A System to Calculate Market Value-at-Risk using SAS/IML® and Oracle®

SESUG George Rezek, GMAC Enterprise Risk Management, Minneapolis, MN

ABSTRACT

SAS/IML® is used with Base SAS and Oracle® to produce a system to calculate value at risk with the flexibility to reflect changes in the database in the calculation and reporting. Value at risk is calculated using Monte Carlo simulation. The overall process is covered and aspects of the calculation are highlighted. Finally, the impact of the results on the Enterprise are outlined..

INTRODUCTION

Three years ago upper management at GMAC Mortgage Group decided that it was important to establish policies and programs designed to monitor the market, credit, and operational risks the three companies of the Group faced. The program described here was written to measure the risks of market assets of the enterprise. SAS/IML® greatly facilitated the task and the elegance of the resultant code has given transparency to the process.

WHAT IS VALUE-AT-RISK?

Value-at-Risk is the maximum amount of money that may be lost on a portfolio on average over a given period of time, with a given confidence interval *under normal circumstances*. For traders this period of time, or holding period, can be a day or less. For our model the holding period is one month. When we say that our VaR is -$47m for a given company at the 95th confidence interval, we mean that, under normal markets conditions, we expect to lose more than $47m, 5% of the time. Note that this does not consider catastrophic events.

VALUE-AT-RISK at GMAC

While there are various ways of calculating Value-at-Risk, we use a two factor, interest rate and spread, correlation model. The help explain the mechanics of the model, I've illustrated an same example involving three risk factors, three products, running 10 simulations in Appendix A.

PRODUCT PROFILES

The model assumes that each product is at risk because of movements in interest rates and spreads, and we associate an IR security to each product that makes sense, such as the 5 year Treasury bond, and use that as the products interest rate risk factor. We also associate another series as the spread factor, which is the difference in bid and ask price of that product. We measure the historical movement of all such risk factors by calculating the standard deviation over a 60 month period. As we see on the right side of the chart, these standard deviations or volatilities are used by traders at their desks in models to "shock" the corresponding products to produce product profiles. Please note that I've subtracted the base value (the nominal value) from all seven elements of the profiles, as we're interested in the amount we expect to gain or lose. In our case each product has two profiles, one for interest rate, one for spread, that reflect the dollar value of the position in response to up and down three standard deviation, or sigma, moves in the associated risk factor. From these profiles we see how a given movement in interest rate can cause a gain in one product and a loss in another.

Traders are accustomed to conducting such sensitivity analysis and we continued to use their input, even though this was less efficient than performing such analysis in the model. We also worried that the process could produce inconsistent profiles; two traders at different companies could use different programs to shock their products or use different assumptions in the same model and derive different profiles for the same product. In fact, unexpected VaR results prompted market managers to review assumptions in these models. Allowing traders retain this function helped us win confidence in our results.

RISK FACTOR SIMULATION

On the left side of the chart, we note that the 60 period risk factor table is used to calculate a correlation and covariance table. From the correlation table we see the extent to which each risk factor move with or against each other risk factor. The Singular Value Decomposition function in IML uses the covariance matrix of risk factors, the number of simulations and the random number generator to create a matrix of simulated risk factors. It's a simple matter to verify that the SVD function is working: computing the covariance matrix of the simulated risk factor matrix produces the same covariance matrix used in the SVD function.

Each element of the simulated matrix is standardized by the corresponding risk factor's standard deviation. The standardized risk factor matrix consists of units (sigma's) that can be compared to the product profiles the traders have provided. We create profit and loss matrices, one for interest rate, one for spread, by interpolating each product's risk factor 's column of simulated sigma's (standardized) against that product's profile. This results in two columns (one for interest rate, one for spread) of profit and loss values for each product; the number of rows corresponds to the number of simulations.

We can see that these profit and loss results are correlated because they are determined using risk factors that are correlated. Add the interest rate and spread profit and loss tables to obtain the total profit and loss results. Add all of these columns (products) laterally to get a single column of profit and loss; sort this column in descending order, VaR at the 95% confidence interval is the 95^{th} percentile (or 5^{th} percentile from the bottom) of that column.

VaR can be calculated similarly for any level of aggregation, or even at the product level. Simply add all the columns that comprise a given company, or business unit, etc., across and determine the 95^{th} (or 99.9^{th} or 99.95^{th}) percentile of the resulting column. The main point here is that, because the elements within each row of the profit and loss matrix are correlated, one cannot sum product level VaRs and obtain the total VaR. Theoretically, it is possible that a level of aggregation can equal the sum of the products, but typically the diversification benefit, the difference in the sum of the product VaR's to the total VaR, caused by the difference in profiles of the products and the correlation of the risk factors, is quite significant. One must choose those products to be included with the level of aggregation desired and sum the corresponding columns into a single column, which is then used to determine VaR at the various confidence intervals.

THE OLD MODEL

Three separate installations of a system to calculate VaR existed at the three companies that comprise the GMAC Mortgage Group. These systems had been written in VBA in MS Access. The system had serious drawbacks:

- Without extensive rewrites to the code the system could not accept new inputs in either risk factors nor products
- the system was unable to replicate runs for previous periods without removing current data from the database
- it was slow
- it didn't produce a total VaR number for the entire Mortgage Group
- the methodology was buried in confusing code and was therefore not transparent

The task was to rewrite this existing system to correct these shortcomings. A database in Oracle was designed to accommodate all elements necessary for a Mortgage Group level VaR calculation. SAS/Base and SAS/IML were chosen to create an efficient, flexible and transparent system.

ORACLE DATABASE

The database is designed to:

- maintain description of the variables used in the model
- accommodate expansion in the reporting structure
- validate the data

The database schema is illustrated in Appendix B. The reporting structure reflects the organization of the Enterprise, company group, company , business, business unit, and desk. The desk summarizes the structure and each product is assigned to a single desk, i.e., product numbers are assigned such that single product cannot appear in two different desks.

While there are several tables to maintain audit trails and aid in special calculations, we need to focus on just a few tables for our discussion. The monthly risk factor updates are stored in the risk_factor_value table and are described in the risk_factor table. As this database is also the repository for the counterparty system, this descriptive table is used to select only risk factors the VaR model uses. The monthly profiles and associated risk factors are entered into the product_market_value table and these products are described in the product table. Referential integrity ensures that a risk factor exists for each risk factor entered into the product_market_value table. Finally we maintain the results for all business unit levels, for all confidence intervals, for all months reported in the var_result table.

THE NEW MODEL

As the existing model based much of its assumptions on J.P Morgan's *Risk Metrics – Technical Document* that required coding customized statistics, SAS/IML was chosen for the rewrite.

VOLATILITIES

We assume that the distribution of the interest rate risk factors to be log normal and that of spread to be normal; we use the log of the ratio of adjacent values for interest rate factor and the difference of adjacent values for spread. For computational ease interest rate and spread factors were read from the database and denormalized into separate datasets using proc transpose. The risk_factor_id's , a numeric field that becomes the column names after proc transpose, e.g., values for risk_factor_id 12 appear under the column header of "_12." We will need to retain this series to use to label our results (we repeat the following code for spread factors as well):

```
proc transpose data = risk_val_IR
   out = risk_val_IR_flat
         (drop=_NAME_ _LABEL_ valdat) ;
  id risk_factor_id ;
  by descending valdat ;

proc sql ;
  %global risk_number_IR_list ;
  select input(substr(name,2,3),3.)
    into :risk_number_IR_num separated by ','
    from dictionary.columns
      where libname='WORK'
        and memname='RISK_VAL_IR_FLAT'  ;
quit ;
```

The methodology calls for a dampening effect of .97 for monthly data to be incorporated into the calculation of the standard deviation, i.e., a weighting scheme such that more recent values count for values further in the past. This was accomplished with very few lines of code compared to the original model (maxrow is the number of periods back we want to go, which we initially had to vary for lack of data). First, we transform our interest rate risk factors and calculate the vector of weights:

```
IR_trans =  log(dataAll_IR[1:&maxrow-1,]) -  log(dataAll_IR[2:&maxrow,]) ;
back = &maxrow -1 ;
lambda = 0.97 ;
index = (1:back)` ;
w = (1-lambda) * lambda**(index-1)/ (1-lambda**back) ;
```

Then we calculate the volatilities for interest rate factors:

```
mu = w # IR_trans ;
mu = j(back,1,1) * mu[+,] ;
cov_ir = (w # (IR_trans-mu))` * (IR_trans-mu) * (back/(back-1)) ;
VolGT = sqrt(vecdiag(cov_ir))` ;
mean_IR = (log(dataAll_IR[1,]) - log(dataAll_IR[&maxrow,]))/(&maxrow - 1);
regu = mean_IR +VolGT#VolGT/2;
sd_IR  = 100 * dataAll_IR[1,]#exp(regu) # sqrt(exp(VolGT # VolGT) - 1) ;
```

This represents a considerable savings in lines of code compared to the original VBA program. The elegance of matrix algebra also makes the methodology more transparent. Because the spread series are first differenced and the mean is assumed to be zero, calculation is somewhat simpler:

```
nofac_SP = ncol(dataAll_SP) ;
SP_trans = dataAll_SP[1:&maxrow-1,] - dataAll_SP[2:&maxrow,] ;
mu = w # SP_trans ;
mu = j(back,1,1) * mu[+,] ;
cov_SP = (w # (SP_trans-mu))` * (SP_trans-mu) * (back/(back-1)) ;
sd_SP = sqrt(vecdiag(cov_SP))` ;
```

The interest rate and spread results are combined and labeled. The following code gives a report of the risk factor id, the current base value and the standard deviation:

```
cols_num_IR = {&risk_number_IR_num} ;
cols_num_SP = {&risk_number_SP_num} ;
cols_num_IR_SP = cols_num_IR // cols_num_SP ;
sd = sd_IR` // sd_SP` ;
dataAll_IR_SP = dataAll_IR[1,]` // dataAll_SP[1,]` ; ;
risk_factor_volatility = cols_num_IR_SP || dataAll_IR_SP || sd ;
rfv_colname ={"risk_factor_id" , "risk_factor_value" , "risk_factor_volatility"} ;
```

Below we calculate the covariance and correlation of the combined series.

```
IR_SP_trans = IR_trans || SP_trans ;
mu = w # IR_SP_trans ;
cov = (w # (IR_SP_trans-mu))` * (IR_SP_trans-mu) * (back/(back-1)) ;
nofac_IR_SP = nofac_IR + nofac_SP ;

dim = 0 ;
do i = nofac_IR_SP to 1 by -1 ;
  dim = i + dim ;
end ;
covcor = j(dim,5,.) ;
nofac_IR_SP = nofac_IR + nofac_SP ;

cor = j(nofac_IR_SP,nofac_IR_SP,1) ;
  do i = 1 to nofac_IR_SP-1 ;
    do j = i+1 to nofac_IR_SP ;
      cor[i,j] = cov[i,j] / (sqrt(cov[i,i] * cov[j,j])) ;
      cor[j,i] = cor[i,j] ;
    end ;
  end ;
  rec = 0 ;
  do i = 1 to nofac_IR_SP ;
    do j = 1 to i ;
      rec = rec + 1 ;
      covcor[rec,1] = cols_num_IR_SP[i] ;
      covcor[rec,2] = cols_num_IR_SP[j] ;
      covcor[rec,3] = cov[i,j] ;
      covcor[rec,4] = cor[i,j] ;
      covcor[rec,5] = rec      ;
    end ;
  end ;
```

VAR CALCULATION

As the elements of IML matrices are either all number or all character, we have to prepare the data in base SAS such that we can keep track of the variable names, etc. The names of the organization levels have to be converted into numbers. If this is a static list you can use proc format and use the number assignment to force the order of the reports or read the data into data step as in the example below:

```
data desktemp1 ;
  set sasdb.desk(keep=desk desk_sort_order process_group) ;
  where process_group = 'MVAR' ;
proc sort data = desktemp1 ;
  by desk ;
data desktemp2(drop=process_group) ;
  set desktemp1 ;
  by desk ;
  if last.desk ;
proc sort data = desktemp2 ;
  by desk_sort_order ;
data desk(rename=(desk=start) drop=process_group) ;
  retain fmtname '$deskfmt' ;
  set desktemp2 ;
  by desk notsorted ;
  if first.desk then do ;
    label + 1;
    output ;
  end ;
PROC format cntlin = desk fmtlib ;
```

The advantage of the having the data create the format is that the code doesn't have to be adjusted with each change in the database. Next we denormalized various database tables such that one matrix (mark_sql) holds risk factor and product id's and profiles, any information we'll need in to alter VaR values (currency ratio, liquidity adjustment), plus information we'll need for aggregation (business unit, asset class, etc.).

We want the report to appear as a tree structure, aggregating VaR results in a total, then at the company group level (GMAC, GM), then see each of these branch off into the various company in the groups, etc. We construct our reporting structure by merging a dataset with all the (numerical) permutations of the organization (business_group, company, etc.) with the desks in the mark_sql dataset, adding those numerical representations to create mark_val.

```
data primenum ;
  set sasdb.desk ;
  where process_group = 'MVAR' ;
      groupnum = input(put(company_group,$grpfmt.),best32.) ;
      conum = input(put(company,$compfmt.),best32.) ;
      busnum = input(put(business,$busfmt.),best32.) ;
      busunit = input(put(business_unit,$bnitfmt.),best32.) ;
      desknum = input(put(desk,$deskfmt.),best32.) ;
* proc print data = primenum ; * title 'primenum' ;
proc sort data = primenum out = prime_match ;
  by desk_id ;
data prime_mat(keep=desk_id groupnum conum busnum busunit desknum) ;
  set prime_match ;
  by desk_id ;
  if last.desk_id ;
proc sort data = sasdb.mark_sql ;
  by desk_id ;
data mark_val ;
  merge sasdb.mark_sql(in=a)
        prime_mat(in=b) ;
  by desk_id ;
  if a ;
```

We create a macro variable of product id's to use as a label for product level profit and loss matrix.

```
proc sort data = mark_val ;
  by product_id ;
data _null_ ;
%global prodcols ;
proc sql ;
select compress( tranwrd(tranwrd(tranwrd(tranwrd(upcase(tranwrd(tranwrd
       ( translate(left(trim(p.product_name)),'________','()&/-+ ')
          || '_' ||
         translate(left(trim(p.product_term)),'________','()&/-+ ')
          || '_' ||
         translate(left(trim(p.product_rating)),'________','()&/-+ ')
         || '_' ||
       put(p.product_id,z3.0),'__','_' ),'__','_' )),'_1YR_1YR','_1YR'),
         '_3YR_3YR','_3YR'),
         'RESIDUAL_FINANCING','RES_FIN'),
         'FIXED_RATE_LOANS','FIXRATLNS'))
  into :prodcols separated by ','
  from mark_val mv
      ,sasdb.product p
  where p.product_id = mv.product_id ;
quit ;
```

To flatten out the tree reporting structure into something we could use in a spread sheet we remove duplicates from five files, each representing a level of the organization, retaining the numbers that will appear in the mark_val matrix and concatonating the names for the report

```
data primerow0 ;
  length primecut $ 200 ;
  set primenum(keep=groupnum company_group desk_id) ;
  primecut = left(trim(company_group)) ;

data primerow1 ;
  length primecut $ 200 ;
  set primenum(keep=groupnum company_group conum company desk_id) ;
  primecut =  left(trim(company_group)) || '_' || left(trim(company)) ;
      ...
```

And merge the results:

```
data primerow4 ;
  length primecut $ 200 ;
  set primenum(keep=groupnum company_group conum company busnum business busunit business_unit
                       desknum desk desk_id) ;
  primecut = left(trim(company_group)) || '_' || left(trim(company))  || '_' ||
             left(trim(business)) || '_' || left(trim(business_unit)) || '_' || left(trim(desk)) ;

data sasdb.primerow ;
  length primecut $ 200 ;
  set primerowt ;
  by groupnum conum busnum busunit desknum ;
  if last.desknum ;

proc contents data = sasdb.primerow out = list(keep=nobs) noprint ;
data _null_ ;
  %global coln ;
  set list ;
  nobs = nobs + 1 ;
 call symput('coln','col' || left(trim(nobs))) ;  stop;
```

As SAS variable names are limited to 32 characters, we call label the levels of aggregation col1, col2, col3… We can link the label in the first row of the dataset, primerow, with col1, the label in the second row with col2, etc.

We read the covariance matrix, plus the interest rate information into IML:

```
use sasdb.mean_IR ;
    read all var{mean_IR} into mean_IR where (valuation_date = &curdat) ;

use sasdb.regu ;
    read all var{regu} into regut where(valuation_date = &curdat)  ;

regu = regut` ;

use sasdb.covcor where (valuation_date =         &curdat) ;
    read all var{risk_factor_id_row risk_factor_id_col cov cor} into covcor ;

dim_covcor = (-1+sqrt(1+(8*nrow(covcor)))) / 2 ;
* print dim_covcor ;

cov = j(dim_covcor,dim_covcor,.) ;
cor = j(dim_covcor,dim_covcor,.) ;
cov_cor_name = {&covcorlist} ;
rec = 0 ;
do i = 1 to dim_covcor ;
  do j = 1 to i ;
    rec = rec + 1 ;
    cov[i,j] = covcor[rec,3] ;
    cor[i,j] = covcor[rec,4] ;
    cov[j,i] = cov[i,j] ;
    cor[j,i] = cor[i,j] ;
  end ;
end ;
```

Singular Value Decomposition (SVD) is a function that allows the creation of a matrix of simulated values that has the covariance as specified. In our case &sims resolves to 15,000 (simiulations):

```
call svd(u , q , v ,cov) ;
  e = j(dim_covcor,&sims,100) ;
  e = normal(e) ;
  meanAll = j(&sims,dim_covcor,0) ;
  meanAll[,1:nrow(mean_IR)] = j(&sims,1,1) * mean_IR`;
  nrq = SQRT(diag(q)) * v` ;
  y = nrq` * e ;
  random = y` + meanAll ;
```

Because interest rate risk factors are assumed to have non-zero means, we use saved values of mean_IR and regu to calculate our risk factor simulations:

```
random[,1:nrow(mean_IR)] = 100 * (j(&sims,1,1) * IR[1,1:nrow(mean_IR)]) #
       (exp(random[,1:nrow(mean_IR)]) - (j(&sims,1,1) * exp(regu[1,1:nrow(mean_IR)]))) ;
  use sasdb.sd ;
    read all var{sd} into sd
      where(valuation_date=&curdat) ;
  randomsm = random / (j(&sims,1,1) * sd`) ;
```

We subtract the base values from the profile, but concatenate these to the mark_val matrix, as we'll use these for liquidity adjustments. We create two matrices to hold the results of the module which interpolates the simulated standardized risk factors with the products profiles (note that the dimensions of the matrix are number of simulations by the number of products).

```
start intp(mv, rsm, pl, scol, ecol, m) ;
  sigma_range = ecol - scol ;
  offset = scol + (sigma_range/2) ;

  low_sigma = -1 * (sigma_range/2) ;
  high_sigma = (sigma_range/2) ;
  sigmas = low_sigma ;
  do i = low_sigma + 1 to high_sigma ;
    sigmas = sigmas || i ;
  end ;
  do i = 1 to nrow(mv) ;
    if mv[i,m] > 0 then do ;
      p = ( (floor(rsm[,mv[i,m]]) <> low_sigma) >< (high_sigma -1) ) ;
      q = p + 1 ;
      d = rsm[,mv[i,m]] - p ;
      l = p + offset ;
      h = l + 1 ;
      pl[,i]  =  pl[,i] +
      (((mv[i,h]- mv[i,l]) # d`) + mv[i,l])` ;
    end;
  end;
finish ;

PL_IR = j(&sims,nrow(mark_val),0) ;
PL_SP = j(&sims,nrow(mark_val),0) ;

SigmaRange = {-3 -2 -1 0 1 2 3} ;

run intp(mark_val,randomsm,PL_IR,12,18,11) ;
run intp(mark_val,randomsm ,PL_SP,4,10,3) ;
```

The interpolation module was originally written using nested loops. The interpolation process is the most time intensive of the entire program; substituting matrix notation is some parts of the module reduced the time of the program by one third. Not only is matrix notation elegant and easy to follow, it often proves to be more efficient, but it consumes more memory.

We read the file that contains the primary business categories into a matrix (primerow), dimension the matrix that will hold the totaled profit and loss aggregations by these categories:

```
use sasdb.primerow ;
read all var{groupnum conum busnum busunit desknum}  into primenum ;
GMAC = j(&sims,nrow(primenum)+1,0) ;
GMAC_IR = j(&sims,nrow(primenum)+1,0) ;
GMAC_SP = j(&sims,nrow(primenum)+1,0) ;
```

The liquidity adjustment is calculated by dividing the sum of the base values of the products times the liquidity factor by the sum of the base values within each primary business category.

```
* group offset ;

    do j = 1 to nrow(primenum) ;
       if (primenum[j,1] = mark_val[i,22]
         & primenum[j,2] = .
         & primenum[j,3] = .
         & primenum[j,4] = .
         & primenum[j,5] = .)
       then do ;
         GMAC[,j+1] = GMAC[,j+1] + PL[,i] ;
         GMAC_IR[,j+1] = GMAC_IR[,j+1] + PL_IR[,i] ;
         GMAC_SP[,j+1] = GMAC_SP[,j+1] + PL_SP[,i] ;
         cut_rate[1,j+1] = cut_rate[1,j+1] +
                  (abs(mark_val[i,28]) * liquida) ;
         cut_rate[2,j+1] = cut_rate[2,j+1] +
                  (abs(mark_val[i,28])) ;
       end ;
    end ;

* company offset ;
    do j = 1 to nrow(primenum) ;
       if (primenum[j,1] = mark_val[i,22]
         & primenum[j,2] = mark_val[i,23]
         & primenum[j,3] = .
         & primenum[j,4] = .
         & primenum[j,5] = .)
       then do ;
         GMAC[,j+1] = GMAC[,j+1] + PL[,i] ;
         GMAC_IR[,j+1] = GMAC_IR[,j+1] + PL_IR[,i] ;
         GMAC_SP[,j+1] = GMAC_SP[,j+1] + PL_SP[,i] ;
         cut_rate[1,j+1] = cut_rate[1,j+1] + (abs(mark_val[i,28]) * liquida) ;
         cut_rate[2,j+1] = cut_rate[2,j+1] + (abs(mark_val[i,28])) ;
       end ;
     end ;
```

The profit and loss numbers are aggregated to the relevant business category by comparing the individual business level numbers in mark_val with the numbers in primenum.

Once we've aggregated all the products into their primary categories we can calculate the respective liquidity adjustment:

```
cut_rate[3,] = cut_rate[1,] / cut_rate[2,] ;
do i = 1 to ncol(cut_rate) ;
  if cut_rate[3,i] = 0
    then cut_rate[3,i] = 1 ;
end ;
```

Separate row matrices can be made for each confidence level using an IML module. These matrices can adjusted for liquidity by dividing multiplying by the liquidity ratio:

```
fifha = fifth(GMAC) ;
gmac9995 = fifha[4,] ;
gmac999 = fifha[3,] ;
gmac990 = fifha[2,] ;
gmac950 = fifha[1,] ;

agmac9995 = gmac9995 # cut_rate[3,] ;
agmac999 = gmac999 # cut_rate[3,] ;
agmac990 = gmac990 # cut_rate[3,] ;
agmac950 = gmac950 # cut_rate[3,] ;
```

REPORTING OF VAR RESULTS

The process is repeated for each month specified and the results appended to SAS data sets. Typically we want a 12 month run to evaluate VaR over time. The results are read in IML, transposed and the relevant names added as row names.

CONCLUSION

SAS/IML is a convenient and powerful language that allows for writing customized statistics and rapid development. The improvement in performance over the original VBA model was significant: the old model required 40 minutes for 3,000 simulations for a given month; the IML model takes 10 minutes for 15,000 simulations.

REFERENCES

Philippe Jorion, *Value at Risk*, 2nd edition.

J.P. Morgan's *Risk Metrics – Technical Document* available on the internet at http://www.jpmorgan.com/RiskManagement/RiskMetrics/RiskMetrics.html

Rezek, G., (2003) "A System for Calculating Value-at-Risk using Oracle® and Monte Carlo Simulation in SAS/IML®. *Proceedings of the Sixteenth Annual Conference of the Northeastern SAS Users Group*

ACKNOWLEDGMENTS

Special thanks to Serena Tiong who was very helpful in just about every aspect of the development of this model.

CONTACT INFORMATION

George Rezek
GMAC Enterprise Risk Management
8400 Normandale Lake Blvd
Bloomington, MN 55437
Work phone: (952) 857-6042
Fax: (952) 857-6688
email: george.rezek@gmacerm.com

Appendix A

GMAC Two-Factor Value-at-Risk Model

Correlation of Risk Factors

Risk Factor	Gov Treas 5yr	AA Loan	10yr Swap
Gov Treas 5yr	1.000	-0.145	0.194
AA Loan	-0.145	1.000	0.572
10yr Swap	0.194	0.572	1.000

Covariance of Risk Factors

Risk Factor	Gov Treas 5yr	AA Loan	10yr Swap
Gov Treas 5yr	0.005	-0.136	0.130
AA Loan	-0.136	192.415	78.837
10yr Swap	0.130	78.837	98.713

Singular Value Decomposition

Simulated Risk Factor Values

RISK FACTOR	Gov Treas 5yr	COM AA	10yr Swap
Simulation 1	0.03	16.20	8.35
Simulation 2	-0.04	2.69	0.72
Simulation 3	0.07	26.82	6.43
Simulation 4	0.00	5.68	13.59
Simulation 5	-0.07	0.29	-4.02
Simulation 6	-0.08	-2.66	-4.21
Simulation 7	-0.04	-10.22	8.94
Simulation 8	0.08	-1.99	-1.51
Simulation 9	0.03	21.57	13.07
Simulation 10	0.01	-7.32	-3.21

Standardized Risk Factor Values

Risk Factor	Gov Treas 5yr	COM AA	10yr Swap
Simulation 1	0.54	1.17	0.84
Simulation 2	-0.53	0.19	0.07
Simulation 3	1.20	1.93	0.65
Simulation 4	0.11	0.41	1.37
Simulation 5	-0.98	0.02	-0.40
Simulation 6	-1.02	-0.19	-0.42
Simulation 7	-0.46	-0.74	0.90
Simulation 8	1.31	-0.14	-0.15
Simulation 9	0.57	1.56	1.32
Simulation 10	0.20	-0.53	-0.32

History of Risk Factors

Year-Month	Gov Treas 5yr	AA Loan	10yr Swap
2002M04	4.405	113.500	54.000
2002M03	4.838	124.300	66.500
2002M02	4.189	130.600	66.500
2002M01	4.369	136.900	72.000
2001M12	4.300	153.800	76.500
2001M11	4.062	148.700	69.650
2001M10	3.473	146.700	66.670
2001M09	3.802	146.700	66.670
2001M08	4.378	141.000	81.710
2001M07	4.525	141.600	80.160
2001M06	4.949	151.000	90.180
2001M05	4.949	141.300	77.710
2001M04	4.949	148.800	81.370
2001M03	4.949	174.900	93.140
2001M02	4.949	154.200	89.330
2001M01	4.949	143.400	79.390
2000M12	4.949	163.000	102.140
2000M11	4.949	171.100	118.100
2000M10	4.949	169.000	112.100
2000M09	4.949	164.000	111.450
2000M08	4.949	182.000	127.250
2000M07	4.949	175.000	122.500
2000M06	4.949	179.500	123.300
2000M05	4.949	193.000	136.900
2000M04	4.949	179.000	119.000
2000M03	4.949	175.000	125.750
2000M02	4.949	163.000	102.100
2000M01	4.949	140.000	85.550
1999M12	4.949	144.000	79.000
1999M11	4.949	143.000	79.000
1999M10	4.949	155.000	87.000
1999M09	4.949	163.000	97.000
1999M08	4.949	179.000	108.000
1999M07	4.949	167.000	100.000
1999M06	4.949	144.000	83.500
1999M05	4.949	138.000	79.500
1999M04	4.949	134.000	71.000
1999M03	4.949	137.000	79.000
1999M02	4.949	137.000	74.000
1999M01	4.949	145.000	75.000
1998M12	4.949	161.000	76.000
1998M11	4.949	155.000	80.500
1998M10	4.949	210.000	77.000
1998M09	4.949	165.000	84.500
1998M08	4.949	123.000	85.500
1998M07	4.949	98.000	55.500
1998M06	4.949	96.000	55.500
1998M05	4.949	90.000	53.500
1998M04	4.949	90.000	49.500
1998M03	4.949	95.000	51.500
1998M02	4.949	90.000	48.500
1998M01	4.949	90.000	48.500
1997M12	4.949	90.000	47.500
1997M11	4.949	90.000	45.500
1997M10	4.949	88.000	53.500
1997M09	4.949	70.000	44.500
1997M08	4.949	67.000	42.500
1997M07	4.949	67.000	40.500
1997M06	4.949	67.000	38.500
1997M05	4.949	70.000	35.500

Compute 's for Interest Rate and Spread

	Current Month	+1 σ	+2 σ	+3 σ
Gov Treas 5yr	4.405	29.674	59.348	89.023
COM AA	113.500	13.871	27.743	41.614
Swap 10yr	54.000	9.936	19.871	29.807

Interest Rate

Spread

Traders use volatility reports to predict market values of their portfolios corresponding to -3, -2, -1, 0, +1, +2 & +3 σ movement of risk factors

Interest Rate Shocks

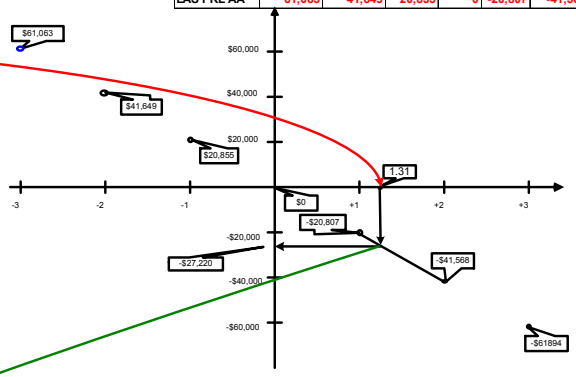
Product	-3σ	-2σ	-1σ	Base Value	+1σ	+2σ	+3σ
LAS FRL AA	61,063	41,649	20,855	0	-20,807	-41,568	-61,894
LAS ARML AA	46	31	16	0	-17	-34	-53
Sec Agency	-2,011,065	-1,340,709	-670,354	0	670,356	1,340,712	2,011,067

Spread Shocks

Product	-3σ	-2σ	-1σ	Base Value	+1σ	+2σ	+3σ
LAS FRL AA	61,980	41,376	20,711	0	-20,829	-41,668	-62,387
LAS ARML AA	-13,123	-8,744	-4,369	0	-11,381	-7,012	-2,649
Sec Agency	-858,826	-572,550	-286,275	0	286,276	572,552	858,827

Example of Straight-Line Interpolation

Product	-3 σ	-2 σ	-1σ	Base Value	+1σ	+2 σ	+3 σ
LAS FRL AA	61,063	41,649	20,855	0	-20,807	-41,568	-61,894



Interest Rate	LAS FRL AA	LAS ARML AA	Sec Agency
Simulation 1	-11,182	-9	360,270
Simulation 2	11,012	8	-353,964
Simulation 3	-25,026	-20	806,582
Simulation 4	-2,251	-2	72,531
Simulation 5	20,341	16	-653,818
Simulation 6	21,287	16	-684,284
Simulation 7	-9,508	7	-305,632
Simulation 8	-27,220	-22	877,428
Simulation 9	-11,866	-10	382,307
Simulation 10	-4,201	-3	135,341

+

Spread	LAS FRL AA	LAS ARML AA	Sec Agency
Simulation 1	-24,323	-10,648	240,527
Simulation 2	-4,041	-2,208	20,793
Simulation 3	-40,280	-7,303	185,304
Simulation 4	-8,530	-4,661	391,543
Simulation 5	-430	-235	-115,826
Simulation 6	3,966	-837	-121,256
Simulation 7	15,257	-3,219	257,699
Simulation 8	2,964	-625	-43,636
Simulation 9	-32,400	-8,955	376,694
Simulation 10	10,931	-2,306	-92,622

=

Interest Rate & Spread	LAS FRL AA	LAS ARML AA	Sec Agency
Simulation 1	-35,505	-10,657	600,797
Simulation 2	6,971	-2,199	-333,170
Simulation 3	-65,306	-7,324	991,886
Simulation 4	-10,781	-4,662	464,074
Simulation 5	19,911	-219	-769,644
Simulation 6	25,253	-820	-805,539
Simulation 7	24,765	-3,211	-47,932
Simulation 8	-24,256	-647	833,791
Simulation 9	-44,266	-8,965	759,002
Simulation 10	6,730	-2,309	42,719

Sum the simulations across all products

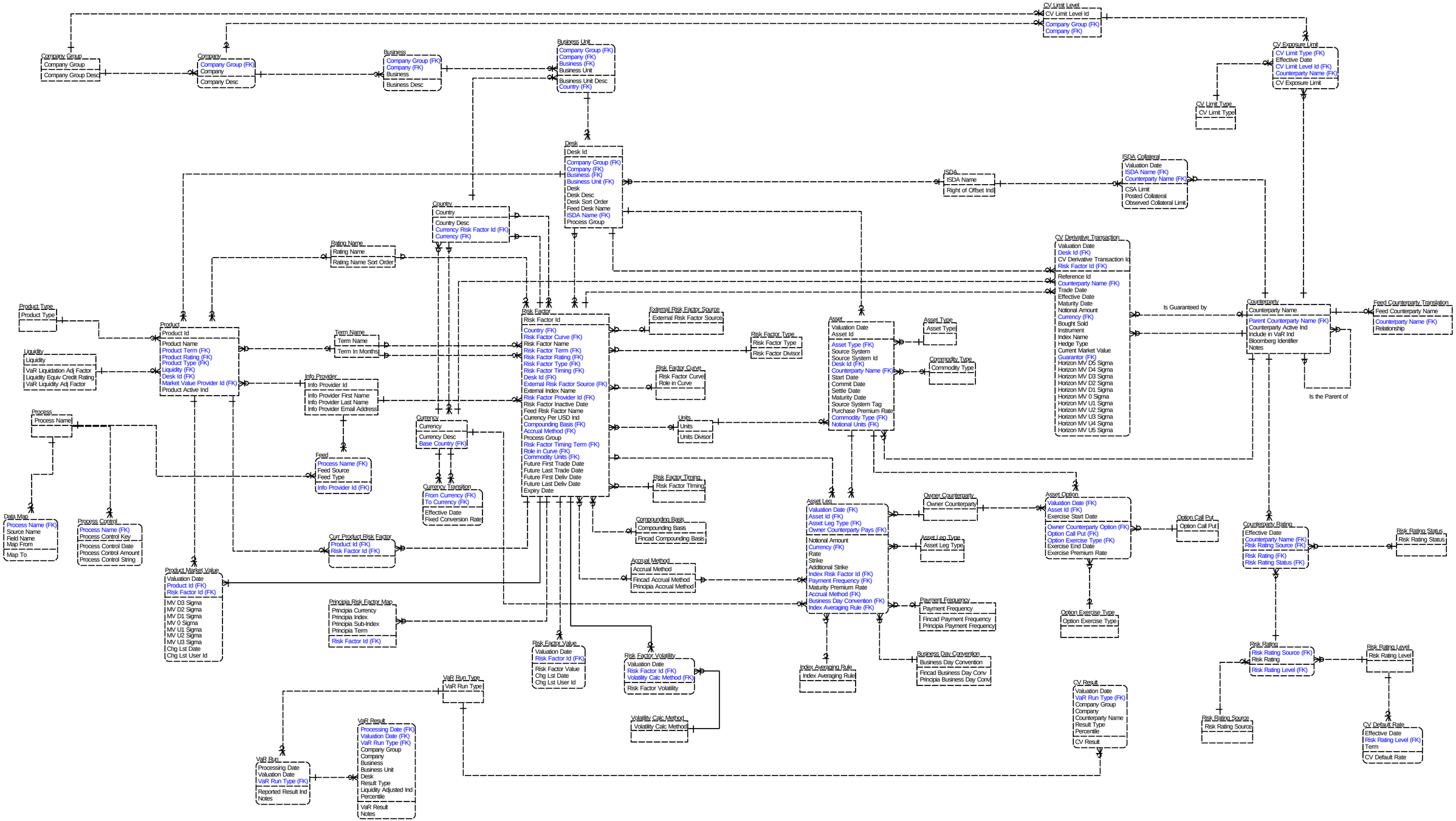
Interest Rate & Spread	LAS FRL AA		LAS ARML AA		Sec Agency		Sum of All Products
Simulation 1	-35,505	+	-10,657	+	600,797	=	554,634
Simulation 2	6,971	+	-2,199	+	-333,170	=	-328,398
Simulation 3	-65,306	+	-7,324	+	991,886	=	919,257
Simulation 4	-10,781	+	-4,662	+	464,074	=	448,630
Simulation 5	19,911	+	-219	+	-769,644	=	-749,952
Simulation 6	25,253	+	-820	+	-805,539	=	-781,106
Simulation 7	24,765	+	-3,211	+	-47,932	=	-26,378
Simulation 8	-24,256	+	-647	+	833,791	=	808,888
Simulation 9	-44,266	+	-8,965	+	759,002	=	705,771
Simulation 10	6,730	+	-2,309	+	42,719	=	47,140

Sort Values

Sum of All Products (Sorted)
919,257
808,888
705,771
554,634
448,630
47,140
-26,378
-328,398
-749,952
-781,106

Determine VaR at 95% Confidence Interval

George Rezek
george.rezek@gmacerm.com


CV Limit Level
CV Limit Level Id
Company Group (FK)
Company (FK)
Company Group
Company Group
Company Group Desc
Company
Company Group (FK)
Company
Company Desc
Business
Company Group (FK)
Company (FK)
Business
Business Desc
Business Unit
Company Group (FK)
Company (FK)
Business (FK)
Business Unit
Business Unit Desc
Country (FK)
CV Exposure Limit
CV Limit Type (FK)
Effective Date
CV Limit Level Id (FK)
Counterparty Name (FK)
CV Exposure Limit
CV Limit Type
CV Limit Type
Desk
Desk Id
Company Group (FK)
Company (FK)
Business (FK)
Business Unit (FK)
Desk
Desk Desc
Desk Sort Order
Feed Desk Name
ISDA Name (FK)
Process Group
ISDA
ISDA Name
Right of Offset Ind
ISDA Collateral
Valuation Date
ISDA Name (FK)
Counterparty Name (FK)
CSA Limit
Posted Collateral
Observed Collateral Limit
Country
Country
Country Desc
Currency Risk Factor Id (FK)
Currency (FK)
Rating Name
Rating Name
Rating Name Sort Order
CV Derivative Transaction
Valuation Date
Desk Id (FK)
CV Derivative Transaction Id
Risk Factor Id (FK)
Reference Id
Counterparty Name (FK)
Trade Date
Effective Date
Maturity Date
Notional Amount
Currency (FK)
Bought Sold
Instrument
Index Name
Hedge Type
Current Market Value
Guarantor (FK)
Horizon MV D5 Sigma
Horizon MV D4 Sigma
Horizon MV D3 Sigma
Horizon MV D2 Sigma
Horizon MV D1 Sigma
Horizon MV 0 Sigma
Horizon MV U1 Sigma
Horizon MV U2 Sigma
Horizon MV U3 Sigma
Horizon MV U4 Sigma
Horizon MV U5 Sigma
Is Guaranteed by
Counterparty
Counterparty Name
Parent Counterparty Name (FK)
Counterparty Active Ind
Include in VaR Ind
Bloomberg Identifier
Notes
Is the Parent of
Feed Counterparty Translation
Feed Counterparty Name
Counterparty Name (FK)
Relationship
Product Type
Product Type
Product
Product Id
Product Name
Product Term (FK)
Product Rating (FK)
Product Type (FK)
Liquidity (FK)
Desk Id (FK)
Market Value Provider Id (FK)
Product Active Ind
Liquidity
Liquidity
VaR Liquidation Adj Factor
Liquidity Equiv Credit Rating
VaR Liquidity Adj Factor
Term Name
Term Name
Term In Months
Info Provider
Info Provider Id
Info Provider First Name
Info Provider Last Name
Info Provider Email Address
Risk Factor
Risk Factor Id
Country (FK)
Risk Factor Curve (FK)
Risk Factor Name
Risk Factor Term (FK)
Risk Factor Rating (FK)
Risk Factor Type (FK)
Risk Factor Timing (FK)
Desk Id (FK)
External Risk Factor Source (FK)
External Index Name
Risk Factor Provider Id (FK)
Risk Factor Inactive Date
Feed Risk Factor Name
Currency Per USD Ind
Compounding Basis (FK)
Accrual Method (FK)
Process Group
Risk Factor Timing Term (FK)
Role in Curve (FK)
Commodity Units (FK)
Future First Trade Date
Future Last Trade Date
Future First Deliv Date
Future Last Deliv Date
Expiry Date
External Risk Factor Source
External Risk Factor Source
Risk Factor Type
Risk Factor Type
Risk Factor Divisor
Risk Factor Curve
Risk Factor Curve
Role in Curve
Units
Units
Units Divisor
Asset
Valuation Date
Asset Id
Asset Type (FK)
Source System
Source System Id
Desk Id (FK)
Counterparty Name (FK)
Start Date
Commit Date
Settle Date
Maturity Date
Source System Tag
Purchase Premium Rate
Commodity Type (FK)
Notional Units (FK)
Asset Type
Asset Type
Commodity Type
Commodity Type
Process
Process Name
Data Map
Process Name (FK)
Source Name
Field Name
Map From
Map To
Process Control
Process Name (FK)
Process Control Key
Process Control Date
Process Control Amount
Process Control String
Feed
Process Name (FK)
Feed Source
Feed Type
Info Provider Id (FK)
Currency
Currency
Currency Desc
Base Country (FK)
Currency Translation
From Currency (FK)
To Currency (FK)
Effective Date
Fixed Conversion Rate
Curr Product Risk Factor
Product Id (FK)
Risk Factor Id (FK)
Risk Factor Timing
Risk Factor Timing
Compounding Basis
Compounding Basis
Fincad Compounding Basis
Accrual Method
Accrual Method
Fincad Accrual Method
Principia Accrual Method
Product Market Value
Valuation Date
Product Id (FK)
Risk Factor Id (FK)
MV D3 Sigma
MV D2 Sigma
MV D1 Sigma
MV 0 Sigma
MV U1 Sigma
MV U2 Sigma
MV U3 Sigma
Chg Lst Date
Chg Lst User Id
Principia Risk Factor Map
Principia Currency
Principia Index
Principia Sub-Index
Principia Term
Risk Factor Id (FK)
Risk Factor Value
Valuation Date
Risk Factor Id (FK)
Risk Factor Value
Chg Lst Date
Chg Lst User Id
Risk Factor Volatility
Valuation Date
Risk Factor Id (FK)
Volatility Calc Method (FK)
Risk Factor Volatility
Volatility Calc Method
Volatility Calc Method
VaR Run Type
VaR Run Type
VaR Run
Processing Date
Valuation Date
VaR Run Type (FK)
Reported Result Ind
Notes
VaR Result
Processing Date (FK)
Valuation Date (FK)
VaR Run Type (FK)
Company Group
Company
Business
Business Unit
Desk
Result Type
Liquidity Adjusted Ind
Percentile
VaR Result
Notes
Asset Leg
Valuation Date (FK)
Asset Id (FK)
Asset Leg Type (FK)
Owner Counterparty Pays (FK)
Notional Amount
Currency (FK)
Rate
Strike
Additional Strike
Index Risk Factor Id (FK)
Payment Frequency (FK)
Maturity Premium Rate
Accrual Method (FK)
Business Day Convention (FK)
Index Averaging Rule (FK)
Owner Counterparty
Owner Counterparty
Asset Leg Type
Asset Leg Type
Payment Frequency
Payment Frequency
Fincad Payment Frequency
Principia Payment Frequency
Business Day Convention
Business Day Convention
Fincad Business Day Conv
Principia Business Day Conv
Index Averaging Rule
Index Averaging Rule
Asset Option
Valuation Date (FK)
Asset Id (FK)
Exercise Start Date
Owner Counterparty Option (FK)
Option Call Put (FK)
Option Exercise Type (FK)
Exercise End Date
Exercise Premium Rate
Option Exercise Type
Option Exercise Type
Option Call Put
Option Call Put
CV Result
Valuation Date
VaR Run Type (FK)
Company Group
Company
Counterparty Name
Result Type
Percentile
CV Result
Counterparty Rating
Effective Date
Counterparty Name (FK)
Risk Rating Source (FK)
Risk Rating (FK)
Risk Rating Status (FK)
Risk Rating Status
Risk Rating Status
Risk Rating
Risk Rating Source (FK)
Risk Rating
Risk Rating Level (FK)
Risk Rating Level
Risk Rating Level
Risk Rating Source
Risk Rating Source
CV Default Rate
Effective Date
Risk Rating Level (FK)
Term
CV Default Rate