Using Macro and ODS to Overcome Limitations of SAS® Procedures
Jing Su and Wei (Lisa) Lin, Merck & Co, Inc., North Wales, PA

ABSTRACT

SAS® users may face situations where some "powerful" options are only available in certain SAS® procedures but not available in others. For example, the model selection options are available in PROC REG, LOGISTIC, PHREG, etc., but not in PROC GENMOD, CATMOD, MIXED, etc. "Manually" coding model selection steps for each specific task is often very labor intensive, and the code is lengthy and hard to maintain and reuse.

As we all know, SAS® programming productivity can be greatly improved when you know how and when to use SAS® macros. This paper demonstrates how to use macro (nested macro call), %do %until loops and ODS (Output Delivery System) to create an automated backward model selection process for PROC GENMOD which does not come with model selection options. Our automated process uses ODS to output relevant summary statistics from each step which is then used in %do %until loop to perform backward selection. This process can be easily modified to apply to other model selection methods as well as to other SAS/STAT modeling procedures.

This paper is intended for people who are interested in macro, ODS, and %do loops to create highly adaptable code to overcome the option limitations of SAS® procedures.

BUSINESS PROBLEM

Currently, SAS® does not provide the capability to fit logistic regression models for repeated measure data using automated model selection methods such as forward selection and backwards elimination. PROC LOGISTIC can provide automated model selection, but it does not support repeated measure data. While PROC GENMOD can run logistic regression with repeated measure data, it does not have automated model selection options.

MANUALLY PERFORM BACKWARD MODEL SELECTION IN PROC GENMOD

For illustration purposes, let us assume we have a clinical dataset in which some subjects have multiple records due to multiple treatments/time points. In the data, \( Y_1, Y_2, Y_3, \) and \( Y_4 \) are the binary response variables; \( C_1, C_2, \) and \( C_3 \) are the potential categorical explanatory variables; and \( M_1, M_2, M_3, M_4, M_5, M_6 \) and \( M_7 \) are the potential numeric explanatory variables. \( S \) is the subject ID. In the analysis, we will use logistic regression with the backward model selection method to find the main effect model for each response variable. As stated above, we will use PROC GENMOD for the analysis. First we analyze \( Y_1 \) and find its main effect model. The criterion for removing a variable is defined as 0.1, and it can be any number between 0 and 1 (this is the user’s judgment). During the backward elimination process, the maximum p-value from each fitted model will be compared to this criterion. If it is greater than this criterion, it will be removed. At the end of the selection process, all the variables' p-values in the final model should be less than or equal to the criterion, and p-values from Type III analysis in each selection step are summarized into a dataset which can be used in future reporting.

The steps to manually perform backward model selection in PROC GENMOD are as follows:

Step 1: Run the initial logistic regression model with all 10 explanatory variables:

```sas
proc genmod data=indat descending;
  class S C1 C2 C3;
  model Y1 = C1 C2 C3 M1 M2 M3 M4 M5 M6 M7 /dist=bin link=logit type3 lrci;
  repeated subject=S /type=cs corrw covb;
  title "Y1 = C1 C2 C3 M1 M2 M3 M4 M5 M6 M7";
run;
```
Step 2: Create a temporary dataset which contains the explanatory variables and their p-values from the PROC GENMOD outputs. Identify the maximum p-value and the variable with the highest p-value, and compare the maximum p-value with the retention criterion 0.1. Let us assume the categorical variable $C_1$ has the highest p-value which is greater than 0.1.

Step 3: Run the logistic regression model again without $C_1$.

```
proc genmod data=indat descending;
class S C1 C2 C3;
model Y1 = C2 C3 M1 M2 M3 M4 M5 M6 M7 /dist=bin link=logit type3 lrci;
repeated subject=S /type=cs corrw covb;
title "Y1 = C2 C3 M1 M2 M3 M4 M5 M6 M7";
run;
```

Step 4: Repeat Step 2 and Step 3 until all remaining variables have p-values less than 0.1.

Step 5: Summarize the p-values of explanatory variables from each elimination step.

The next steps are to analyze $Y_2, Y_3$, and $Y_4$. In the situation of performing exploratory analysis, we may need to change the initial list of explanatory variables, or change the retention criterion and perform the backward selection again. It is a very time consuming, labor intensive, and error prone process. As a result, we have to find an easier way to tackle this problem.

**AUTOMATED SOLUTION WITH MACROS, ODS, AND %DO LOOPS**

In the following section, we demonstrate how to convert the above steps to macros with the help of ODS and %do loops, and make it more dynamic, flexible, and easy to maintain. We created two macros: %MdStmt and %MdSelect. The submacro %MdStmt contains the PROC GENMOD statements, and the main macro %MdSelect will perform the backward model selection process and create a summary dataset of this process. Each macro is explained separately.

**%MdStmt: the PROC GENMOD statements**

As presented in the manual steps above, we need to run PROC GENMOD statements multiple times depending on data and the retention criterion. At each run, we changed the following strings: the response variables and explanatory variables in the MODEL statement, and the classification variables in the CLASS statement. This is a perfect situation to write a macro. The changing strings will be replaced by macro variables: response variable replaced by &RESVAR; explanatory variables replaced by &EXPVAR; and classification variables except S in CLASS statement replaced by &CLSVAR. The code is then wrapped with %macro MdStmt, and %mend MdStmt. "MdStmt" is the macro name, and &RESVAR, &EXPVAR, and &CLSVAR are the macro parameters. See below for the code of the macro %MdStmt. Each time we run PROC GENMOD to fit the model, it is not necessary to re-type everything. Instead, call %MdStmt, and specify the values of the three macro parameters. Sample calls are provided in the macro %MdSelect.

```
%macro MdStmt(
    resvar = /*response variable        */
    ,expvar = /*list of explanatory variables, separated by ' '       */
    ,clsvar = /*classification variables in the CLASS statement separated by ' ' */
);
ods output Type3=pval(rename=source=parm);
proc genmod data=indat descending;
class S &clsvar;
model &resvar = &expvar /dist=bin link=logit type3 lrci;
repeated subject=S /type=cs corrw covb;
title "&resvar = &expvar";
run;
ods output close;
%mend MdStmt;
```

%MdStmt is a stand-alone macro. The main macro, %MdSelect, consists of multiple calls to the macro %MdStmt. This design is more efficient than nesting the %MdStmt definition inside of %MdSelect. If we nest %MdStmt inside of %MdSelect, the
macro processor will compile the same macro %MdStmt every time that %MdSelect is called and executed which is unnecessary and inefficient. It is also easier for users to understand the macros and modify the model statements because %MdStmt is defined separately.

The ODS OUTPUT statement in %MdStmt creates a temporary dataset "pval". Dataset "pval" contains the same information as in "Score Statistics for Type 3 GEE Analysis" from PROC GENMOD output, the explanatory variables, their p-values, etc. This dataset is important for automating the model selection process and for the final summary dataset. The OUTPUT statement in PROC GENMOD cannot be used for this purpose because the generated dataset contains all the variables and observations in the input dataset and required statistics. Prior to ODS OUTPUT, lines of code were written to retrieve such data from STAT procedure results by using PROC PRINTTO and DATA step.

In SAS® v9, the SAS/STAT procedures use the Output Delivery System (ODS) to manage their output. In this paper, we only utilize ODS to generate temporary SAS® datasets from the PROC GENMOD output tables which are used in subsequent analysis and reporting. Other common examples of using ODS are: selecting/excluding individual output objects; displaying the selected output in RTF which can be inserted to a MS Word document; and sending the output to a MS Excel document.

%MdSelect: the Backward Model Selection Process

The macro %MdSelect is the main macro to automate the backward model selection process and to generate a final summary dataset. It follows the similar steps as in the manual section. The macro flow is presented in the Figure 1.

---

**Figure 1. %MdSelect Macro Flow Chart**
There are four macro parameters in the macro %MdSelect: &VAR, &INTVAR, &CATVAR, and &SLSTAY:

- &VAR is the response variable which will be passed into &RESVAR when calling the macro %MdStmt;
- &INTVAR includes all the potential explanatory variables which will be passed into &EXPVAR in %MdStmt only for the first call;
- &CATVAR contains all the categorical explanatory variables which will be passed into %CLSVAR in %MdStmt;
- And &SLSTAY is the criteria for removing variable.

The code of %MdSelect is presented below:

```plaintext
%macro MdSelect(
    var= /*response variable       */,
    intvar= /*initial explanatory variables for full model */,
    catvar= /*categorical explanatory variables   */,
    slstay= /*criterion for removing variable          */
);
    %let var=%upcase(&var);
    %let intvar=%upcase(&intvar);
    %let catvar=%upcase(&catvar);

%*-------------------------------------------------------------------------*;
%* Create empty dataset "step" with only one column "parm". It will be         *;
%* merged with "pval" from PROC GENMOD by "parm"        *;
%*-------------------------------------------------------------------------*;
proc sql;
    create table step_&var (parm char(9));
quit;

%*------------------------------------------------------------------------------*;
%* %do %until performs multivariate backward model selection:            *;
%* In each iteration:                 *;
%*   1. Run the logistic regression model              *;
%*   2. Update the dataset "step_&var"               *;
%*   3. Create &pmax as the maximum p-value, and &varlist as the list of        *;
%*  variables without the one with the max p-value           *;
%*   4. Check whether the max p-value <= &SLSTAY            *;
%*   5. If NO, then eliminate the variable with max p-value, repeat step 1 to 4.*;
%*      If YES, the loop stops                                                  *;
%*------------------------------------------------------------------------------*;
%let i=1;
%do %until (&pmax<=&slstay);
    %if &i = 1 %then
        %MdStmt(resvar=&var ,expvar=&intvar, clsvar=&catvar); %*initial model;
    %else %do;
        %MdStmt(resvar=&var ,expvar=&varlist, clsvar=&catvar); %*reduced model;
    %end;
    proc sort data=step_&var; by parm;
    proc sort data=pval; by parm;
    data step_&var;
        merge step_&var pval;
        by parm;
        p&i=put(ProbChiSq, pvalue6.3);
        drop ProbChiSq ChiSq DF;
    run;
    proc sql noprint;
        select max(ProbChiSq) into :pmax
        from pval;
```

Coders' Corner
NESUG 2007
select distinct parm into :varlist separated by ' ' 
from pval 
having ProbChiSq^=max(ProbChiSq);
quit;

%let i=%eval(&i+1);
%end;

proc print data=step_&var;
title "&var: model selection process";
run;

%mend MdSelect;

With the help of %do loops, the macro follows similar steps as in the manual section but without users' evaluation for eliminating any factor. Both %do %until and %do %while execute a section of a macro repetitively until a condition is true. The difference is the %do %until statement checks the value of the condition at the bottom of each iteration, while the %do %while statement tests the condition at the top of the loop.

For the backward elimination process, %do %until is more appropriate than %do %while. The %do %until loop starts with fitting the initial full model. From the dataset "pval" generated by ODS, it calculates the maximum p-value and creates a new macro variable &VARLIST which contains the explanatory variables except the one with maximum p-value. The %do %until checks whether the maximum p-value is or is not less than or equal to the retention criteria. If it is false, the %do loop will iterate again until the condition is true. Note that in the first iteration, &EXPVAR in %MdStmt equals &INTVAR - the initial list of explanatory variables. In the second or later iterations &EXPVAR are the same as &VARLIST - the reduced list of variables.

If %do %while is adopted for this process, the code will be somewhat complicated (see the Appendix for the code) and will require running the initial full model to get the maximum p-value outside of the %do loop. The %do %while loop will start with testing whether or not the maximum p-value is greater than &SLSTAY. It will iterate while this condition is true.

Using the same example in the manual section, we will call %MdSelect four times with different response variables.

```
%MdSelect (var=Y1, intvar=C1 C2 C3 M1 M2 M3 M4 M5 M6 M7, catvar=C1 C2 C3, slstay=0.1);
%MdSelect (var=Y2, intvar=C1 C2 C3 M1 M2 M3 M4 M5 M6 M7, catvar=C1 C2 C3, slstay=0.1);
%MdSelect (var=Y3, intvar=C1 C2 C3 M1 M2 M3 M4 M5 M6 M7, catvar=C1 C2 C3, slstay=0.1);
%MdSelect (var=Y4, intvar=C1 C2 C3 M1 M2 M3 M4 M5 M6 M7, catvar=C1 C2 C3, slstay=0.1);
```

We also can easily utilize a different initial full model.

```
%MdSelect (var=Y1, intvar=C1 C2 C3 M1 M2 M3, catvar=C1 C2 C3, slstay=0.1);
```

With the flexibility of the macro, we also can try different criterion.

```
%MdSelect (var=Y1, intvar=C1 C2 C3 M1 M2 M3 M4 M5 M6 M7, catvar=C1 C2 C3, slstay=0.5);
```

**CONCLUSION**

This paper presents two macros that automate PROC GENMOD’s backward model selection process. The sub-macro, %MdStmt, uses ODS output to pass model statistics to the main macro where such statistics are evaluated to determine the keeping and removing of model variables. The nature of the problem, the need to repeatedly build and select models for multiple response variables, provides a perfect example to illustrate the benefits of using macros. This paper also demonstrated that ODS output is not only for producing html and word document, but for bridging communications between macros by producing datasets directly from the output tables.
REFERENCE

SAS OnlineDoc® 9.1.3, Copyright © 2002-2005, SAS Institute Inc., Cary, NC, USA; All rights reserved. Produced in the United States of America.

ACKNOWLEDGMENTS

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.

The authors would like to offer their sincere appreciation to Amy Gillespie and Allan Glaser for their review and valuable comments. Special thanks also go to Dr. Eunhee Hwang for her statistical advice.

CONTACT INFORMATION

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

Jing Su
Merck & Co., Inc.
UG1D-88
Po Box 1000
North Wales, PA 19454-1099
Work Phone: 267-305-6949
Email: jing_su@merck.com

Wei (Lisa) Lin
Merck & Co., Inc.
UG1D-88
Po Box 1000
North Wales, PA 19454-1099
Work Phone: 267-305-6521
Email: lisa_lin@merck.com
APPENDIX:

SAS code of using %do %while to do backward model selection in %MdSelect:

```sas
%MdStmt(resvar=&var, expvar=&intvar, clsvar=&catvar);
proc sql noprint;
   select max(ProbChiSq) into :pmax
       from pval;
quit;

%let i=1;
%do %while (&pmax>&slstay);
   proc sql noprint;
      select distinct parm into :varlist separated by ' ' 
           from pval
      having ProbChiSq^=max(ProbChiSq);
   quit;

   %MdStmt(resvar=&var, expvar=&varlist, clsvar=&catvar);
   proc sql noprint;
      select max(ProbChiSq) into :pmax
           from pval;
   quit;

   %let i=%eval(&i+1);
%end;
```

Coders' Corner
NESUG 2007