Paper 211-26

Mixing It Up with the SAS® System

William C. Murphy
Howard M. Proskin & Associates, Inc., Rochester, NY

ABSTRACT

All of the subjects have been examined. You have acquired a mountain of data. The data has been read into the SAS system and the basic demographic tables have been outputted. Now how do you proceed? You are trying to model an effect, but you have dozens of variables. The selection of a discrimination model is both an art and a science. Often, the choice of the most suitable model involves a consideration of the sensitivity and specificity of potential model candidates. PROC DISCRIM provides an easy way to find these quantities for variable combinations, but you still must come up with those combinations. What you need to do is run PROC DISCRIM for hundreds or even thousands of combinations of variables. A SAS macro program can readily generate these combinations and execute the procedure by employing the self-calling feature of version 8 macros. To avoid the massive number of pages generated, the SAS Output Delivery System (ODS) can be used to extract the pertinent information. The details of this programming technique will be described and the limitations of older versions of the SAS system for performing this algorithm will be pointed out.

INTRODUCTION

The physical examination of the subjects has been finished. All of the laboratory reports are in. The data has been entered into the SAS system and checked for errors and consistency. Tables describing the basic demographic makeup of the study population have been made. Simple statistics (means, medians, standard deviations, etc.) of the test variables have been created. Now is the time for a more in depth analysis of the data, but where do you begin? Your objective is to construct a model or combination of variables that relate to a measured effect. But what model or combination of variables do you use when your study data set may have dozens and dozens of variables?

Our organization, a statistical consulting firm, has been confronted with this problem, especially for some dental studies that we have been asked to analyze. Our solution was to develop a program in the SAS system, which produces a table of specificity and sensitivity for various models. This program employs the Output Delivery System (ODS) and the recursive feature of macros available in version 8 of the SAS system.

PRELIMINARIES

The objective of many dental studies is to determine the relationship of various variables descriptive of oral health with the presence or absence of dental caries (decay). To determine the exact mix of variables that appear in your model can be a daunting task. In fact with many initial variables, before you apply a programming solution, clinical insight must be employed to limit the number of variable under consideration. Even with this automated program, too many variables in the model could produce long execution times. Combinations of up to 10 variables or so produced overnight runs on NT computers with state-of-the-art processors.

To use our automated solution to help in model selection, you must start the programming by making a list of the selected variables and reading them into SAS macro variables. As has been done previously (Murphy *et al*, 1993), this can be done with a simple DATA step:

```
data  null ;
   infile cards missover eof=AllOver;
   length tmp $8;
   input tmp;
   call symput(compress('Vrbl'||left( n )),
                                  trim(tmp)));
   return;
   AllOver: call symput('nVrbl',right(_n_-1));
      return;
   cards;
FL
CA
PHOS
LACTO
MUTANS
PLAQUE
;
   run;
```

where we have used a _null_ DATA step to read a list of variables after a CARDS statement. The variables are then stored in macro variables (&Vrbl1, &Vrbl2,…) using CALL SYMPUT. The total number of variables is stored in the macro variable &nVrbl. We have limited our sample to 6 commonly measured oral parameters: the fluoride (FL), calcium (CA), and phosphate (PHOS) concentration in the saliva, the amount of lactobacilli (LACTO) and mutans (MUTANS) in the mouth, and the quantity of plaque (PLAQUE) on the teeth.

COMBINATIONS

We now want to make a list of combinations of these variables and find their sensitivity and specificity. PROC DISCRIM will provide the information that we are searching for. Therefore, we imbed this procedure in a looping macro program that generates a list of the variables (single combinations) for the VAR statement:

```
%macro mix;
   %do i1=1 %to &nVrbl;
         %let model=&&Vrbl&i1;
         proc discrim data=master;
            class caries;
            var &model;
            run;
            %end /* i1 loop */;
   %mend;
```

where PROC DISCRIM will analyze a different variable for each iteration of the %DO loop. The input data set, master, must contain all of the variables needed the PROC DISCRIM (*i.e.* those listed in the VAR and CLASS statements). If we now want to examine both single and pair combination of the variables, we imbed a new %DO loop into the previous %DO loop in our macro:

```
%macro mix;
   %do i1=1 %to &nVrbl;
      %let model=&&Vrbl&i;

      proc discrim;
         class caries;
         var &model;
         run;

      %do i2=%eval(&i1+1) %to &nVrbl;
         %let model=&&Vrbl&i1 &Vrbl&i2;

         proc discrim data=master;
            class caries;
            var &model;
            run;
            %end  /* i2 loop */;

         %end  /* i1 loop */;
   %mend;
```

where the inner loop index starts at one plus the outer index to prevent duplicate combinations of variables. Indeed we can examine higher levels of combinations by imbedding more and more %DO loops into our macro. However, there is no simple way to make these %DO loops self-generating in out macro in versions of the SAS system before version 8. Before the macro was executed, you would have to choose the level of combinations that was desired and imbed the appropriate number of macros. With version 8 of the SAS system, macro programs can call themselves. Since the inner %DO loops are of the same form as the outer, a recursive macro would enable us to specify a parameter to determine the level of combinations that we wanted. The new macro for our analysis would be

```
%macro mix(Combo=,,j=1,start=1,xmod=);
   %do i&j=&start %to &nVrbl;
      %let model=&xmod &&&&Vrbl&&i&j;

      proc discrim data=master;
         class caries;
         var &model;
         run;

      %if &j<&Combo %then %mix(Combo=&Combo,
                               j=%eval(&j+1),
                         start=%eval(&&i&j+1),
                                 xmod=&model);
      %end;
    %mend;
```

where we have introduced four macro parameters into our program: &combo, which contains the desired number of combinations (i.e., singles, pairs, triplets, etc.); &j, which is the level of nested macro calls used to index the %DO-loop parameter; &start, which is the starting value of the %DO loop; and &xmod which contains the outer loops variable combinations. This is very similar to our previous macro, except the imbedded %DO loop is replaced by a conditional call of the macro itself. This effectively produces as many nested %DO loops as indicated by the macro parameter &combo. The other major difference with the previous macro is the %model definition. It now contains the macro parameter %xmod, which saves the variable combinations from outer %DO loops. Also, the variable macro now has four ampersands, due to the indexing of the %DO-loop parameter: for the outer %DO loop, one pass of the macro processor resolves &&&&Vrbl&&i&j into &&Vrbl&i1; for the next %do-loop, &&&&Vrbl&&i&j becomes &&Vrbl&i2; and so on.

OUTPUT

The %mix macro as it now stands will solve our problem of generating sensitivity and specificity for an arbitrary combination of study variables. However, PROC DISCRIM will generate considerably more information than we need. Indeed a number of pages of output would be produced for each combination. Unless we are interested in running a major deforestation project, we must limit the output of PROC DISCRIM to the desired information. We could write the procedure section of our macro as

```
proc discrim data=master out=TmpData1 noprint;
   class caries;
    var &model;
    run;
```

This method does indeed cut out the printed output but the data set TmpData1 does not contain the exact information that we desire. A PROC FREQ on the data set TmpData1 could be used to produce the tables of sensitivity and specificity. The output of the PROC FREQ would be further processed in a DATA step in order to get the variable counts used in the analysis. However, even though the output data set does not contain the exact information that we need, the printed output of PROC DISCRIM does. With version 8 of the SAS system, ODS can be use to avoid the overhead of this additional processing by capturing the desired part of the printed output. Using ODS, our procedure now becomes

```
ods listing close;
ods output ClassifiedResub=
         TmpData1(drop=_0 _1 Ptotal);

proc discrim data=master;
   class caries;
   var &model;
   run;

ods listing;
```

The first ODS statement stops the flow of data to the output window. This is similar to using the NOPRINT option in the procedure. The second ODS statement sends the ClassifiedResub component of the PROC DISCRIM output into the SAS data set TmpData1. The name of this component was determined by running a sample PROC DISCRIM program without the ODS statements and examining the Results window for the appropriate name. The final ODS statement returns normal flow of data to the output window. The output data set, TmpData1, contains the variables fromCaries, which identifies the levels of the CLASS variable caries; the count, total, predicted by the model for each level; and the desired percentages associated with the count, p_ and p_2.

FINAL DATA SET

The data set TmpData1 takes a modest amount of processing to produce our desired results. We merge the specificity and sensitivity information so that it will be in the same line of the data set; we rename the variables accordingly; and we introduce

the variable model, to label the data line with the variable combinations:

```
data TmpData2;
   merge TmpData1(where=(fromcaries='0')
            rename=(total=n0 P =specfcty))
         TmpData1(where=(fromcaries='1')
            rename=(total=n1 P_2=senstvty));
   length model $200;
   model="&model";
   keep model specfcty senstvty n0 n1 ;
   run;
```

Finally, we accumulate the data from the iterations of the %do-loops into an output data set:

```
proc append base=DiscrimOut data=TmpData2;
   run;
```

This output data set will contain a listing of all models examined along with their specificity and sensitivity. The resulting information could be further processed by sorting or sub-setting before printing or on-screen examination.

CONCLUSIONS

Using the recursive macro capabilities and the ODS of version 8 of the SAS system, we were able to develop a macro program that produces various combinations of variables and then generates the related specificity and sensitivity. For determining the best study variable combinations for further analysis, the researcher can use this list.

APPENDIX

The following is a listing of the complete macro that is presently in use. This macro can be compiled and stored in a central library for convenience (Murphy, 1998).

```
%*
╔═══════════════════════════════════════════════════════════╗
 Project:    DISCRIMINATE MODELING

 Programmer:  WILLIAM C. MURPHY
 Date:  JANUARY 21, 2001

 Objectives:  CREATE A SAS DATA SET WITH THE SPECFICITY
AND SENSITIVITY OF VARIOUS COMBINATIONS OF VARIABLES

╚═══════════════════════════════════════════════════════════╝;
%macro mix(Combo=,j=1,start=1,xmod=);
   option nonotes nosource;
   %do i&j=&start %to &nVrbl;
      %let model=&xmod &&&&Vrbl&&i&j;

      %**Write Message to Log***;
      options notes source;
      %put;
      %put NOTE: Processing &model;
      %put;
      options nonotes nosource;

      %**Discriminat analysis***;
      ods listing close;
      ods output
         ClassifiedResub=
            TmpData1(drop=_0 _1 Ptotal);

      proc discrim data=master;
         class caries;
         var &model;
         run;

      ods listing;

      %**Take only desired information ***;
      data TmpData2;
         merge TmpData1(where=(fromcaries='0')
                     rename=(total=n0
                              P_=specfcty))
               TmpData1(where=(fromcaries='1')
                     rename=(total=n1
                              P_2=senstvty));
         length model $200;
         model="&model";
         keep model specfcty senstvty n0 n1 ;
         run;

      %**Append information to ouput dataset***;
      proc append base=DiscrimOut data=TmpData2;
         run;

      %if &j<&Combo %then
            %mix(Combo=&Combo,j=%eval(&j+1),
              start=%eval(&&i&j+1),xmod=&model)
           ;
      %end;

   %*** Clean Up ***;
   proc datasets nolist library=work;
      delete TmpData1 TmpData2;
      quit;

   option notes source;
   %mend;
```

REFERENCES

Murphy, W. C., Proskin, H. M., and Freeman, G.B. (1993), "Using the SAS® System for Error Control in Clinical Trials Databases," *Proceedings of the Eighteenth Annual SAS Users Group International Conference*, 18, 956-958.

Murphy, W. C. (1998), "Creating and Maintaining a Central SAS® Library for Health Care Management," *Proceedings of the Twenty-Third Annual SAS Users Group International Conference*, 23, 1128-1130.

ACKNOWLEDGEMENTS

CONTACT INFORMATION

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

William C. Murphy
Howard M. Proskin & Associates, Inc.
2468 E. Henrietta Rd.
Rochester, NY 14623
Phone 716-359-2420
FAX 716-359-0465
E-mail wmurphy@hmproskin.com