Paper 353-2011

Scenario and Stress Testing using SAS/IML®

Dmitry Donin, Dmytro Korol, Mykola Tkachenko*
Bank of Montreal, Toronto, Ontario

Abstract

The proposed paper presents a methodology for Scenario and Stress testing of a model based on defining and determining quantiles of the joint distribution of model's variables. The implementation of the methodology in the SAS® system gives a view on rarely used SAS® capabilities for dealing with quite sophisticated algorithms and uses Interactive Matrix Language (IML) available with SAS®. The methodology can serve as a non-parametric method to determining extremes of a model, when the distribution of model variables is unknown. The technique we propose can be used in wide range of problems related to the analysis of multivariate data.

1 Introduction

After an empirical model has been developed using some data set (for example, a financial forecast model), the question immediately arises about how sensitive the model is to extreme events. By 'extreme event' we mean an event that has not been observed, but that could realistically be expected to occur with some low probability. For example, in financial modeling, this question is equivalent to testing a model's sensitivity to downturn events that are considered extreme. The question would have a complete answer if we knew the distribution function from which the development data is taken.

*At the time of publication of this paper the second and the third authors are not employees of Bank of Montreal.

The extreme scenarios would be the tail events of this distribution. Thus, if we want to test a model on events (scenarios) at a level corresponding to, for example, 95-th quantile, we would take the 95-th quantile set (called the quantile contour), restrict our model to this set and take its maximum or minimum values. These values will represent extremes of the model on the events which occur with probability 5 percent.

This 'unified' framework of Stress testing (when applied to financial modeling problems) was proposed by Berkowitz (2004) (see also McNeil, Smith (2010)).

Several problems are associated with such approach. First, the joint distribution function of the model variables is usually unknown and one is forced to fit distribution to the data (parametric approach) which usually leads to uncontrolled errors. Second, because there is no natural ordering in the space of dimension greater than one, the notion of a quantile can be ambiguous. Another problem is that the model under consideration may be quite complex (for example non-linear). For such a model it may happen that extreme points in the sense of their distribution are not extreme for the model and the values of the model on these points are less (or greater, depending on the context), than, say, its average value, making the results meaningless.

In what follows we make use of a notion of half-space depth, also known as Tukey depth or location depth (cf. Miller et al.(2003)) to approach the notion of a quantile of a distribution. The half-space depth of a point x relative to a multivariate data set first occurred as a test statistic of Hodges (1955) for the hypothesis that x is the center of probability distribution from which the data was drawn. Half-space depth gives a way to partially "order" points in dimensions greater than two, allowing to consider points of equal depth as the level curves corresponding to some probability. We then restrict our model to these levels and find its extremes.

With dimension growth an optimization problems (like finding extremes of a model) become very complex and require certain non-trivial algorithms. In what follows we make use of two algorithms: the Genetic algorithm (cf. Goldberg (1989)) implemented in SAS/IML® and a variation of the Shake-and-Bake algorithm (cf. Boender et al. (1991)). We give a detailed explanation of how to implement them using SAS/IML® in the context of our problem.

Because the nature of problems related to multivariate distributions are

often very complex, throughout our paper we make certain assumptions under which these problems become more tractable. In particular we restrict ourselves to consideration of only linear models (which are, for example, the results of the regression analysis).

2 Description of the methodology

We start with the definition of the main tool used in our analysis – the notions of half-space depth relative to a data set. For reference regarding the subject please refer to Miller et al.(2003), McNeil, Smith (2010).

Definition: Let Z be a finite set of data points in (real) n-dimensional space. The half-space depth of a point x relative to Z is the smallest number of points of Z lying in any closed half-space determined by a hyper-plane through x:

$$\min_{a \in \mathbf{R}^n \setminus \{0\}} |\{z \in Z \mid \langle y, z \rangle \leq \langle x, z \rangle\}|,$$

where y is the outward normal vector of the closed half-space.

For a fixed positive integer k the set of points with location depth k is a convex polygonal region. We will call its boundary the k-th depth contour and denote it by C_k.

Note that if the data set contains m points, then the k-th contour is the boundary of the intersection of all half-spaces containing $m - k + 1$ data points.

The concept of half-space depth was introduced by Hodges (1955) as a way to extend the univariate notion of ranking to a bivariate configuration of data points. It was used for robust estimation, hypothesis testing, and graphical display.

Below we use the depth contours as a proxy for the quantiles of multivariate distribution of the data. Thus, finding the extremes of a restriction of the model under consideration to the p-th depth contour will result in the p-probable point of model's extremes.

In the implementation of the methodology below we make use of the Genetic algorithm to find a proxy of the depth of a point together with the corresponding boundary hyper-plane. The collection of these hyper-planes forms a (bounded) polyhedra, representing the corresponding depth contour. In order to find extremes of the model under consideration on this contour, we

make use of a variation of Shake-and-Bake algorithm to uniformly generate a sufficiently large set of points on the contour (cf. Boender et al. (1991)) and then find the model's extremes on this set.

The complexity of the above algorithms can be found in SAS/IML® User Guide and Boender et al. (1991).

3 Implementation into the SAS® system

In this section we present a code demonstrating the described methodology.

We start with a data set with variables correlated by a matrix s. For simplicity we assume the data set consists of two variables.

We can simulate the data set with the help of 'proc model' implemented in SAS®. In the code below the data set 'main' is generated with two variables 'aa', 'ab' distributed uniformly on the unit segment with correlation matrix

$$s = \begin{pmatrix} 1 & 0.4 \\ 0.4 & 1 \end{pmatrix}.$$

Note that instead of uniform distribution one could have taken one of the following distributions implemented in 'proc model': Normal, t, F, Cauchy, Chisquare, Poisson.

```
data s;
   _NAME_ = "aa";
   aa = 1; ab = 0.4;
   output;
   _NAME_ = "ab";
   aa = 0.4; ab = 1;
   output;
run;
data all ;
   y = 0 ;
   z = 0 ;
   output ;
run ;

%let nsim = 10;
proc model data=all ;
   aa = 0;
   ab = 0;
```

```
   errormodel aa ~ uniform(0,1);
   errormodel ab ~ uniform(0,1);
   solve aa ab  / random = &nsim  sdata = s
   out = sim_uni(where=(_rep_>0)) ;
run;

data main (keep = aa ab);
set sim_uni;
run;
```

For convenience we keep the variable names and their count. The data set 'varName' created below contains the variable names, which are stored separately in the corresponding macro variables 'vari', $i = 1, 2$:

```
proc contents data=main
out = varName (keep = name type varnum);
run;

proc sort data=varName out=varName;
by varnum;

proc sql;
create table varNum as select sum(1) as rows from varName;

data _NULL_;
set varNum;
call symputx('nvars',rows);
run;

data newVar;
set varName;
call symputx('var'||left(_n_),name);
run;
```

The following macro 'cVar' removes the records with missing variables and writes the variables in the data set 'constVar':

```
%macro cVar;
data constVar;
set main;
%do i = 1 %to &nvars;
```

```
   if &&var&i ^= . then
%end;
   output;
run;
%mend cVar;
```

The macro 'constraint' in the following code results in a data set 'indVar', containing the scaled ordinal values (ordinal values between 0 and 1) of the variables from the data set 'constVar' without allowing repetitions:

```
%macro constraint;
data tempVar;
set constVar;
run;
%do i = 1 %to &nvars;
      proc sort data = tempVar;
      by &&var&i.;
      data tempVar (drop = ret prevCount);
      retain ret prevCount;
      set tempVar;
      if _n_ ^= 1 then
        if &&var&i. = ret then
          ind&&var&i. = prevCount;
                    else
          ind&&var&i. = prevCount+1;
      else
        ind&&var&i. = 1;
      ret = &&var&i.;
      prevCount = ind&&var&i.;
      run;

      proc sql;
      create table distvar&i. as select distinct(&&var&i.)
        as &&var&i. from constVar;

        proc sort data=distvar&i.;
      by &&var&i.;
%end;
proc sql;
create table indVar as select ind&var1/max(ind&var1) as &var1
from tempVar;
quit;
```

```
%mend constraint;
```

We now make use of SAS/IML® in order to calculate the half-space depth contours relative to the dataset 'indVar'. The code below results in a vector 'maxVarCounts' of the length equal to the number of variables, and with the entries equal to the number of distinct records for each variable respectively:

```
use indVar;
read all into rankedVariables[colname=name];
numberRecords = nrow(rankedVariables);
%do i = 1 %to &nvars;
    use distvar&i.;
    read all into distinctVarValue&i[colname=&&var&i.];
    maxVarCounts = maxVarCounts||nrow(distinctVarValue&i);
%end;
```

Let us show how to calculate the half-space depth of a point relative to the data set. The first function, 'halfspaceCount', in the following code calculates the number of points to one side of a given plane through a point. The function 'halfspaceCount' has one local variable 'planeCoeffs' representing the coefficients determining the plane, and four global variables: rankedVariables, numberRecords, maxVarCounts, pointForDepth.

```
start halfspaceCount (planeCoeffs) global (rankedVariables,
    numberRecords, maxVarCounts, pointForDepth);
  if (planeCoeffs*planeCoeffs`) = 0 then
    return (numberRecords);
  constant=(pointForDepth / maxVarCounts)*planeCoeffs`;
  diff = rankedVariables*planeCoeffs`
           - J(numberRecords,1,constant);
  halfspaceCount = 0;
  halfspaceCount = ncol(loc(abs(diff)+diff))+(numberRecords
                     - ncol(loc(diff)));
  return (halfspaceCount);
finish halfspaceCount;
```

The function 'halfspaceDepth' below minimizes 'halfspaceCount' using the Genetic Algorithm, implemented in SAS/IML®, thus finding the proxy

for the half-space depth of a point. Once the depth of the point is found, the function stores the coefficients of the corresponding plane in 'bestPlaneCoeffs (normalized coefficients) and 'constantForPlane' (free terms).

```
start halfspaceDepth(point) global(rankedVariables,
    numberRecords, maxVarCounts, pointForDepth,
    bestPlaneCoeffs, constantForPlane);
  pointForDepth = point;
  id = gasetup(1, &nvars,0);
  call gasetobj(id, 0, "halfspaceCount");
  call gasetcro(id, 1 ,1);
  call gasetmut(id, 0.05, 1);
  call gasetsel(id, 10, 1, 1);
  call gainit(id, 100, J(1,&nvars,-1)//J(1,&nvars,1));
  iterations = 1;
  do i = 2 to iterations;
     call garegen(id);
  end;
  call gagetmem(bestMember, depth, id, 1);
  bestPlaneCoeffs = bestMember/sqrt(ssq(bestMember));
  constantForPlane = (pointForDepth / maxVarCounts)*
     bestPlaneCoeffs';
  return (depth/numberRecords);
finish halfspaceDepth;
```

Let us assume we are interested in the extremes of a model when restricted to a depth contour for some value p, which we will keep in a macro variable 'qtile'. We look for all the points in the data set (more precisely, their row indices) whose depth is within 10% relative error of 'qtile' (we keep them in 'equiDepth') together with the corresponding planes defining the depth contours (we keep them in 'planes' and 'constants').

```
do i = 1 to numberRecords;
  if abs(&qtile -halfspaceDepth(ceil(rankedVariables[i,]#
     maxVarCounts))) < 0.02 then do;
    equiDepth = equiDepth//i;
    planes = planes//bestPlaneCoeffs;
    constants = constants//constantForPlane;
  end;
end;
```

We assume that the model, defined on our data set is represent in SAS® as a function 'pModel' of the variables in the data set, returning a real value.

The following code implements a variation of the Shake-and-Bake algorithm (Boender et al. (1991)), which uniformly populates points on the depth contour constructed in the previous step (here we populate 1000 points). We evaluate the model on each of the populated points and print the resulting maximum value. This maximum is the extreme model's value on the quantile represented by the depth contour.

```
do i = 1 to &nvars;
  k = J(1,&nvars,0);
  k[i] = 1;
  planes = planes//k;
  constants = constants//1;
    k[i] = -1;
  planes = planes//k;
  constants = constants//0;
end;
print equiDepth;
x = rankedVariables[equiDepth[1],];
c = 1;
do i = 1 to 1000;
  u = normal(repeat(0,1,&nvars));
  do while (ssq(u) = 0);
    u = normal(repeat(0,1,&nvars));
  end;
  u = u / sqrt(ssq(u));
  r = uniform(0);
  v = r/sqrt(1-(planes[c,]*u`)**2)*u-(r*(planes[c,]*u`)/sqrt
     (1-(planes[c,]*u`)**2)+sqrt(1-
             r**2))*planes[c,];
  l =  (constants - planes*x`)/(planes*v`);
  ind = 1;
  do j = 2 to nrow(l);
    if (l[ind] <= 10**-14 | (l[j] < l[ind] & l[j] > 10**-14))
        then
      ind = j;
  end;
  c = ind;
  totx = (x||pModel(ceil(x#maxVarCounts)))//totx;
  x = x + l[ind]*v;
end;
call sortndx(ind,totx,{%eval(&nvars+1)},{%eval(&nvars+1)});
print "The highest Model's value found is:"
```

```
(pModel(ceil(totx[ind[1],1:&nvars]#maxVarCounts)));
```

To visualize the results of the Shake-and-Bake algorithm one can make use of the following graphical mode:

```
call gstart;
call gwindow({-1 -1  2 2});
call gpoint(rankedVariables[,1],rankedVariables[,2],"dot","
   black");
call gpoint(totx[,1],totx[,2],"dot","green");
call gpoint(rankedVariables[equiDepth,1],
            rankedVariables[equiDepth,2],"dot","red");
call gshow;
call gstop;
```

For example, the data set of 100 records ('nsim=100') with the 'qtile = 0.05' results in the following visualization of the depth contour:

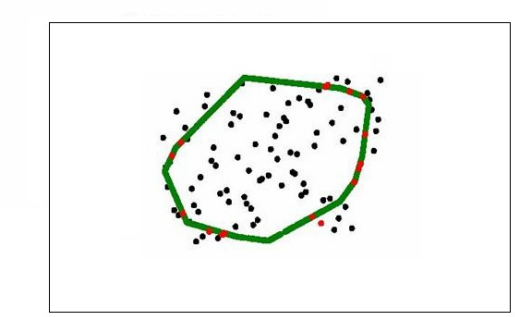

Figure 1: 5-th depth contour

The black dots on this figure are the data set points, and the red points are those data set points which lie on the corresponding depth contour. The green dots are the points on the depth contour, generated by the Shake-and-Bake algorithm.

4 Conclusion

In the paper we discussed a methodology for Scenario and Stress testing of a model based on non-parametric estimation of quantiles of the distribution of model's variables. The methodology can serve as a unified approach to Stress testing and can be further improved depending on the model specifications to obtain even more accurate results. The realization of the methodology is done with the help of IML procedure available with SAS®, which serves as the perfect environment for implementation of very sophisticated algorithms.

References

J. Berkowitz (2004), *A coherent framework for stress-testing*, In Jorion, P., editor, Innovations in Risk Management: Seminal Papers from the Journal of Risk. Risk Books.

D. E. Goldberg (1989), *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Professional; 1 edition.

J. L. Hodges (1955), *A bivariate sign test*, The Annals of Mathematical Statistics 26:523-527.

A. J. McNeil, A. D. Smith (2010), *Multivariate Stress Testing* for Solvency II.

K. Miller, S. Ramaswami, P. Rousseeuw, J. A. Sellares, D. Souvaine, I. Streinu, A. Struyf (2003), *Efficient computation of location depth contours by methods of computational geometry*, Statistics and Computing 13:153-162.

C. G. E. Boender, R. J. Caron, J. F. McDonald, A. H. G. Rinnooy Kan, H. E. Romeijn, R. L. Smith, J. Telgen, A. C. F. Vorst (1991), *Shake-and-bake algorithms for generating uniform points on the boundary of bounded polyhedra*, Operations Research, Volume 39, Issue 6, 945-954

Contact Information

Dmitry Donin
Bank of Montreal
55 Bloor Street West 10th Floor
Toronto, ON M4W3N5
Tel.: 1-416-927-5718
Email: Dmitry.Donin@bmo.com