Paper 100-26

Macros for Euclidean Distances

Robert O. Bernstein, California Department of Education

ABSTRACT

This paper describes how SAS® software was used to Measure the Euclidean distances between observations for data on every school in California. Distances were measured in order to test a method of identifying sets of the 100 most similar schools for each particular school. Results could be used to compare school performance measures between similar schools in California.

Macros were written to do the repetitive calculations on each school. Simple, but effective, macros were used. The power of macros has increased drastically since 1985 when I published my last SUGI paper, which also covered the use of macros.

The program used in this report uses only base SAS. While it was written using the SAS system for PCs, it should work on all operating systems. Macro code is presented for any level SAS user, but is aimed for SAS users that are beginning SAS macro programmers.

DISCLAIMER

Please note that I wrote this paper before I was employed by the California Department of Education, and the ideas, views and other material are presented here without the approval of the California Department of Education and therefore the views are my own. The methods for developing a list of similar schools contained in this paper have not been reviewed nor even considered for use by any officials in the California Department of Education or elsewhere.

INTRODUCTION

California, like many other states, is very concerned with accurately measuring the performance of their public schools. California established the Academic Performance Index (API) to measure school performance. The API is a performance measure based upon test score results of pupils in each school. One meaningful way to look at a school’s API score would be to compare it only to schools that are most like that particular school.

"Most like" in California currently means the 100 schools that have the closest predicted API score when their demographic data are run through a multiple regression equation. None of their actual data values may be close, but the combination of factors results in similar predicted API scores. Factors used include: student social economic status, average parent education levels, average class size, year around school indicator, percent of students by 7 different ethnic groups, percent of students who are English language learners, percent of teachers with a full credential, percent with an emergency credential and student mobility.

This paper uses the same data, but a different method, to try and determine which 100 schools are most similar to each particular school. SAS is used to measure the multi-dimensional distance between each school. Each variable used is treated as one dimension. These Euclidean distances are theoretical distances between each point (school). Distances are measured using the basic formula for the distance between any two points:

$$D = (\Sigma\ (x_i - y_i)^2\ )^{½}$$

The distance is the square root of the sum of the squared differences between each point in each dimension. X_i represents the values for point (school) X over all of the i dimensions (variables). Y represents the values for the second point (school) for each variable (i).

PROBLEM

Distance measurements can only be calculated when the data for each particular school is compared to the data for every other school, but for one school at a time. First you must have a file where each school has data that can then be compared to every other school. Renaming and retaining values in order to build a distance matrix can do this. However, the results would not be in a useful format since the first observation would contain no distances, the second observation only one distance, the third observation only two distances and so on. All of the distances would be calculated, but they would only be with one of the two observations needed. That is, the distance between School A and School B would be in the observation for School B, but not School A. Each observation would need to store several thousand variables, as the same variables for each school would need their own unique variable names.

Another method to calculate the distances would be to create a separate file for each school and then merge the file for all schools into each of the individual school files. This method results in a file for each school that contains all of the distances to every other school. This is the method used in this report.

HOW THE PROGRAM WORKS

First formats are created that will be used to change some character data to numeric.

```
* create format to change year around yes/no
to numeric 100/000;
proc format;
  value $ yearr
  'no '='000'
  'yes'='100'  ;
```

The Data step starts by adding in the usual data, set, length and keep statements to create the file using the least space possible. A where statement is used to keep only one type of school (H for High schools, M for Middle schools or E for Elementary schools). Similar schools are only calculated for the same type of schools (H, M or E). This paper presents data only for high schools. Since data values are necessary to calculate distances, records missing critical data are deleted. No attempt was made to use average values or other estimates in place of missing values.

Call symput is used to place values in macro variables. This allows the use of do loops later. Do loops use the file names for each school and also the number of files, in the &nobs macro variable, to process each school file. School names are also placed in macro variables so they can be used later in title statements.

School codes are stored in the macros elmXXX where XXX is the value i. Since there are about 800 high schools in California with valid data, the value of i varies from 1 to 800. Elm1 might be 1000001, the unique seven digit school code for the first school, while elm800 will be 9964489, the school code of the last high school in the file. These are macro variables that can now be used anywhere in the program. Similarly, nam1 through nam800 contain the names of these same 800 high schools. Nam1 might have the value of “Alameda California High School” the first high school in the file and nam800 could be “Last High School”.

```
data cluster;
  length pct_sd yearround 3;
set berniec.api1999B(keep=school sch_type
   sch_code api99 co_code   num_tested mobility
   pct_af_am pct_am_ind pct_asian pct_fil
   pct_hisp pct_pac pct_white sd_num
   pct_cred pct_emerg pct_el core yr_rnd);

   * keep only high schools ;
   where   sch_type='H';

   * delete schools with missing data;
   if (num_tested='    ' and yr_rnd='   ') or
      (pct_cred='   ' and pct_emerg='   ' and
   pct_white='   ' ) then do;
   put school= num_tested= yr_rnd=
   mobility= api99=  'has been deleted';
        delete;
   end;

   i+1;
   call
   symput('elm'||trim(left(i)),trim(left(sch_cod
   e)));
   call
   symput('nam'||trim(left(i)),trim(left(school)
   ));
   match='y';
   pct_sd=(100*sd_num/num_tested);
   YearRound=put(yr_rnd,yearr.);

   * put number of observations kept into &nobs;
   call symput('nobs',trim(left(i)));

   run;

   %put &nobs= ;

   run;
```

Data step cluster places the number of observations kept in the data set into the macro variable nobs.   This number is needed as a counter in the macro do loops later in the program.

CLUSTER 2 MARCO

Macro cluster2 creates one data set for each school. Each particular schools data set will have their school code as part of the data set name. Do loops are used to output the actual data from the first data set to each of the individual school data sets.   Data for only one school is contained in each data set.

Macro cluster2 uses 2 basic macro do loops. First, the data set created is based on the value of elmxxx, where elm1 through elm800 contain the school codes of every high school in the state. Thus the first data set created, when i has the value of 1, might be E1000001, which is the school code placed after the letter E. Note, &&elm resolves to &elm while &I resolves to the value of i, leaving E&elmi, which then resolves to E followed by the school code.

```
   %macro cluster2;
   data %do i=1 %to &nobs;
        E&&elm&i
             %end ;;
                %do i=1 %to &nobs;
                set cluster;
                i+1;
                match='y';
                %if &i>0 %then if
   sch_code="&&elm&i" then output E&&elm&i;
                ;
        %end;
   %mend cluster2;

   %cluster2;

   run;
```

The second do loop just brings in the same data set each time, cluster and increments the value of i. It also creates a dummy merge variable, called match. The do loop also contains the %if code that checks for the correct record in the cluster data set to output.

The %cluster2 line simply runs the macro named “cluster2.”

DISTANCE MACRO

Macro “distance” does the distance calculation. Data from data set cluster, which has all schools, is merged with data for each individual school (E&&elm&i). Data from the individual school files has to be renamed or it will overwrite the data values from the cluster file that contains variables with the same names. Only those data elements needed are kept from each school.

Merging is by a dummy variable called match since match=y for all observations. This allows each school file to be merged with the full file for all schools. The resulting data sets for each school with now contain every school and also each record for every school will contain the individual data for the school whose code number is used to name the file.

```
   %macro distance;
   %do i=1 %to &nobs;
   data D&&elm&i;
     merge cluster
   E&&elm&i(rename=(mobility=ddmobility
        pct_af_am=ddpct_af_am
        pct_am_ind=ddpct_am_ind
        pct_asian=ddpct_asian
        pct_fil=ddpct_fil
        pct_hisp=ddpct_hisp
        pct_pac=ddpct_pac
        pct_white=ddpct_white
        pct_other=ddother
        pct_sd=ddpct_sd
        pct_cred=ddpct_cred
        pct_emerg=ddpct_emerg
        pct_el   =ddpct_el
        core     =ddcore
        yearround=ddyearround
        sch_code=eeschcode
        school=ddschool        )

        keep=mobility pct_af_am pct_am_ind
         pct_asian pct_fil pct_hisp
         pct_pac pct_white pct_sd
         pct_cred pct_emerg pct_el
         core yearround api99
         sch_code school match);

   by match;

   length ddschcode 6;
   ddschcode=put(eeschcode,7.);
```

Some lines left out check for ‘N/A’, which was used for missing character data even though all of the data should be numeric. These checks avoid errors when trying to convert some character data to numeric.   Missing data will result in inaccurate data and these schools should be excluded or have the missing data replaced with an average or some other estimate.

An example of one of the data checks for invalid numeric data is:

```
   if pct_cred='n/a' then pctcred=.;
      else pctcred=put(pct_cred,3.);
```

DISTANCE CALCULATIONS

Since some of the data used are correlated, changes need to be made before distance can be calculated. Data are available by 8 ethnic groups. Differences in ethnicity can either be measured by using each of the 8 variables separately, using 7 of the variables (the 8th being fully accounted for since the total of the 8 is always 100), using Mahalanobis distances to adjust for the correlation between ethnic variables or by measuring the difference in ethnicity as one variable. This program uses the latter method in order to avoid problems with the other 3 methods. Ethnicity difference is simply 100 percent minus the amount of matching ethnicities. The matching ethnicity is equal to the minimum of the percent of ethnicity for each pair of schools for each ethnicity. A similar formula was used to adjust the two teaching credential variables, percent full and percent emergency teaching credentials.

Distances are calculated using unadjusted data. Many times data are standardized, to a mean of 0 and standard deviation of 1, prior to calculating the distance.

The actual formula to calculate the distance is:

```
   Diffethnic=100-min(af_am,ddaf_am)
   - min(pct_am_ind,ddam_ind)
   - min(pct_asian,ddasian)
   - min(pct_fil,ddfil)
   - min(pct_hisp,ddhisp)
   - min(pct_pac,ddpac)
   - min(pct_white,ddwhite)
   - min(pct_other,ddother);

   diffteach=max(100-min(pctcred,ddpctcred)
    -min(pctemerg,ddpctemerg),0);

   Distance=(sum(((mobile    - ddmobile)**2),
        ((diffethnic)**2),
        ((pct_sd    - ddpct_sd)**2),
        ((diffteach)**2),
        ((pct_el    - ddpct_el)**2),
        ((cores     - ddcores)**2),
        ((yearround - ddyearround)**2)
                       ))**.5;

   drop ddpct_af_am ddpct_am_ind ddpct_asian
   ddpct_fil ddpct_hisp ddpct_pac  ddpct_white
   ddmobility  ddcores ;

         format distance 7.1;

   label        ddschool='School in file'
                School  ='Comparison School'
                eeschcode='school Code this file'
                ddyearround='YearRoundSchool'
                ddaf_am='% AfAm File School'
                distance='Dist-ance'
        ;

   %end;
   %mend distance;
```

After the macro is completed, it is run:

```
   %distance;
   run;
```

DISTANCE ORDER MACRO

Macro sortus sorts each school file by distance in order to identify only the 100 closest schools (plus the one comparison school). This macro is just a do loop with a proc sort in the middle.

```
   %macro sortus;
   %do i=1 %to &nobs;
    proc sort  data=D&&elm&i ;
                      by distance;
   %end;
   %mend sortus;
```

BERNIE MACRO CALCULATIONS

Macro bernie outputs each school to its own permanent file. The set option, obs=101, keeps only the particular school itself and the 100 closest observations in each data set. The macro uses the counter variable place to track the number of the 100 comparison schools that have a higher API than the particular school. The comprank variable uses rankc format to convert the value of the place variable of each record to a decile rank for each school. Deciles are from 1 to 10 and are divided by the tenth, twentieth, etc percentiles. The last observation then holds the correct value of comprank.

```
   libname library 'c:\sas\formats\';
   %macro bernie;
    %do i=1 %to &nobs;
   data E&&elm&i;
     set D&&elm&i(obs=101 keep=ddschool
   ddschcode school distance mobility pct_sd
   pct_af_am pct_am_ind pct_asian pct_fil
   pct_hisp pct_pac pct_white pct_cred pct_emerg
   pct_el      core yearround i api99 co_code
   sch_code);

   retain ddapi;

   if _n_=1 then ddapi=put(api99,3.);
     else if api99 LE ddapi then place+1;

   comprank=put(place,rankc.);

   %end;
   %mend bernie;

   %bernie;
   run;
```

MACRO SORTUS2, BERNIE2 AND SORTUS3

Macro sortus2 then sorts each record in the opposite order, by descending distance. This puts the correct value of comprank in the first record. Macro bernie2 then retains the first value of rank (which is initially set to the first value of comprank) and then retains it to place it in each observation. Macro sortus3 sorts the files back in distance order.

```
   %macro sortus2;
   %do i=1 %to &nobs;
    proc sort  data=E&&elm&i ;
                      by descending distance;
   %end;
   %mend sortus2;

   %sortus2;
   run;
```

```
%macro bernie2;
 %do i=1 %to &nobs;
data D&&elm&i;
  set E&&elm&i;
retain rank;
if _n_=1 then rank=comprank;
label rankcomp='Comp.#Schools#Rank';
drop comprank place rank;
if _n_=101 then rankcomp=rank;

%end;
%mend bernie2;

%bernie2;   run;

%macro sortus3;
%do i=1 %to &nobs;
 proc sort  data=D&&elm&i
out=bernied.H&&elm&i;
                    by distance;
%end;
%mend sortus3;

%sortus3;   run;
```

MACRO BERNIE3

Macro Bernie3 is used to output one record for each school into the file bernied.highcomp.   The macro starts with one data statement and then uses a %do statement to loop through each of the bernied.H&&elm&i High school files.

```
%macro bernie3;
data bernied.highcomp;
  %do i=1 %to &nobs;
  set bernied.H&&elm&i;
if sch_code=ddschcode then output
bernied.highcomp;
%end;
drop i distance ddapi ddschcode ddschool;
%mend bernie3;

%bernie3;
run;
```

MACRO PRINT1

Macro print1 prints out a list of comparable schools for every school. Only a few variables are printed, as this is a rather long list. This printout is set up for landscape printing with a smaller font to fit more output on each page. This macro is a good one to skip unless you want your output window filled to capacity.

The title2 and title3 statements place both the school code and school name in the title. When using macro variables in the title you must use double quotes.

```
* print out 101 most comparable schools for
each school;
* print out a few data elements only;
* need to manually set font size in Menu
File-Print-Setup-Font;
* options FONT=(Sasfont 7) ls=120 ps=50
ORIENTATION=LANDSCAPE;
run;

%macro print1;
%do i=1 %to &nobs;
%let schools&i=put(&&elm&i,schname.);
proc print data=bernied.H&&elm&i label  ;
      var school distance mobility pct_sd
pct_white yearround rankcomp;
title2 "One Hundred (101) Most Comparable
Schools to school code &&elm&i";
title3 "&&nam&i";
%end;
%mend print1;

%print1;
run;
```

MACRO PRINT2

Macro print2 prints only a few selected schools. Unlike macro print1, print2 prints many more variables. Prior to running macro print2 you need to place values in the variables &start and &nnobs. &start is the observation number of the first record to print and &nnobs is the observation number of the last record to print.

In the example of High Schools, observations 230 through 233 are printed. This includes 4 schools in Los Angeles County. The file was sorted in an order which includes county and district code in addition to school code. The sort code was not included since it involved no macros and was just a simple sort statement.

```
* partial print of some High Schools;

%macro print2;
%do i=&start %to  &nnobs ;
%let schools&i=put(&&elm&i,schname.);
proc print data=bernied.H&&elm&i label
split='#';
      var school co_code distance mobility
pct_sd pct_el  pct_white pct_hisp pct_af_am
pct_am_ind pct_asian pct_fil pct_pac
Core yearround pct_cred pct_emerg rankcomp;
;title2 "One Hundred (101) Most Comparable
Schools to school code &&elm&i";
title3 "&&nam&i";
%end;

%mend print2;

%let start = 230;
%let nnobs = 233;
%print2;
run;

* The previous %let statements set up the
macro print2 to print a sample of
observations, from observations 230 through
233.  This includes schools only from Los
Angeles County and includes a few examples of
schools that are yearround=yes.;
```

MACRO DELETE

Macro Delete removes some of the work files in order to free up space for other processing. This macro is a proc datasets, with a delete statement followed by a do loop. The do loop includes the names of each of the files that is to be deleted. In this case, the files are named E and D followed by the school codes for each school. This one macro deletes 2 files for each school, about 1,600 total using the high school example.

```
%macro deletes;
proc datasets lib=work;
     delete %do i=1 %to &nobs;
     E&&elm&i D&&elm&i      %end ;;
%mend deletes;
```

FUTURE ENHANCEMENTS

Countless other variables may also be added to this type of analysis to improve the results. These results are mostly used for information for schools and parents. Since schools and parents are most familiar with local schools, the method could be altered to increase the likelihood of having more nearby schools in the list of similar schools. This method would include using a reward/penalty term. Variables may be added to identify if each school is in the same county and the same district. Values for each pair of schools might be 0 difference if the schools are in the same district, 10 if the schools are in the same county, but different districts and 20 if the schools are in different counties. This would increase the chances of having more local schools in each schools similar school list.

Would it be statistically correct? Not really. However, the selection of similar school measures is a political decision and not really a statistical one. Support for using such measures comes from the fact that schools in the same district are under the same governing board, have the same funding and draw students from the same populations. County designation is a less logical variable, but schools in the same county usually have the same countywide tax rates and are under the same county board of Education. Better variables might be individual measures for each district (such as tax rates, bonds, etc.) but most policies of local boards cannot be reduced to quantifiable variables.

Not all elementary schools have the same grade levels. The typical elementary school covers K-6 (Kindergarten through grade 6). However, many schools only cover K-5 and a fair amount of others cover only K-2, 3-6 or some other combination other than K-6. These differences can have a large effect on the API scores. Percentage of students by grade level can be added in as a variable in the same way ethnicity is used. This would increase the similarity of similar schools for schools that do not have the traditional K-6 or 7-8 grade levels. This issue will increase in importance as grade level tests differ even more in the future. Starting next year 4th graders and 7th graders will take written tests in addition to the standard multiple-choice tests. Adding in grade level as a variable would improve the similar schools method as the tests change even more in the future.

The year-round school variable in this paper uses a zero or 100 value for each school. This has the effect to make sure that almost every year-round school has all of the other year-round schools in their comparison group. This can be avoided by coding it as 100 or 75, since non year-round schools are still in school 75% (9 of 12 months) the time.

Please note that this section is by no means a complete list of possible future enhancements. Countless other variables may also be added to this type of analysis. The current system, which is not the system described in this paper, is constantly being evaluated for possible improvements.

CONCLUSION

Macros can be very useful tools for certain problems. Repeating tasks using Macros was the best solution for this problem. Macros can save a lot of time and cut down on typing mistakes. In my example, each time a macro is used to identify files it replaces writing each of the 800 school codes, a very tireless and error riddled process.

REFERENCES

Bernstein, Robert O. (1991), Hospital Peer Grouping, Sacramento, CA: California Department of Health Services.

Bernstein, Robert and Tim Tyler, Ph.D. (1982), Hospital Peer Grouping for Efficiency Comparison, Sacramento, CA: California Health Facilities Commission.

Bernstein, Robert. (1985), "Unique Uses of SAS Software for Determining Medi-Cal (Medicaid) Hospital Inpatient Reimbursement," Proceedings of the Tenth Annual SAS User Group International Conference, Pages 524-529.

Carpenter, Art "Carpenter's Complete Guide to the SAS Macro Language," Cary, NC: SAS Institute., 1998. 242 pp.

Everitt, Brian (1980), Cluster Analysis, New York, NY: Halsted Press.

SAS Institute Inc. (1990), SAS/STAT User's Guide Volume 1, ACECLUS-FREQ, Cary, NC: SAS Institute, Inc.

ACKNOWLEDGMENTS

SAS is a registered trademark of SAS Institute Inc.
My thanks to my friends for reviewing my drafts.

CONTACT INFORMATION

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

Robert O. Bernstein
California Department of Education
P.O. Box 13
Carmichael, CA 95609-0013

Work Phone:	(916) 654-3676
Fax:	(916) 657-5201
Email:	rbernste@cde.ca.gov
Web:	www.cde.ca.gov