Paper SAS2114-2018

Economic Capital Modeling with SAS® Econometrics

Mahesh V. Joshi, SAS Institute Inc.

ABSTRACT

A statistical approach to developing an economic capital model requires estimation of the probability distribution model of the aggregate loss that an organization expects to see in a particular time period. A well-developed economic capital model not only helps your enterprise comply with industry regulations but also helps it assess and manage risks. A loss distribution approach decomposes the aggregate loss of each line of business into frequency and severity of individual loss events, builds separate distribution models for frequency and severity, and then combines the two distribution models to estimate the distribution of the aggregate loss for each business line. The final step estimates a copula-based model of dependencies among the business lines and combines simulations from the dependency model with the aggregate loss models of individual business lines. This process yields an empirical estimate of the enterprise-wide loss distribution, which helps you develop an economic capital model. The process is characterized by both big data and big computation problems. This paper describes how SAS® Econometrics software can help you take advantage of the distributed computing environment of SAS® Viya® to implement each step of the process efficiently.

INTRODUCTION

A financial enterprise that incurs losses, either because of the nature of its business or from adverse events, needs to estimate the extent of those losses across multiple lines of business (LoBs) or risk categories. Developing a statistical model for the enterprise-wide losses is an integral part of the overall framework of enterprise risk management. The model not only helps calculate a reasonably accurate estimate of the minimum capital that an enterprise must set aside to cover the worst-case losses and to satisfy regulatory requirements such as Basel III (banking industry) or Solvency II (insurance industry), but it also helps assess how much economic capital is available to improve the business. Further, the model-based framework makes it easy to estimate the capital requirements for the various socioeconomic scenarios that an enterprise operates in.

For the purposes of this paper, a probability distribution model of the total loss that an enterprise can incur because of the losses across its multiple LoBs or risk categories is referred to as an economic capital model (ECM). The following proposed six-step process of estimating an ECM takes into account the probabilistic nature of the losses. It is particularly useful because it provides a recipe for computing both the location and dispersion estimates of any risk measure, such as the value at risk (VaR), for the distribution of the total loss.

1. Collect the loss event data in all LoBs that incur financial losses. The loss event data consist of frequency (count) of losses that an LoB incurs in a particular time period and the severity (magnitude) of each loss.

2. Estimate separate loss distribution models of the frequency and severity for each business line. The frequency model is a parametric model that consists of a discrete probability distribution, and the severity model is a parametric model that consists of a continuous probability distribution. Each model can contain regression parameters that measure the impact of external factors on the location or scale of the probability distribution.

3. Create a compound distribution model (CDM) of the aggregate loss that an LoB can incur in the particular time period. This model combines the frequency and severity models. Because frequency and severity models can be complex, you often cannot write the CDM in a closed, parametric form. Hence, it is usually estimated by simulating a large empirical sample of the aggregate loss.

   At this step in the process, you can estimate the worst-case losses for an individual LoB by computing a metric such as VaR or tail value at risk (TVaR) from the large empirical sample of the CDM.

   In the context of ECM development, the distribution of aggregate loss for each LoB is referred to as a marginal distribution.

4. Estimate a dependency structure among the losses of all LoBs. This dependency structure essentially estimates how a loss in one LoB correlates with the losses in other LoBs. A typical method of estimating the dependency is to fit a copula model to the aggregate losses of all LoBs, where losses are aggregated from the historical loss event data and matched by the time period of interest.

5. Use the estimated dependency structure to simulate a large sample of uniform probabilities of losses in all LoBs. Each observation in the sample essentially records the cumulative probability of seeing a loss in each LoB in the time period of interest. In other words, each observation records one possible scenario where all LoBs simultaneously incur losses, and the extent of each LoB's loss is encoded in its probability, which accounts for the dependency of that LoB on other LoBs. Simulating a large number of such scenarios gives you a comprehensive picture of enterprise-wide loss probabilities. These simulation data need to be transformed from the probability scale to the loss scale, which takes us to the next and final step.

6. To transform the loss probabilities from the copula simulation table to losses, invert the probability of each LoB according to that LoB's marginal distribution. The inversion requires an estimate of the cumulative probability distribution (CDF) of the marginal distribution. Although the CDF is not available in the closed form, you can compute its empirical estimate, an empirical distribution function (EDF), by using the large CDM sample of that LoB. Inversion of EDF leads to a percentile estimate of that LoB's simulated loss probability, which is essentially the loss estimate for that LoB. Aggregating the losses across all LoBs gives you an estimate of the total loss. Repeating this process for all the observations in the copula simulation table results in a sample of the total loss, which essentially encodes the probability distribution model of the enterprise-wise losses. You can use this sample to compute empirical estimates of all risk metrics of your choice and to determine the economic capital needs of your enterprise.

SAS Econometrics software contains a number of procedures to help you implement each modeling and simulation step in this process. The CNTSELECT and SEVSELECT procedures help you estimate a wide range of frequency and severity models, respectively. The CCDM procedure combines the models that the CNTSELECT and SEVSELECT procedures fit in order to simulate a large CDM sample that represents the distribution of the aggregate loss in each LoB. The CCOPULA procedure simulates various types of copula models. Finally, you can use a SAS® DATA step to combine the simulation output of the CCOPULA procedure with the outputs of multiple PROC CCDM steps, one for each LoB, to prepare the sample that represents the distribution of the enterprise-wide loss.

The losses in each LoB often depend on the economic and social environments that the respective business operates in. It is important to estimate the effect of such external factors on the frequency and severity models. The dependence of frequency and severity models on the external factors in turn implies that the CDM and ECM also depend on those factors. Hence, it is not sufficient to estimate the ECM for just one set of values of the external factors, referred to as an external scenario. You should estimate the ECM for multiple external scenarios, each representing a different possible state of the world. Because the CDM and ECM modeling steps must be repeated for each external scenario, it is important for those steps to run as efficiently as possible. SAS Econometrics procedures are designed to help you achieve computational and modeling efficiency.

The frequency and severity models contain parameters that are estimated. Because the parameters themselves are random variables, there is uncertainty associated with them. The estimates of the CDM and ECM should ideally reflect that uncertainty. The CCDM procedure enables you to conduct a parameter perturbation analysis to estimate the variability in the summary statistics and percentiles of an individual LoB's aggregate loss distribution. The perturbation analysis also prepares multiple perturbed samples of the aggregate loss, where each sample corresponds to one set of parameters that are perturbed from their mean estimates by using the estimate of the covariance among parameters. In the last step of the ECM process, which combines the copula simulation table with the aggregate loss samples from multiple LoBs, you can use these perturbed samples to estimate the dispersion (variability) in the summary statistics and risk measures of the enterprise-wide loss.

The rest of this paper describes each step of the ECM process in detail and illustrates how you can implement that step by using a SAS Econometrics procedure or a tool from Base SAS® software. It also illustrates how you can use some other languages, such as Python, to implement many of the steps.

DATA PREPARATION

Collecting and preparing historical loss data for analytical modeling are a crucial prerequisite in any loss modeling solution. Depending on the definitions of the LoBs or risk categories that you want to analyze, the data granularity varies, but it is recommended that you collect and store the loss data at the level of the loss event. At a minimum,

you need to record the time of occurrence and severity of the loss event. The data are rarely available in a form that is readily usable for building loss models. You often need to prepare the data by transforming and combining various internal and external data sources. Depending on the source of the data, you might not be able to observe all loss events, or you might not know the exact severity of each loss event. Such limitations are known as truncation and censoring effects, respectively. Truncation and censoring occur mostly by design, and you usually know the provisions in the policy or the data collection process that produce these effects. For example, for an insurance policy, the deductible results in truncation and the maximum payment limit results in censoring. For each loss event, you should record the truncation thresholds and censoring range if you know that the loss event is generated by an entity that is subject to such provisions. The SEVSELECT procedure accounts for truncation and censoring effects and builds the model for *ground-up loss*.

For brevity, this paper uses the term line of business (LoB) rather broadly to refer to the actual business lines, risk event types, or risk categories that you might want to include in your economic capital model. The definition of an LoB depends on your industry and the specifics of your enterprise. For example, an insurance company might want to define an LoB according to different types of policies (perils), such as homeowner's policies, the physical damage part of automobile policies, the liability part of automobile policies, and workers' compensation policies. A bank might want to develop a separate ECM for different risk categories, such as credit risk, market risk, and operational risk. It might want to further divide the operational risk category into multiple event types, such as internal fraud, external fraud, and workplace safety, that the Basel Committee on Banking Supervision defines. A bank can also analyze the operational risk by its business lines, such as corporate finance, commercial banking, retail banking, and so on, according to the Basel Committee's definitions. Further yet, you can define an LoB as an intersection of business lines and event types.

More often than not, the losses depend on external factors, referred to in statistics as regressors. For such cases, you should augment the loss event data to record all the relevant characteristics of the entity that generates the loss event and the environment in which the entity operates. In the financial industry, regressors are known as key risk indicators (KRIs). It is a critical exercise to identify, define, and collect the data about KRIs for each LoB that you want to build loss models for. After the regressors are identified and data are collected for them, it is recommended that you properly code, scale, or standardize their values.

The data that you require for estimating the dependency structure among different LoBs can be derived from the data that you use for modeling the marginal distributions of the LoBs. However, you might also want to use different types of data, such as those available from state departments of insurance or data vendors.

The amount of data available for modeling varies by the level of maturity of the data collection process and the nature of losses. For example, data for operational risk events might not be as abundant as data for insurance claims. When the frequency and severity data are relatively small, you will have faster estimation times for frequency and severity models even with modest computational resources. However, the steps that estimate the CDM and ECM always use simulation and can benefit from as large a data sample as you can simulate. So your investment in a distributed computing platform can pay off in the form of more accurate CDM and ECM estimates. As your data collection process matures and loss data grow, the distributed computing platform can also serve as a distributed storage platform, and the tools that account for the distributed nature of data will offer significant computational advantages over tools that are limited by their design to do modeling on a single machine.

This paper illustrates the ECM process by using the following simple, simulated input data that are inspired by one form of operational risk modeling that the banking industry uses.

EXAMPLE OPERATIONAL RISK DATA

Consider that you are a bank, and as part of creating an economic capital model for your operational risk, you want to analyze losses in two LoBs, retail banking and commercial banking. For each LoB, you want to use key risk indicators (KRIs) that are appropriate to that LOB. Let **CorpKRI1**, **CorpKRI2**, **CbKRI1**, **CbKRI2**, **CbKRI3**, **RbKRI1**, **RbKRI2**, and **RbKRI3** be the available KRIs, where the prefixes **Corp**, **Cb**, and **Rb** indicate corporate-level KRIs, KRIs specific to commercial banking, and KRIs specific to retail banking, respectively. Some examples of corporate-level KRIs are the ratio of temporary to permanent employees and the number of security breaches that are reported during a year. Some examples of commercial banking KRIs are the number of credit defaults, the proportion of financed assets that are movable, and the penalty claims against your bank that can result from processing delays. Some examples of retail banking KRIs are the number of credit cards that are reported stolen, the fraction of employees who have not undergone fraud detection training, and the number of forged drafts and checks that are presented in a year. To mimic the real-world applications, let different yet overlapping sets of KRIs be used for the frequency and severity models. Let the analysis period be one month, and let the data preparation process result in the following data sets:

- The **Work.OpRiskLossCounts** data set contains the frequency regressors and the variable **NumLoss**, which records the number of losses that an LoB incurs in a month. The LoBs are identified by the BY variable **line**.
- The **Work.OpRiskLosses** data set contains the severity regressors and the variable **LossValue**, which records the severity of each loss. The LoBs are identified by the BY variable **line**.
- The **Work.OpRiskMatchedLosses** data set contains the aggregate losses of all LoBs, where the losses in each LoB are aggregated by month. These matched losses are useful for copula modeling.

When your frequency or severity models contain regressors, you must also create an external scenario data set. Each observation in an external scenario data set contains values that you expect to observe for each regressor in some future state of the world. The final CDM and ECM estimates depend on this scenario. For this example, the scenario data set is named **Work.MultiConditionScenario**, and it contains observations for 12 months of a future year.

The SAS code that generates all example data sets accompanies this paper on the conference proceedings website (http://support.sas.com/resources/papers/proceedings18/).

LOADING DATA INTO SAS® CLOUD ANALYTIC SERVICES (CAS)

SAS Econometrics procedures run on SAS Viya. One component of SAS Viya is SAS Cloud Analytic Services (CAS), which is the analytic server and associated cloud services. This paper assumes that you have a CAS server already available; contact your SAS Viya system administrator if you need help starting and terminating a server. To run the SAS Viya procedures, you must first create a CAS session and set up a CAS engine libref that you can use to connect to the CAS session. If your system administrator has not already created a CAS session and a CAS engine libref for your use, you can use the following statements to create them:

```
cas mysess;
libname mycas cas sessref=mysess;
```

The CAS statement creates the CAS session named **mysess**, and the LIBNAME statement creates the **mycas** CAS engine libref that you use to connect to this session.

To load a SAS data set into a data table in your CAS session, you can use the CAS engine libref in a DATA step as follows:

```
data mycas.CASTable;
   set SASDataSet;
run;
```

If you create your own CAS session and CAS engine libref, then after you finish your analysis, you can terminate the CAS session and clear the CAS engine libref that is associated with it by submitting the following CAS statement:

```
cas mysess terminate;
```

For more information about CAS, see the SAS Econometrics documentation that is available at http://support.sas.com/documentation/onlinedoc/econometrics/index.html.

FREQUENCY (COUNT) MODELING

The CNTSELECT procedure in SAS Econometrics estimates frequency models. The following key features of the procedure are relevant to the ECM process:

- It fits a count regression model, where the dependent variable is a count variable that, conditional on the regressors, has one of the following discrete probability distributions: Poisson, negative binomial with linear or quadratic variance, and Conway-Maxwell-Poisson (CMP). It can also fit a zero-inflated version of each distribution, enabling you to account for the large number of zeros often present in real-life loss count data. The regressors are assumed to affect different aspects of each distribution, such as the mean, dispersion, and probability of observing a zero count.
- It support a variety of interaction and classification regression effects. It can also perform automatic model (variable) selection by using methods such as forward, backward, or stepwise selection.

- It can store the fitted model in an item store, which is a CAS table that stores the model information and parameter estimates in a binary format that the CCDM procedure can later use.

The following statements show how to use the CNTSELECT procedure to fit Poisson and negative binomial distribution models to the example data:

```
/*** Fit Poisson count models ***/
proc cntselect data=mycas.OpRiskLossCounts;
   by line;
   model numloss=corpKRI1 corpKRI2 cbKRI1 cbKRI2 cbKRI3 cbKRI2*cbKRI3
         rbKRI1 rbKRI2/dist=poisson;
   selection;
run;
/*** Fit negative binomial count models ***/
proc cntselect data=mycas.OpRiskLossCounts;
   by line;
   model numloss=corpKRI1 corpKRI2 cbKRI1 cbKRI2 cbKRI3 cbKRI2*cbKRI3
         rbKRI1 rbKRI2/dist=negbin;
   selection;
run;
```

The MODEL statement specifies all the possible regression effects that you believe can affect the number of losses in all LoBs. The SELECTION statement performs model (variable) selection independently for each BY group. The default settings of the SELECTION statement choose the best subset of regression effect to maximize the Schwarz Bayesian criterion (SBC). The comparison of the fit summary of the final models is shown in Figure 1. As the results show, the negative binomial model is the best for the commercial banking LoB, because it has a smaller SBC value than the Poisson model. For the retail banking LoB, both the Poisson and negative binomial models have identical SBC values. This usually implies that the negative binomial model overfits the data. You can confirm this by observing the estimate of the _Alpha parameter; it is close to zero.

Figure 1 Fit Summary and Parameter Estimates for Count Models

line	Model	Property	PropertyValue
CommercialBanking	Poisson	SBC	3670.618
CommercialBanking	NegBin(p=2)	SBC	3605.559
RetailBanking	Poisson	SBC	2912.296
RetailBanking	NegBin(p=2)	SBC	2912.296

The final parameter estimates for the best models of each LoB are shown in Figure 2.

Figure 2 Parameter Estimates of the Best Selected Count Models

line=RetailBanking

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	0.167906	0.031452	5.34	<.0001
corpKRI1	1	1.017142	0.019652	51.76	<.0001
rbKRI1	1	0.506545	0.022196	22.82	<.0001
rbKRI2	1	0.224964	0.021750	10.34	<.0001

Figure 2 *continued*

line=CommercialBanking

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Intercept	1	0.550337	0.031249	17.61	<.0001
corpKRI1	1	0.705707	0.026977	26.16	<.0001
corpKRI2	1	0.320722	0.025791	12.44	<.0001
cbKRI3	1	0.551402	0.025861	21.32	<.0001
cbKRI2 * cbKRI3	1	0.344673	0.023246	14.83	<.0001
_Alpha	1	0.188980	0.025294	7.47	<.0001

You need to refit the final selected model to create an item store of estimates that the CCDM procedure can use. The following statements store the negative binomial model of the commercial banking line in the **mycas.OpriskStoreNegbin** item store and the Poisson model of the retail banking line in the **mycas.OpriskStorePoisson** item store:

```
/*** Refit the best count model of commercial banking LoB with its selected effects ***/
proc cntselect data=mycas.OpRiskLossCounts(where=(line='CommercialBanking'))
      store=mycas.opriskStoreNegbin(promote=yes);
   by line;
   model numloss=corpKRI1 corpKRI2 cbKRI3 cbKRI2*cbKRI3 /dist=negbin;
run;

/*** Refit the best count model of retail banking LoB with its selected effects ***/
proc cntselect data=mycas.OpRiskLossCounts(where=(line='RetailBanking'))
      store=mycas.opriskStorePoisson(promote=yes);
   by line;
   model numloss=corpKRI1 rbKRI1 rbKRI2/dist=poisson;
run;
```

The BY statement is necessary in the preceding steps even if you are fitting the model for one LoB at a time by using the WHERE= data set option, because PROC CCDM needs the BY variable information when it reads the item store. The PROMOTE=YES data set option ensures that the item store is available to CAS sessions other than the current CAS session that generates the item store.

SEVERITY MODELING

The SEVSELECT procedure in SAS Econometrics estimates severity models. The following key features of the procedure are relevant to the ECM process:

- In the simplest form, it fits multiple continuous probability distributions to the input loss severity data and prepares a comparative summary of the fit statistics of all candidate distributions to help you choose the best candidate. It includes a default set of 10 distributions: Burr, exponential, gamma, generalized Pareto (GPD), inverse Gaussian (Wald), lognormal, Pareto, scaled Tweedie, Tweedie, and Weibull.
- It can fit a distribution that you define on your own by using the FCMP procedure in Base SAS. For example, you can define a mixture of lognormal body and GPD tail or a mixture of multiple lognormal distributions.
- It enables you to specify censoring and truncation effects. It accounts for these effects by appropriately modifying the likelihood function and by using an appropriate method, such as Kaplan-Meier's or Turnbull's method, to estimate the EDF that it needs for initializing the parameters and estimating the goodness-of-fit statistics.
- It models the effect of regressors on the scale parameter or the log-scale parameter of the distribution. Hence, the severity models that include regression effects are referred to as scale regression models.
- It supports automatic model (variable) selection, where it identifies the best subset of regressors for each specified distribution. Primary supported variable selection methods include forward, backward, and stepwise.

- It enables you to define your own objective function, such as the Cramér–von Mises minimum distance estimator. By default, it uses the maximum likelihood method to estimate the parameters.
- It can store the fitted model to an item store that the CCDM procedure can later use.

The following statements show how to fit severity models for three candidate distributions—gamma, lognormal, and Weibull—to the example data:

```
/*** Fit severity models ***/
proc sevselect data=mycas.OpRiskLosses store=mycas.opriskSevstore(promote=yes)
               covout print=(allfitstats);
   by line;
   loss lossValue;
   dist gamma logn weibull;
   scalemodel corpKRI1 corpKRI2 cbKRI2 rbKRI1 rbKRI3 corpKRI1*rbKRI3;
   selection method=forward(select=aicc);
run;
```

The LOSS statement specifies the severity loss variable. The DIST statement specifies the list of candidate distributions. The STORE= option names an item store to hold the final model specifications and their parameter estimates. The COVOUT option requests that the parameter covariance estimates be kept in the item store to later help the CCDM procedure conduct parameter perturbation analysis. The SCALEMODEL statement lists all candidate regression effects. The SELECTION statement specifies that the best subset of regression effects be chosen according to the forward selection method, where candidate subsets are compared using the corrected Akaike's information criterion (AICC). Within each BY group, PROC SEVSELECT runs the regression effect selection process independently for each candidate distribution, and at the end of it displays the summary of the selection process, as shown in Figure 3 for the gamma distribution for the retail banking LoB.

Figure 3 Stepwise Selection Summary for the Gamma Distribution (Retail Banking LoB)

The SEVSELECT Procedure

Gamma Distribution
Selection Details

line=RetailBanking

Selection Summary

Step	Effect Entered	Number Effects In	AICC
0	**Intercept**	1	26268.0216
1	**rbKRI1**	2	25172.5534
2	**rbKRI3**	3	23644.9507
3	**corpKRI2**	4	22785.1467
4	**corpKRI1*rbKRI3**	5	22637.5511*
5	**corpKRI1**	6	22639.5436
6	**cbKRI2**	7	22641.5533

*** Optimal Value Of Criterion**

line=RetailBanking

Selection stopped because all effects are in the model.

line=RetailBanking

The model at step 4 is selected.

line=RetailBanking

Selected Effects: Intercept corpKRI2 rbKRI1 rbKRI3 corpKRI1*rbKRI3

After the selection process runs for all distributions, PROC SEVSELECT prepares a comparative summary of the

fit statistics of the final selected scale regression models of all candidate distributions, as shown in Figure 4. There are two classes of fit statistics: likelihood-based (AIC, AICC, and SBC) and EDF-based (KS, AD, CvM). Akaike's information criterion (AIC) is an asymptotic version of AICC. The Kolmogorov-Smirnov (KS), Anderson-Darling (AD), and Cramér–von Mises (CvM) statistics are the EDF-based statistics, which measure the distance between the CDF and the EDF. For scale regression models, the CDF depends on the values of the regressors, which can vary from one observation to another. So to compute the EDF-based statistics, PROC SEVSELECT uses the CDF of a mixture of distributions that are implied by a sample of observations. Although this choice of CDF makes the EDF-based statistics comparable across distributions, it also makes them less interpretable. So you should use the likelihood-based statistics to compare the scale regression models. For the example data, the lognormal distribution is the best for the commercial banking line, and the gamma distribution is the best for the retail banking line, according to all the likelihood-based fit statistics.

Figure 4 Comparison of Statistics of Fit for the Final Selected Severity Models

line=CommercialBanking

All Fit Statistics

Distribution	-2 Log Likelihood	AIC	AICC	SBC	KS	AD	CvM
Gamma	44297	44305	44305	44329	7.46106	339.56114	19.85227
Logn	44052*	44060*	44060*	44084*	6.90770*	291.56138	21.00704
Weibull	44518	44526	44526	44550	7.17637	286.36704*	13.35768*

Asterisk (*) denotes the best model in the column.

line=RetailBanking

All Fit Statistics

Distribution	-2 Log Likelihood	AIC	AICC	SBC	KS	AD	CvM
Gamma	22626*	22638*	22638*	22672*	8.25428	534.75332	27.24088
Logn	22825	22837	22837	22872	7.11756*	332.63032*	23.14326*
Weibull	22670	22682	22683	22717	8.35900	640.32342	25.28416

Asterisk (*) denotes the best model in the column.

The final set of selected parameters and their estimates are shown in Figure 5 for each LoB.

Figure 5 Parameter Estimates of the Best Selected Severity Models

line=CommercialBanking

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Mu	1	4.97340	0.01815	274.04	<.0001
Sigma	1	0.75306	0.00949	79.37	<.0001
corpKRI1	1	1.00112	0.01412	70.92	<.0001
cbKRI2	1	0.31003	0.01244	24.92	<.0001

line=RetailBanking

Parameter Estimates

Parameter	DF	Estimate	Standard Error	t Value	Approx Pr > |t|
Theta	1	30.70901	0.95468	32.17	<.0001
Alpha	1	2.10650	0.05683	37.06	<.0001
corpKRI2	1	0.51833	0.01567	33.08	<.0001
rbKRI1	1	-0.78056	0.01561	-50.00	<.0001
rbKRI3	1	0.58987	0.02052	28.74	<.0001
corpKRI1 * rbKRI3	1	0.19473	0.01563	12.46	<.0001

COMPOUND DISTRIBUTION MODELING

The CCDM procedure in SAS Econometrics estimates a compound distribution model (CDM) for the aggregate loss by combining the best frequency and severity models. Before illustrating the use of PROC CCDM, it is useful to define the CDM. If N and X represent the frequency and severity random variables, respectively, then the aggregate loss S is defined as $S = \sum_{j=1}^{N} X_j$. If $F_X(x)$ denotes the cumulative distribution function (CDF) of X and $\Pr(N = n)$ denotes the probability of seeing n losses as per the frequency distribution, then the CDF of S is theoretically computable as $F_S(s) = \sum_{n=0}^{\infty} \Pr(N = n) \cdot F_X^{*n}(x)$, where $F_X^{*n}(x)$ denotes the n-fold convolution of the CDF of X. The probability distribution model for S, characterized by $F_S(s)$, is referred to as a CDM. Direct computation of F_S is usually a difficult task because of the need to compute the n-fold convolution. Some methods exist (for example, Klugman, Panjer, and Willmot 1998, Ch. 4) for certain combinations of severity and frequency distributions, but even those methods do not offer closed-form or easily computable expressions for F_S. Further, when the distributions of X and N are conditional on external factors (regressors), as is usually the case, each set of regressor values results in a different distribution for N and X. For all these reasons, PROC CCDM estimates the CDM by using a Monte Carlo simulation method. The following key features of the procedure are relevant to the ECM process:

- In the simplest form, it accepts frequency and severity models that contain no regression effects in the form of item stores that the CNTSELECT and SEVSELECT procedures create, respectively, and simulates a sample of the aggregate loss. The simulation process makes a random draw from the frequency distribution to get a value of N, makes N random draws from the severity distribution, and adds those N severity values to form one sample point of the aggregate loss sample. The process is repeated as many times as the number of replicates that you specify. In the end, the aggregate loss sample is an empirical representation of the CDM. PROC CCDM analyzes this sample to display summary statistics and percentiles of the CDM. You can also write the sample to an output data table.
- It performs scenario analysis when frequency or severity models contain regression effects. In this case, it reads a scenario data table that contains the values of the regressors. For each observation, it uses the regressor values to compute the mean or scale of the frequency or severity distribution, and it makes random draws from the resulting frequency or severity distribution, respectively.
- It can perform parameter perturbation analysis to translate the uncertainty in the estimates of the frequency and severity model parameters to the uncertainty in the estimates of the CDM. In particular, it generates multiple perturbed samples and analyzes the summary statistics and percentiles of all such samples to compute the estimate of variability in each summary statistic and percentile estimate. To generate one perturbed sample, it perturbs the parameters of the frequency and severity models and uses those perturbed parameter values to make the random frequency and severity draws. It perturbs the parameters by using the covariance estimates that the CNTSELECT and SEVSELECT procedures create. The frequency and severity parameters are perturbed separately by making random draws from their respective multivariate normal distributions.

The following statements illustrate the use of the CCDM procedure for generating multiple perturbed CDM samples by using the item-store tables for the best frequency and severity models that the CNTSELECT and SEVSELECT steps in the preceding sections have estimated:

```
/*** Simulate CDM for commercial banking LoB ***/
proc ccdm data=mycas.multiConditionScenario(where=(line='CommercialBanking'))
          countstore=mycas.opriskStoreNegbin severitystore=mycas.opriskSevstore
          seed=13579 nreplicates=10000 nperturb=50 print=all;
   by line;
   severitymodel logn;
   output out=mycas.aggregateLossCB samplevar=agglossCB / perturbOut;
run;

/*** Simulate CDM for retail banking LoB ***/
proc ccdm data=mycas.multiConditionScenario(where=(line='RetailBanking'))
          countstore=mycas.opriskStorePoisson severitystore=mycas.opriskSevstore
          seed=13579 nreplicates=10000 nperturb=50 print=all;
   by line;
   severitymodel gamma;
   output out=mycas.aggregateLossRB samplevar=agglossRB / perturbOut;
run;
```

Because the frequency and severity models contain regression effects, you need to use the DATA= option to specify a data table that contains an external scenario that you want to analyze. This example uses a simulated scenario in the **mycas.MultiConditionScenario** data table. The NPERTURB= option requests that parameter perturbation analysis be conducted to generate 50 perturbed samples. The NREPLICATES= option specifies the size of each unperturbed and perturbed aggregate loss sample. The OUTPUT statement and its PERTURBOUT option specify that all 51 samples be stored in a data table. This is useful for estimating the uncertainty in the final ECM estimates in a later stage of the ECM process.

Figure 6 shows the perturbation analysis summary tables that the preceding PROC CCDM steps create for the commercial and retail banking lines. If you use the 99.5th percentile to estimate the VaR for each line, then you can use these results to conclude that for the scenario that the DATA= data table encodes, the VaR for the commercial banking business is approximately 26,986 $\pm$ 1,069 units, and the VaR for the retail banking business is 8,502 $\pm$ 406 units.

Figure 6 Perturbation Analysis of Aggregate Losses

line=CommercialBanking

Sample Perturbation Analysis

Statistic	Estimate	Standard Error
Mean	12219.9	375.41383
Standard Deviation	4474.6	186.23944
Variance	20056679	1692721.8
Skewness	0.77281	0.05025
Kurtosis	1.00014	0.18489

Number of Perturbed Samples = 50

Size of Each Sample = 10000

line=RetailBanking

Sample Perturbation Analysis

Statistic	Estimate	Standard Error
Mean	2524.7	110.95743
Standard Deviation	1554.8	75.33551
Variance	2423110.3	236571.7
Skewness	1.48392	0.07261
Kurtosis	3.05166	0.49169

Number of Perturbed Samples = 50

Size of Each Sample = 10000

line=CommercialBanking

Sample Percentile Perturbation Analysis

Percentile	Estimate	Standard Error
1	4241.2	212.73698
5	5937.8	237.13194
25	8999.7	280.21057
50	11663.4	350.53443
75	14831.2	463.58819
95	20375.1	694.31363
99	25105.5	943.77497
99.5	26985.9	1068.9

Number of Perturbed Samples = 50

Size of Each Sample = 10000

line=RetailBanking

Sample Percentile Perturbation Analysis

Percentile	Estimate	Standard Error
1	549.60864	25.37650
5	817.21183	33.25657
25	1390.7	60.38340
50	2104.0	100.12762
75	3267.9	152.79220
95	5600.6	255.11648
99	7665.1	352.30396
99.5	8502.0	405.72821

Number of Perturbed Samples = 50

Size of Each Sample = 10000

The perturbation analysis gives you an estimate of the variability in each statistic of interest, which is more informative than just the estimate of the mean. However, preparing such estimates requires a large amount of computation. Also, each sample needs to be large enough to capture as many intricacies of the distribution as possible, especially in the right-tail region of the distribution. The CCDM procedure is specifically designed to solve this *big computation* problem by making use of the distributed nature of the CAS server. It divides the simulation process among multiple computation nodes and among multiple processor threads within each node. It also creates the output sample table in a distributed fashion to take advantage of the distributed memory and storage capacities of the CAS server.

COPULA MODELING

The CCOPULA procedure in SAS Econometrics simulates a sample of uniform probabilities of losses across multiple random variables by accounting for the dependency structure among them. It accepts the dependency structure in terms of an elliptical copula (normal and t copulas) or an Archimedean copula (Clayton, Frank, and Gumbel copulas). A copula is especially useful for modeling the dependency structure in the ECM process, because it decouples the marginal distributions—that is, the probability distributions of individual random variables—from their joint probability distribution. This is due to an important theorem by Sklar (1959), which shows that a copula can be derived from any joint distribution function and that any copula can be combined with any set of marginal distributions to result in a multivariate distribution function. Different types of copulas focus on different measures of dependence, including linear correlation, rank correlation, and tail dependence. For financial applications, tail dependence is of special importance, because it measures the dependence when all random variables have extreme values. Of the copulas that PROC CCOPULA supports, the t copula captures dependence in both the left (lower) tail and the right (upper) tail, the Gumbel copula captures dependence in the right tail, and the Clayton copula captures dependence in the left tail (McNeil, Frey, and Embrechts 2015). Normal and Frank copulas do not show tail dependence (Venter (2002)).

You can use the COPULA procedure in SAS/ETS software, which is included with SAS Econometrics, to fit the copulas. The following statements illustrate its use for the example in this paper:

```
proc copula data=OpRiskMatchedLosses(where=(CB > 0 and RB > 0));
   var CB RB;
   fit Gumbel  / marginals=empirical plots(tail unpack)=(datatype=uniform);
   fit T       / marginals=empirical;
   fit Frank   / marginals=empirical;
   fit Normal  / marginals=empirical;
   fit Clayton / marginals=empirical;
run;
```

This analysis considers only observations in which both LoBs incur a nonzero loss. The VAR statement specifies the marginal variables that appear in the DATA= data set. The FIT statements fit the specified copulas. The marginals in the input data set are on the original loss scale, so specifying MARGINALS=EMPIRICAL tells PROC COPULA to transform them to uniform scale before fitting the copulas. The PLOTS= option prepares the tail dependence plot and a scatter plot of marginals in the uniform probability scale. All copulas are fit by using the default maximum likelihood method.

Figure 7 shows a summary of a fit statistic (SBC) for the copulas other than the normal copula. The normal copula essentially estimates the correlation matrices, and you judge its adequacy by observing the plots. If the plots show any tail dependence, then it is best to choose a copula other than the normal copula. The plots of the example data are shown in Figure 8. The first plot is a scatter plot of the data transformed to the uniform scale. It shows some clustering of points closer to the (1,1) corner. However, the second plot, which is a chi plot (Fisher and Switzer 2001), helps you visualize the tail dependence more clearly. The chi plot shows no evidence of values clustering around the λ axis, and there are a large number points with a positive value of λ. These two observations indicate that the two business lines have some degree of tail dependence. The fit statistics in Figure 7 further corroborate this, because the Gumbel copula, which focuses on the right-tail dependence, has the best fit by virtue of having the smallest SBC value.

Figure 7 Fit Statistics for Copula Models

Copula	SBC
Gumbel	-124.18895
T	-110.31631
Frank	-119.09111
Clayton	-57.18754

Figure 8 Transformed Data and Tail Dependence for Aggregate Losses in Commercial and Retail Banking Lines

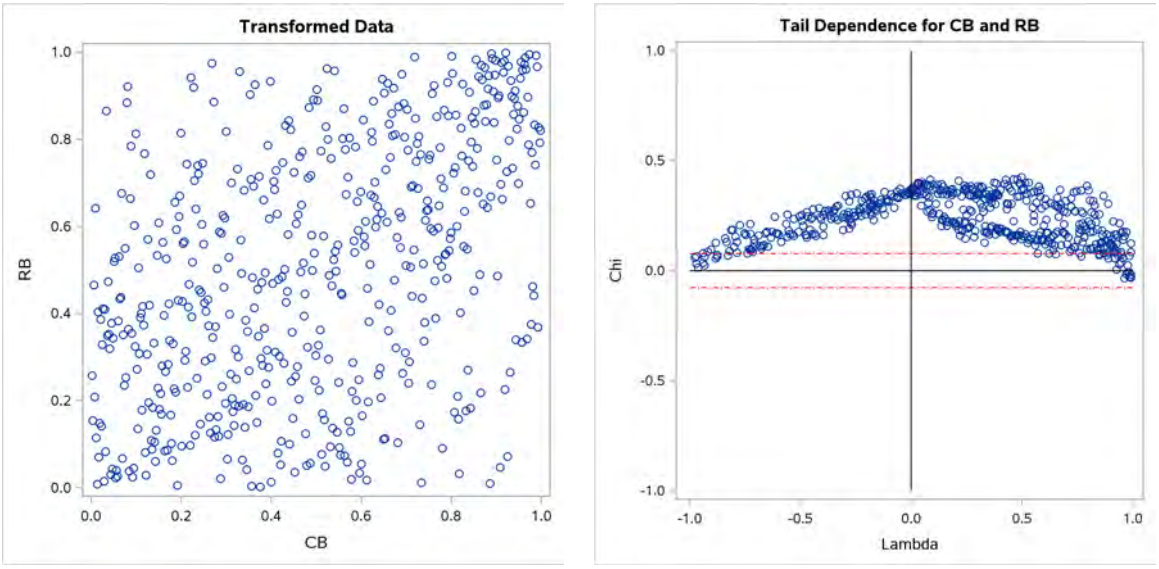

Figure 9 Parameter Estimate for the Gumbel Copula

Parameter Estimates

Parameter	Estimate	Standard Error	t Value	Approx Pr > |t|
theta	1.428523	0.047913	29.82	<.0001

The parameter estimate for the Gumbel copula is shown in Figure 9. The following CCOPULA step uses that estimate to simulate a large number of observations from the Gumbel copula:

```
proc ccopula;
   var CB RB;
   define cop gumbel (theta=1.428523);
   simulate cop / ndraws=10000 seed=234 outuniform=mycas.CbRbProbs;
run;
```

The DEFINE statement specifies the Theta parameter of the Gumbel copula. The SIMULATE statement simulates 10,000 observations from this copula. The OUTUNIFORM= option specifies that the observations be generated in the uniform scale and stored in the **mycas.CbRbProbs** data table. PROC CCOPULA uses the multiple nodes of a CAS session to simulate and store the data in a distributed manner. The first few simulated observations in the data table are shown in Figure 10. This data table, together with the data tables that the PROC CCDM steps in the preceding section generate, forms the input to the next and final step of the ECM process.

Figure 10 Simulated Copula Observations

CB	RB
0.0462542541	0.2593076527
0.9719390327	0.9786392405
0.1763144399	0.2983883887
0.4415681006	0.9819339936
0.6899311222	0.7391266716
0.9241566139	0.5287517597
0.8918401679	0.3469049918
0.5311774348	0.5711362055
0.2130563546	0.4256233217
0.6799937875	0.5902953134

ECONOMIC CAPITAL MODELING: COMBINING COPULA AND CDM

The final step of the ECM process is to estimate the distribution of the total loss across all business lines of interest. This requires transforming each observation in the copula simulation table, which contains the probability of observing a loss in each LoB, to an observation that contains an estimate of the actual loss for each LoB. Each LoB represents a marginal random variable in the copula model (which is simply referred to as a marginal in the rest of this paper). Converting a probability of an LoB to its loss is equivalent to inverting the corresponding marginal's cumulative distribution function (CDF). In other words, the loss estimate for an LoB is equal to the quantile estimate for the corresponding marginal distribution. Formally, let F_k denote the CDF of the kth marginal, and let p_{ik} denote the cumulative probability of the kth marginal in the ith observation in the copula simulation table. $F_k^{-1}(.)$ is the quantile function for the kth marginal. If there are K marginals, then the total loss for the ith observation is

$$L_i = \sum_{k=1}^{K} F_k^{-1}(p_{ik})$$

If the copula simulation table contains N observations, then computation of L_i for all N observations leads to a sample of size N for the total loss. This sample essentially encodes an empirical estimate of the ECM. You can use it to compute summary statistics, percentiles, and risk measures such as VaR and TVaR. The larger the N, the more accurate the estimates.

The key points to note about the ECM estimation process are as follows:

- The CDF F_k of a marginal is often not available in closed form, which is the reason PROC CCDM prepares an empirical sample for each marginal. You can approximate the CDF by estimating $\hat{F}_k$, the empirical distribution function (EDF) of the kth marginal. Inverting $\hat{F}_k$ at probability p_i is equivalent to estimating the p_ith percentile from the empirical sample.
- When severity or frequency models that are used to compute the aggregate loss distribution contain regression effects, the CDF F_k also depends on the regression effects. Consequently, the empirical sample that PROC CCDM creates to estimate the EDF $\hat{F}_k$ depends on the specific scenario that you supply to PROC CCDM. The scenario records one possible future state of the world. If you want to analyze multiple scenarios, you need to repeat the ECM process for each scenario independently.
- Uncertainty in the parameter estimates of the severity and frequency models results in uncertainty in the CDF estimates, and consequently in uncertainty in the EDF estimates of each marginal. Thus, the ECM estimates that are computed by inverting the EDF are also uncertain. PROC CCDM captures the uncertainty in individual marginal distributions by simulating multiple perturbed samples and summarizing them to estimate the location and dispersion (variability) of various summary statistics and percentiles. Similarly, to estimate the location and dispersion in the ECM estimates, you need to generate multiple samples of the total loss by using different combinations of the perturbed samples of all the marginals.

This paper proposes a method that computes both location and dispersion estimates of the risk measures of the distribution of the total loss for a given scenario. For the example in this paper, the method is illustrated by the following code. These programming statements define and use a SAS macro to iterate over different perturbed samples of each marginal, generate an ECM sample for each combination of the marginal samples, compute and accumulate the summary statistics and percentiles of each ECM sample, and finally use a PROC MEANS statement to compute the estimates of location and dispersion for each summary statistic and risk measure of interest.

```
%macro EstimateECMvariability(maxDrawId1=10, maxDrawId2=10);
   %do drawId1=0 %to &maxDrawId1;
      %do draw2=0 %to &maxDrawId2;
         %EstimateECM(inlibref=mycas, copulaSim=CbRbProbs, copulaVars=CB RB, nmarg=2,
                   marginalTables=aggregateLossCB aggregateLossRB,
                   marginalVars=agglossCB agglossRB, drawIds=&drawId1 &draw2,
                   pctldef=5, outsum=ecmstats);
         %if &drawId1=0 and &draw2=0 %then %do;
            data allecmstats;
               set ecmstats;
            run;
         %end;
```

```
         %else %do;
            proc datasets library=work nolist;
               append base=allecmstats data=ecmstats;
            run;
         %end;
      %end;
   %end;
   proc means data=allecmstats n mean std median qrange;
      var _mean_ _median_ VaR95 TVaR95 VaR97_5 TVaR97_5 VaR99_5 TVaR99_5;
   quit;
%mend;

%EstimateECMvariability(maxDrawId1=20, maxDrawId2=20);
```

The %EstimateECM macro, which is the workhorse in the preceding steps, generates the ECM sample for one combination of the marginal samples. It accepts as input the copula simulation table that PROC CCOPULA generates (COPULASIM= option), the names of the marginal variables in that table (COPULAVARS= option), and specifications of marginal samples to use. You specify the marginal samples by specifying the following lists: the list of marginal tables that PROC CCDM creates (MARGINALTABLES= option), the list of the draw (sample) identifiers of the samples to use from those marginal tables (DRAWIDS= option), and the list of the names of the variables that contain the sample (MARGINALVARS= option). All lists must have as many elements as the number of marginals that you specify (NMARG= option). You can control the method of computing the percentiles by using the PCTLDEF= option, which can specify an integer value between 1 and 5 and works in the same way as the PCTLDEF= option in the UNIVARIATE procedure. The macro writes the summary statistics of the generated ECM sample to the OUTSUM= data set. The pseudo-code of the %EstimateECM macro is as follows:

```
%macro EstimateECM(inlibref=mycas, copulaSim=, copulaVars=, nmarg=, debug=0,
                   marginalTables=, marginalVars=, drawIDs=, pctldef=5, outsum=);
   /*** Parse and validate input arguments ***/
   .....
   /*** Read number of observations in each marginal table ***/
   proc cas; %do im = 1 %to &nmarg;
      recordCount result=r / table={name="&&margTab&im", where="_drawid_=&&drawId&im"};
      symputx("marg&nval.nobs", r.RecordCount[1,1]);
   run;      %end;
   quit;
   /*** Generate total loss sample by using a server-side DATA step ***/
   data &inlibref.._tlosssamp_(keep=tloss) / sessref=mysess single=yes;
      /*** Define some arrays ***/
      %do im = 1 %to &nmarg;
         array m&im.{&&marg&im.nobs} _temporary_;
      %end;
      array d{&nmarg} _temporary_;
      array mPctl{&nmarg} _temporary_;
      .....
      if (_n_ = 1) then do;
         %do im = 1 %to &nmarg;
            /*** Read a marginal sample to an array ***/
            %ReadCDMSample(table=&inlibref..&&margTab&im,
               var=&&margVar&im, array=m&im., drawId = &&drawId&im);
            d{&im} = dim(m&im);
            /*** Sort the marginal sample ***/
            %ArrayQuickSort(&im, m&im., 0);
         %end;
      end;
      /*** Read an observation from the copula simulation table ***/
      set &inlibref..&copulaSim;
      /*** Compute percentiles for all marginals ***/
      %do im = 1 %to &nmarg;
         %ComputePercentile(array=m&im, n=d{&im}, pctldef=&pctldef,
                            prob=&&copVar&im, pctl=mPctl{&im});
      %end;
```

```
      /*** Compute total loss across all marginals ***/
      tloss = 0;
      do im = 1 to &nmarg;
         tloss = tloss + mPctl{im};
      end;
   run;
   /*** Summarize total loss sample by invoking CAS actions ***/
   proc cas;
      simple.summary result=sr / table={name="_tlosssamp_", vars={"tloss"}},
         subset={"N", "MIN", "MAX", "MEAN", "STD", "VAR", "SKEWNESS", "KURTOSIS"},
         casout={name="_tmpsumm_", replace=True};
      percentile.percentile result=pr / table={name="_tlosssamp_", vars={"tloss"}},
         values={1, 5, 25, 50, 75, 90, 95, 97.5, 99, 99.5} pctldef=&pctldef,
         casout={name="_tmppctl_", replace=True};
   quit;
   /*** Compute TVaR for a set of percentiles ***/
   %ComputeTVaR(data=&inlibref.._tlosssamp_, tlossvar=tloss,
      pctldata=&inlibref.._tmppctl_, pctlpts=90 95 97.5 99 99.5,
      outdat=&inlibref.._tmptvar_);
   /*** Prepare summary output table ***/
   proc transpose data=&inlibref.._tmppctl_ prefix=VaR
         out=xoutpctl(drop=_name_ rename=(VaR50=_Median_));
      id _pctl_; var _value_;
   run;
   proc transpose data=&inlibref.._tmptvar_ prefix=TVaR out=xouttvar(drop=_name_);
      id _pctl_; var _tvar_;
   run;
   data &outsum;
      merge &inlibref.._tmpsumm_(drop=_column_) xoutpctl xouttvar;
   run;
   /*** Delete temporary data tables ***/
   .....
%mend;
```

Key points to note about the implementation of the %EstimateECM macro are as follows:

- It uses PROC CAS steps to submit CAS actions such as the **recordCount** action. Each action runs on the CAS server to perform the computations that it is designed for and sends some results back to the client machine that invokes the action. Each action has its own fixed set of parameters, just as each SAS procedure has its own unique set of statements and options. One of the key benefits of CAS actions is that they can be invoked from different languages, such as Python, Lua, and CASL, which is a new language that you can use inside a PROC CAS step.

- The macro first finds the number of observations in each marginal's sample for the specified draw (sample) identifier. The PROC CAS step submits the **recordCount** action for each marginal table along with an appropriate where clause. This step is needed to define the arrays inside the DATA step to store all marginal samples.

- The macro uses a DATA step to prepare the sample for the total loss. The SESSREF= option specifies that the DATA step run on the CAS server. This is important, because without that option, the DATA step runs on the client machine that invokes the macro and brings all the referenced data tables back to the client machine; this can be very costly, especially if the CAS server is running in a cloud. By comparison, the cost of moving the data among the nodes of the server is lower. In fact, this macro uses the SINGLE=YES option in the DATA statement to bring all the observations in each marginal table's specified draw and all the observations in the copula simulation table to a single node of the server to perform the required percentile computations.

- Before reading the first observation from the copula simulation table, the %EstimateECM macro invokes the %ReadCDMSample macro to read all specified marginal samples into their respective arrays, followed by the %ArrayQuickSort macro to sort each array in ascending order. This is necessary for the %ComputePercentile macro to efficiently compute the percentile for each probability value that the DATA step reads from the copula simulation table.

- The %ComputeTVaR macro computes an empirical estimate of the TVaR, which is the mean of the losses that exceed the empirical estimate of the VaR. In particular, if L_p denotes the value of the pth percentile, which is the empirical estimate of VaR_p, then the empirical estimate of TVaR_p is

$$\text{TVaR}_p = \frac{\sum_{i:l_i \geq L_p} l_i}{|\{i : l_i \geq L_p\}|}$$

  where l_i denotes the total loss value in the ith observation. The macro uses a DATA step that runs on the CAS server, and computes the TVaR estimates for all the percentile values that you specify by making one pass over the total loss sample table.

The full implementations of the %EstimateECM and all other supporting macros are available with this paper on the conference proceedings website (http://support.sas.com/resources/papers/proceedings18/).

The final results that the %EstimateECMvariability macro prepares are shown in Figure 11. If you define VaR to be the 97.5th percentile of the total loss distribution, then for the example data in this paper, the VaR estimate is about 27,743 $\pm$ 690 if you use the mean and standard deviation as measures of location and dispersion. The corresponding TVaR estimate is 31,231 $\pm$ 794. The VaR estimate is about 27,633 $\pm$ 874 if you use the median and interquartile range as measures of location and dispersion. The corresponding TVaR estimate is 31,089 $\pm$ 1,038.

Figure 11 Location and Dispersion Estimates of Various ECM Summary Statistics and Percentiles

The MEANS Procedure

Variable	N	Mean	Std Dev	Median	Quartile Range
Mean	441	14796.81	324.6491534	14797.76	455.9070340
Median	441	14022.61	310.6647057	14046.68	438.1942189
VaR95	441	25018.97	603.8322928	24930.16	729.6729054
TVaR95	441	28746.19	714.7421021	28622.78	914.8240041
VaR97_5	441	27742.53	690.0607936	27633.44	874.4451691
TVaR97_5	441	31231.19	793.5280546	31088.80	1037.81
VaR99_5	441	33455.23	878.5266001	33290.55	1201.01
TVaR99_5	441	36349.90	963.3442772	36141.74	1368.75

NOTE: The %EstimateECM macro can require significant time and memory resources if the copula simulation table and CDM sample tables are large and are distributed across multiple nodes of a CAS server. Although the macro is designed to avoid bringing the sample data back to the client machine, it still requires the data to be moved to a single node of the CAS server, which can take a long time for large tables. Also, the DATA step inside the macro requires memory that is proportional to $K \times N$, where K is the number of marginals and N is the size of each marginal's sample. This can become prohibitively large as K and N grow. Future work should address this problem.

WIDEN YOUR HORIZONS: IMPLEMENTING THE ECM PROCESS IN PYTHON AND OTHER LANGUAGES

An important feature of SAS Viya is that all the analytic computations are implemented as CAS actions that run on a CAS server. These actions are accessible from different clients in different programming languages. The preceding section has already illustrated one such client, PROC CAS, which uses a new language called CASL. Other popular programming languages that you can use to invoke CAS actions are Python, Lua, and Java. In fact, you can also think of every SAS Econometrics procedure as a client that uses the traditional SAS language interface, because all those procedures work by invoking different CAS actions and presenting the results that the actions return in traditional SAS output format, such as SAS ODS tables.

This section illustrates how you can use the Python language interface to implement the ECM process steps that use the CNTSELECT, SEVSELECT, CCDM, and CCOPULA procedures. Because space is limited, the Python code is shown for only one of the representative procedure invocations in each step, and the Python code is not shown for the final step of generating the ECM samples. But a savvy Python programmer should be able to implement it relatively easily by using the description and SAS macro code in the preceding section as a guide.

Before you submit CAS actions from a Python interface, you need to establish a connection to the CAS server by running the following code, which assumes that the host name of the CAS server controller is “cloud.example.com” and it is listening on port 5570:

```
import swat
s = swat.CAS("cloud.example.com", 5570)
```

FREQUENCY (COUNT) MODELING IN PYTHON

The following Python code loads the **countreg** action set and submits the **countregFitModel** action:

```
s.loadactionset(actionset="countreg")
res = s.countregFitModel(
   table={'name':'OpRiskLossCounts', 'groupby':['line'],
          'where':'line="CommercialBanking"'},
   model={'depVars':[{'name':'numloss'}], 'type':'CNTMODEL',
          'effects':[{'vars':{'corpKRI1','corpKRI2','cbKRI3'}},
                     {'vars':{'cbKRI2','cbKRI3'}, 'interaction':'CROSS'}],
          'modeloptions':{'modeltype':'NegBin2', 'noint':False}},
   store={'name':'opriskStoreNegbin', 'promote':True} )
```

This is equivalent to the following PROC CNTSELECT step:

```
proc cntselect data=mycas.OpRiskLossCounts(where=(line='CommercialBanking'))
      store=mycas.opriskStoreNegbin(promote=yes);
   by line;
   model numloss=corpKRI1 corpKRI2 cbKRI3 cbKRI2*cbKRI3 /dist=negbin;
run;
```

SEVERITY MODELING IN PYTHON

Before you fit severity models in Python or any other language, you need to make the definitions of the severity distributions available in an in-memory CAS table that is accessible to your CAS session. When you install SAS Econometrics, the **samples** caslib on the server nodes contains a presaved CAS table named **predef_svrtdist.sashdat** that includes the definitions of the following 10 distribution families: Burr (burr), exponential (exp), gamma (gamma), generalized Pareto (gpd), inverse Gaussian (igauss), lognormal (logn), Pareto (pareto), reparameterized Tweedie with the scale parameter (stweedie), standard Tweedie with the mean parameter (tweedie), and Weibull (weibull), where the names in the parentheses are the names that you specify in the **distributions** parameter of the **severity** action. You can load this presaved CAS table into an in-memory CAS table named **mysevdists** by submitting the following Python code:

```
s.loadtable(caslib='samples', path='predef_svrtdist.sashdat',
            casout="mysevdists")
```

Alternatively, you can run the following PROC SEVSELECT step by connecting to your CAS server from a SAS® Studio client:

```
proc sevselect;
   dist _predefined_ tweedie stweedie / outfuncdef="mysevdists"(global);
run;
```

You also need to use this PROC SEVSELECT alternative whenever you define or update your own severity distributions by using the FCMP procedure. For more information, see *SAS Econometrics: Econometrics Procedures*.

The following Python code loads the **severity** action set and submits the **severity** action by using the **funcDef** parameter to specify the **mysevdists** CAS table that contains the severity distribution definitions:

```
s.loadactionset(actionset="severity")
res = s.severity(funcDef="mysevdists", table={'name':'OpRiskLosses', 'groupby':['line']},
   lossVariables=[{'name':'lossValue', 'role':'TARGET'}],
   distributions={'logn', 'gamma', 'weibull'},
   scaleModel={'effects':[{'vars':{'corpKRI1','corpKRI2','cbKRI2','rbKRI1','rbKRI3'}},
                          {'vars':{'cbKRI2','cbKRI3'}, 'interaction':'CROSS'}]},
   selection={'method':'FORWARD', 'stop':'AICC'},
   store={'name':'opriskSevstore', 'promote':True}, storeCov=True )
```

The Python code is equivalent to the following PROC SEVSELECT step:

```
proc sevselect data=mycas.OpRiskLosses store=mycas.opriskSevstore(promote=yes) covout;
   by line;
   loss lossValue;
   dist logn gamma weibull;
   scalemodel corpKRI1 corpKRI2 cbKRI2 rbKRI1 rbKRI3 corpKRI1*rbKRI3;
   selection method=forward(select=aicc);
run;
```

COMPOUND DISTRIBUTION MODELING IN PYTHON

The following Python code loads the **cdm** action set and submits the **cdm** action:

```
s.loadactionset(actionset="cdm")
res = s.cdm(
   severityDefinitions={'name':'mysevdists'},
   table={'name':'multiConditionScenario', 'groupby':['line'],
          'where':'line="CommercialBanking"'},
   severityDistributions={'logn'}, severityStore={'name':'opriskSevstore'},
   countStore={'name':'opriskStoreNegbin'},
   nReplicates=10000, seed=13579, nPerturbedSamples=50,
   output={'outSample':{'name':'aggregateLossCB', 'replace':True},
      'sampleVariable':'agglossCB', 'perturbOut':True} )
```

Note that the **mysevdists** data table of severity distribution definitions is the same table that you use for the **severity** action.

The Python code is equivalent to the following PROC CCDM step:

```
proc ccdm data=mycas.multiConditionScenario(where=(line='CommercialBanking'))
          countstore=mycas.opriskStoreNegbin severitystore=mycas.opriskSevstore
          seed=13579 nreplicates=10000 nperturb=50;
   by line;
   severitymodel logn;
   output out=mycas.aggregateLossCB samplevar=agglossCB / perturbOut;
run;
```

COPULA MODELING IN PYTHON

The following Python code loads the **copula** action set and submits the **copulaSimulate** action:

```
s.loadactionset(actionset='copula')
m = s.copulaSimulate(
    define={'copulatype':'gumbel', 'theta': 1.428523, 'name':'cop'},
    var=['CB','RB'], ndraws=10000, seed=234,
    outuniform={'name':'CbRbProbs'})
```

This is equivalent to the following PROC CCOPULA step:

```
proc ccopula;
   var CB RB;
   define cop gumbel (theta=1.428523);
   simulate cop / ndraws=10000 seed=234 outuniform=mycas.CbRbProbs;
run;
```

CONCLUSION

This paper illustrates the economic capital modeling process, which starts with the data collection and preparation phase and results in an estimate of the distribution of the enterprise-wide losses. Not only does the proposed method of estimating an ECM compute the point estimates of the summary statistics and risk measures of the total loss distribution, but it also translates the uncertainty in the parameter estimates of the frequency and severity models to

estimate the uncertainty (variability) in the summary statistics and risk measures of the total loss distribution. This is accomplished by using the parameter perturbation analysis feature of the CCDM procedure.

The paper shows you how to implement each step by using a SAS Econometrics procedure or an available SAS tool. Part of the process is exploratory, requiring you to select the best distribution and the best subset of external effects for the frequency and severity models. Almost all the steps work better when you have large amounts of data and need greater computational efficiency. All SAS Econometrics procedures help you handle large, distributed data and provide the required computational efficiency by making full use of the distributed computing framework of SAS Viya. They perform the computations closer to the data and attempt to minimize data movement.

REFERENCES

Fisher, N. I., and Switzer, P. (2001). "Graphical Assessment of Dependence: Is a Picture Worth 100 Tests?" *American Statistician* 55:233–239.

Klugman, S. A., Panjer, H. H., and Willmot, G. E. (1998). *Loss Models: From Data to Decisions*. New York: John Wiley & Sons.

McNeil, A. J., Frey, R., and Embrechts, P. (2015). *Quantitative Risk Management: Concepts, Techniques, and Tools*. Rev. ed. Princeton, NJ: Princeton University Press.

Sklar, A. (1959). "Fonctions de répartition à *n* dimensions et leurs marges." *Publications de l'Institut de Statistique de L'Université de Paris* 8:229–231.

Venter, G. G. (2002). "Tails of Copulas." In *Proceedings of the Casualty Actuarial Society*, vol. 89, 68–113. Baltimore: United Book Press.

ACKNOWLEDGMENTS

The author is grateful to Mark Little, Jan Chvosta, and Richard Potter of the Advanced Analytics Division at SAS Institute for their valuable assistance in the research and development of the procedures discussed in this paper. Thanks are also due to Ed Huddleston for his valuable editorial comments.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author:

Mahesh V. Joshi
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
919-531-1272
919-677-4444 (Fax)
mahesh.joshi@sas.com