next example again calculates contribution VaR, but uses a linear regression approach:

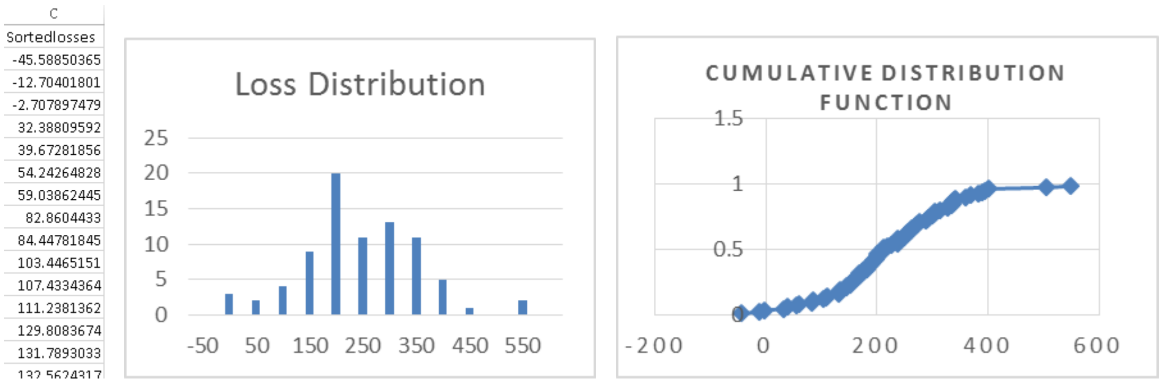
```
/* Function using ordinary least squares to approximate contribution VaR */
function cvar_ordinary_least_squares(OutvarDistribution[*]);
   array topPortValues[1] / nosym;
   array index[1] / nosym;
   call state_info('alpha', varAlpha);
   call get_ranked_values_top(., ., topPortValues, index);

   /* Compute OLS coefficients alpha and beta */
   start_index = round(dim(OutvarDistribution) * varAlpha / 2);
   end_index =round(dim(OutvarDistribution) * varAlpha * 1.5);
   call ordinaryLeastSquares(topPortValues, OutvarDistribution, index,
                             start_index, end_index, olsAlpha, olsBeta);

   /* Compute top portfolio VaR */
   n_percentile = dim(topPortValues) * varAlpha;
   n_up = ceil(n_percentile);
   n_down = floor(n_percentile);
   if n_down eq n_up then do;
      topVar = topPortValues[n_down];
   end;
   else do;
      topVar = topPortValues[n_down] + (topPortValues[n_up]-
               topPortValues[n_down])* (n_percentile-n_down)/(n_up-n_down);
   end;

   /* Compute C-VaR */
   cVar = topVar * olsBeta + olsAlpha;
   return(cVar);
endfunc;
```

The following line is used to register the CVaR_OLS and CVaR_Near functions on HPEXPORT:

```
userstat CVaR_OLS func=cvar_ordinary_least_squares dist_type=standard;
userstat CVaR_Near func=cvar_nearest_neighbor dist_type=standard;
```

DISTORTION MEASURE EXAMPLE: VAR

The preceding examples use the user-defined statistics feature to write custom risk metrics. There is another way to compute custom statistics in HPRISK; you can use distortion risk measures. The difference between user-defined statistics and distortion risk measures is the vector of values on which we apply our function. Distortion risk measures are calculated using the vector of cumulative distribution function (CDF) values, whereas user-defined statistics are computed on the vector of losses directly. The picture below illustrates the relationship between the vector of losses and the corresponding CDF values.



C
Sortedlosses
-45.58850365
-12.70401801
-2.707897479
32.38809592
39.67281856
54.24264828
59.03862445
82.8604433
84.44781845
103.4465151
107.4334364
111.2381362
129.8083674
131.7893033
132.5624317

Figure 1. Loss Distribution and Cumulative Distribution

From the distortion risk measure perspective, we think of risk metrics as different reweightings, or distortions, of the loss distribution. For example, when every loss receives the same weight, the metric is the average or expected loss. When the loss at a given percentile gets all of the weight and all other losses get no weight, the metric is value at risk. When losses above a given percentile get equal weight and losses below the given percentile get no weight, the metric is expected shortfall.

The decision to use distortion risk measures instead of user-defined statistics might be personal preference or it might be that the measure you want to implement has been defined that way at your company or in academic literature.

The syntax for distortion risk measures requires that you create your own distortion function in either the FCMP or COMPILE procedure, and associate it with the risk metric via the DISTORTION statement in the RISK procedure. You can specify the distribution type like you did in the user-defined statistic using the DISTRIBUTION option, and you can provide additional parameters to your function using the PARM option:

```
DISTORTION my_measure
   FUNC= dist_func
   DISTRIBUTION=dist_type
   PARM= parm_value|parm_list;
```

The distortion function minimally accepts a single argument that represents the CDF of a distribution at the current simulated state. You can specify additional parameters as you like. To clarify, we say that distortion risk measures use the vector of CDF values, and the function processes one value at a time, so the input to the distortion risk measure function is a single value:

```
function my_dist_func (x, <parm>);
```

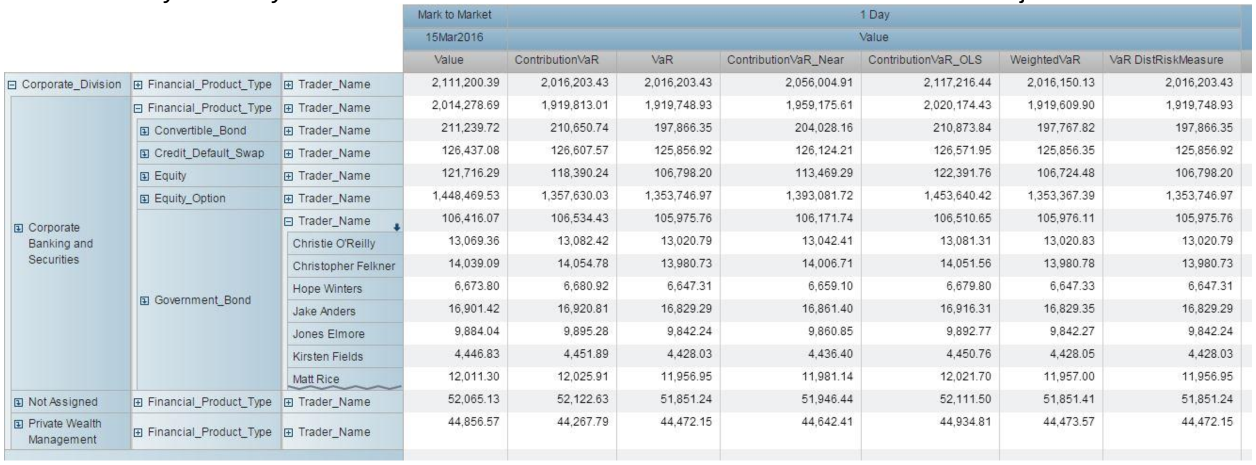
Here is an example of VaR defined as a distortion risk measure. The distortion function returns a value from the newly weighted CDF. This is translated back to the loss distribution to determine the value reported as the risk metric VaRDistMeas:

```
function VaRDistMeas(x,varAlpha);
   if (x le (1-varAlpha)) then return(0);
   else return(1);
endfunc;
```

ANALYZING CUSTOM METRICS IN SAS

Once you have registered your risk metric functions with the SAS risk engine, they are treated the same as the built-in risk metrics. You will see them when you query the results of a simulation, both in batch queries and through the SAS High-Performance Risk Explorer. As with all risk metric computations in the SAS High-Performance Risk environment, custom risk metrics are computed on demand as the user requests them.

Here is a view of the SAS High-Performance Risk Explorer displaying the custom risk metrics to the end user. You may hide any built-in risk measures to restrict the set of available metrics to just the desired set.



			Mark to Market	1 Day						
			15Mar2016	Value						
			Value	ContributionVaR	VaR	ContributionVaR_Near	ContributionVaR_OLS	WeightedVaR	VaR DistRiskMeasure
⊟ Corporate_Division	⊞ Financial_Product_Type	⊞ Trader_Name	2,111,200.39	2,016,203.43	2,016,203.43	2,056,004.91	2,117,216.44	2,016,150.13	2,016,203.43
⊞ Corporate Banking and Securities	⊟ Financial_Product_Type	⊞ Trader_Name	2,014,278.69	1,919,813.01	1,919,748.93	1,959,175.61	2,020,174.43	1,919,609.90	1,919,748.93
	⊞ Convertible_Bond	⊞ Trader_Name	211,239.72	210,650.74	197,866.35	204,028.16	210,873.84	197,767.82	197,866.35
	⊞ Credit_Default_Swap	⊞ Trader_Name	126,437.08	126,607.57	125,856.92	126,124.21	126,571.95	125,856.35	125,856.92
	⊞ Equity	⊞ Trader_Name	121,716.29	118,390.24	106,798.20	113,469.29	122,391.76	106,724.48	106,798.20
	⊞ Equity_Option	⊞ Trader_Name	1,448,469.53	1,357,630.03	1,353,746.97	1,393,081.72	1,453,640.42	1,353,367.39	1,353,746.97
	⊞ Government_Bond	⊟ Trader_Name	106,416.07	106,534.43	105,975.76	106,171.74	106,510.65	105,976.11	105,975.76
		Christie O'Reilly	13,069.36	13,082.42	13,020.79	13,042.41	13,081.31	13,020.83	13,020.79
		Christopher Felkner	14,039.09	14,054.78	13,980.73	14,006.71	14,051.56	13,980.78	13,980.73
		Hope Winters	6,673.80	6,680.92	6,647.31	6,659.10	6,679.80	6,647.33	6,647.31
		Jake Anders	16,901.42	16,920.81	16,829.29	16,861.40	16,916.31	16,829.35	16,829.29
		Jones Elmore	9,884.04	9,895.28	9,842.24	9,860.85	9,892.77	9,842.27	9,842.24
		Kirsten Fields	4,446.83	4,451.89	4,428.03	4,436.40	4,450.76	4,428.05	4,428.03
		Matt Rice	12,011.30	12,025.91	11,956.95	11,981.14	12,021.70	11,957.00	11,956.95
⊞ Not Assigned	⊞ Financial_Product_Type	⊞ Trader_Name	52,065.13	52,122.63	51,851.24	51,946.44	52,111.50	51,851.41	51,851.24
⊞ Private Wealth Management	⊞ Financial_Product_Type	⊞ Trader_Name	44,856.57	44,267.79	44,472.15	44,642.41	44,934.81	44,473.57	44,472.15

CONCLUSION

Organizations have an increasing demand for non-traditional risk statistics in high-performance computing environments. Some organizations might want to design their own risk statistics to better summarize their portfolio's risk of financial loss. SAS High-Performance Risk offers two ways an analyst can compute custom statistics—user-defined statistics and distortion measures—delivering a range of flexibility in computing custom risk measures in a simulation setting.

REFERENCES

Hallerback, W. 2003. "Decomposing Portfolio Value-at-Risk: A General Analysis." *Journal of Risk*, 5:1-18.

RECOMMENDED READING

- *SAS® Risk Dimensions®: Procedures Guide*
- *SAS® High-Performance Risk: Procedures Guide*

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Katherine Taylor
SAS Institute
Katherine.Taylor@sas.com

Steven Miles
SAS Institute
Steven.Miles@sas.com