

What Do You Mean You Don't Have SAS!!!?? Using the Power of the SAS Data Step to Create Cluster Scoring Code That Is Not Platform Dependent

Bill Hartley, M.S., Euro RSCG 4D Discovery, Glen Allen, VA

ABSTRACT

When transferring code, we can't assume all clients have access to SAS. Some of the statistical procedures that are easily accessible in SAS/STAT can't be used in all environments. Statistical procedures, such as mean, standard deviation, minimums, maximums, etc. are found in many software platforms. Even regression is not difficult to code since it is a linear function.

Cluster analysis is more difficult to code for outside platforms. This paper demonstrates how the data step can be utilized to create scoring code for cluster analysis, which can be transferred to software independent environments.

INTRODUCTION

A lot of time and manpower are needed to derive a cluster solution - from the beginning phases of exploratory data analysis to variable reduction, to a final cluster solution. Once the final cluster solution is completed, it is presented to the client with the accompanying SAS code so that the client may score the file.

Although many analysis companies use SAS extensively, their clients may not. Some clients may not have SAS/STAT as part of their licensing agreement. Therefore, it is up to the contracted analysis company to provide code that can be easily adapted to work on the client's system.

This paper provides an example of using SAS to create a cluster scoring code without using the statistical procedures provided by SAS/STAT PROCs. The code can be customized to fit the platform that will be used in scoring the cluster solution.

STEP 1: STANDARDIZING THE VARIABLES

When creating cluster solutions, it is important to standardize the variables, since cluster assignment is highly dependent on the selected units of measurement.

In SAS, PROC STDIZE can be used to standardize the variables using a variety of standardization techniques. The syntax for PROC STDIZE is:

```
proc stdize data=&input out=&output method = std;
  var &varlist;
run;
```

The STD method of standardizing the variables uses the following formula:

$$SV1 = (V1 - \overline{V1}) / \sigma(V1)$$

Once again, if the client does not have SAS, you will need to write out the code for standardizing variables in code that is not platform dependent.

The SAS code below will write out the STDIZE procedure with the STD option into an external text file that could be customized into other platforms:

```
%let input_data = crssamp.bank8dtr; /*input data set*/
%let inputs = atmct adbdda ddatot
ddadep income savbal atres invest; /*variables in cluster solution*/
```

```
*****
**** macro to create code for standardizing ****
**** variables ****
*****
%macro create_std_code;
```

```
*****
**** use proc stdize to obtains means and ****
**** standard deviations for each input ****
**** variable. Output to dataset called ****
**** statout ****
*****
```

```

proc stdize data=&input_data method=std outstat=statout;
    var &inputs;
run;

*****
**** transpose data set to get into format ****
**** easy for creating code ****
*****
proc transpose data=statout out=trans_statout;
    var &inputs;
run;

*****
**** create code for standardizing variables****
**** the standardized variable will have the****
**** "std" prefix ****
**** file to a text file called stdcode.txt ****
**** to be brought in later ****
*****
data _null_;
    set trans_statout;
    length var $ 100;
    var = "std"||_NAME_||" = ("||_NAME_||"-"||col1||")/"||col2||";
    file "c:\stdcode.txt";
    put var;
run;
%mend;
%create_std_code;

```

The output file for this particular example (c:\stdcode.txt) looks like the following:

```

stdATMCT = (ATMCT - 4.189)/5.1741342109;
stdADBDDA = (ADBDDA - 6895.2665)/17193.426329;
stdDDATOT = (DDATOT - 6639.8605)/18524.237064;
stdDDADEP = (DDADEP - 4866.867)/12988.454391;
stdINCOME = (INCOME - 394.8325)/271.79780438;
stdSAVBAL = (SAVBAL - 5477.7065)/18313.107713;
stdATRES = (ATRES - 6.6805)/6.4801555504;
stdINVEST = (INVEST -3700.4125667)/16659.881667;

```

STEP 2: CALCULATING EUCLIDEAN DISTANCES

The centroids from the cluster solution are used to assign observations to clusters. To generate the centroids using SAS procedures, the OUTSTAT option is used in PROC FASTCLUS.

Once the centroids are generated, distances are calculated by summing the squared differences between the standardized observation for each variable and the cluster seed for each variable. The square root of the final sum is the Euclidean Distance.

There will be one Euclidean Distance calculated for each cluster in the cluster solution. So if the final cluster solution has 3 clusters, there will be 3 Euclidean Distances calculated.

The following SAS code creates the centroids needed to calculate Euclidean distances:

```

*****
****standardize the variables ****
*****
proc stdize data=&input_data method=std out=sbank8dtr outstat=statout;
    var &inputs;
run;

*****
**** generate the centroids for each of the 3 cluster ****
**** solutions ****
*****
proc fastclus data=sbank8dtr outstat=centroids noprint maxclusters=3 ;
    var &inputs;
run;

*****
**** print out the seeds ****
*****
proc print data=centroids;
    where _TYPE_ = "SEED";
run;

```

run;

Output from the data set with the centroids looks like this:

TYPE	CLUSTER	ATMCT	ADBDDA	DDATOT
SEED	1	0.019714	-0.03605	-0.09032
SEED	2	-0.03653	1.864435	6.328315
SEED	3	-0.44192	0.19432	-0.07901

DDADEP	INCOME	SAVBAL	ATRES	INVEST
-0.08763	-0.01919	-0.02884	-0.01487	-0.16559
5.973098	0.548555	0.231866	0.043793	-0.09967
-0.01976	0.25502	0.585667	0.327828	3.850775

Once the centroids have been calculated, the Euclidean distances for each observation from each cluster are calculated using the formula

$$d_i = \sqrt{\left(\sum_{j=1}^n (x_{ij} - seed_i)^2 \right)}$$

where i represents the number of clusters and j represents the number of variables used in the cluster solution. Below is the code that calculates the Euclidean distance of each observation for each cluster.

```
*****
**** Calculate distances based on the standardized ****
**** variables created above ****
*****
distance1 = ((STDATMCT-(0.0197144656183025))**2 +
(STDADBDDA-(-0.0360520673211277))**2 +
(STDDDATOT-(-0.0903248317025))**2 +
(STDDDADEP-(-0.08763316609101))**2 +
(STDINCOME-(-0.0191910964676634))**2 +
(STDSAVBAL-(-0.0288449328598474))**2 +
(STDATRES-(-0.0148720219943458))**2 +
(STDINVEST-(-0.16559413139039))**2)**.5;

distance2 = ((STDATMCT-(-0.0365278503216653))**2 +
(STDADBDDA-(1.86443461593895))**2 +
(STDDDATOT-(6.32831519713901))**2 +
(STDDDADEP-(5.97309804794292))**2 +
(STDINCOME-(0.54855509877083))**2 +
(STDSAVBAL-(0.23186596309652))**2 +
(STDATRES-(0.0437930404720257))**2 +
(STDINVEST-(-0.09966532775392))**2)**.5;

distance3 = ((STDATMCT-(-0.44192146597055))**2+
(STDADBDDA-(0.19431997549562))**2 +
(STDDDATOT-(-0.07901089758736))**2 +
(STDDDADEP-(-0.0197568467120874))**2 +
(STDINCOME-(0.2550198726622))**2 +
(STDSAVBAL-(0.58566678217572))**2 +
(STDATRES-(0.32782824922068))**2 +
(STDINVEST-(3.85077484762129))**2)**.5;
```

STEP 3: ASSIGNING CLUSTERS

Observations are assigned to clusters according to the minimum of the Euclidean Distances. If the first Euclidean distance is the minimum of all Euclidean distances for each cluster, then the observation is assigned to cluster 1, and so on.

The following SAS code assigns the observations to each cluster based on Euclidean distance:

```
if distance1=min(distance1,distance2,distance3) then cluster=1;
else if distance2=min(distance1,distance2,distance3) then cluster=2;
else if distance3=min(distance1,distance2,distance3) then cluster=3;
```

CONCLUSION

It is always a good idea to check the work that you have done to see if it matches up perfectly with the SAS procedures. One way to do this is to make sure that the cluster assignments from the cluster procedure and the cluster assignments from the calculated clusters fall on the diagonals. This can be done with the FREQ procedure in SAS:

```
proc freq data=check_distance;
  tables cluster*clusterqc/norow nocol nopercnt;
run;
```

CLUSTER clusterqc				
Frequency	1	2	3	Total
1	1902	0	0	1902
2	0	28	0	28
3	0	0	70	70
Total	1902	28	70	2000

Another quality control check is to make sure that the Euclidean distances have the same values. This can be done with the CORR procedure in SAS:

```
proc corr data=check_distance;
  var distance1 distance;
  where cluster = 1;
run;
```

The output from the CORR procedure looks like this:

The CORR Procedure

2 Variables: distance1 DISTANCE

Simple Statistics

Variable	N	Mean	Std Dev	Sum	Minimum	Maximum
distance1	1902	1.90735	1.21064	3628	0.52320	16.44631
DISTANCE	1902	1.90735	1.21064	3628	0.52320	16.44631

Simple Statistics

Variable Label

distance1
DISTANCE Distance to Cluster Seed

Pearson Correlation Coefficients, N = 1902
Prob > |r| under H0: Rho=0

	distance1	DISTANCE
distance1	1.00000	1.00000 <.0001
DISTANCE Distance to Cluster Seed	1.00000	1.00000 <.0001

This would be repeated for the other 2 clusters.

Many people are intimidated by writing out the stored SAS procedures into code used for other applications. It is not really that difficult to have the DATA STEP assign clusters for the cluster solution.

ACKNOWLEDGEMENTS

I would like to thank Junli Zhou, PhD, the research staff at Euro RSCG 4D Discovery, and David Yeo, PhD for the inspiration and guidance in the writing and review of this paper.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. For more information contact:

Bill Hartley, M.S.
Senior Research Analyst
Euro RSCG 4D Discovery
4490 Cox Road
Glen Allen, VA 23060
Phone: 804.968.7467
e-mail: bill.Hartley@eurorscg.com



TRADEMARKS

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other brand and product names are registered trademarks or trademarks of their respective companies.