Paper SD10

# A SAS® MACRO FOR PERFORMING BACKWARD SELECTION IN PROC SURVEYREG

Qixuan Chen, University of Michigan, Ann Arbor, MI
Brenda Gillespie, University of Michigan, Ann Arbor, MI

## ABSTRACT

This paper describes a macro to do backward selection for survey regression. At each step of backward elimination, p-values are calculated by using PROC SURVEYREG. The p-values are sorted, and the maximum is compared to the staying criterion, which is defined by users, to decide if the corresponding variable should be eliminated. Users are required to provide a list of variables that should be included in all models and a list of variables that could be eliminated during the backward elimination procedure. This macro is also capable of handling categorical variables. In the process of explaining the macro code, we will review three ways of creating macro variables and the following SAS® macro functions: %scan, %str, %eval, and %unquote.

## INTRODUCTION

Although many variables selection methods are available in other SAS® procedures, like REG, LOGISTIC, PHREG, none is available in PROC SURVEYREG. The SURVEYREG procedure performs regression analysis for data with survey sampling weights. When the hypotheses of interest are known, the procedure fits linear models, calculates regression coefficients, and provides significance tests by accounting for the survey sample designs. However, sometimes the hypotheses of interest are unclear, and the pool of potential explanatory variables is big compared to the sample size. Some automatic procedures are in need to identify relatively important explanatory variables. A SAS macro was designed to do backward elimination for survey regression.

## EXPLANATION OF MACRO CODE

### Defining Macro Parameters

The macro has nine parameters that must be provided by the users when the macro is called:

- dataset = the name of the data set that contains the outcome, explanatory variables and all the sample design information, including sampling weights, strata and clusters
- forceInVar = the list of variables that are forced into all the models during the backward selection procedure
- varlist = the list of variables that could be eliminated during the backward selection procedure
- catVarlist = the list of categorical variables that are either in 'forceInVar' or in 'varlist'
- outcome = the name of the outcome variable in the survey regression
- weight = the name of the survey sampling weights in the complex survey design
- strata = the name of the survey sampling strata in the complex survey design
- cluster = the name of the survey sampling cluster in the complex survey design
- alpha = the criterion for retaining a variable in the backward elimination procedure

### The Backward Elimination Procedure

Before starting the backward elimination procedure, the number of variables in the original 'varlist' is counted, denoted as 'nvar'. In addition, we calculate 'nForceInVar', the number of variables in the 'forceInVar', and create a series of macro variables, called forceInVar1, forceInVar2, …, forceInVar&nForceInVar.

At each loop when there is at least one variable in the 'varlist', PROC SURVEYREG is run by putting all the forced in variables and the variables in the 'varlist' at that iteration. The individual p-values for continuous variables and the p-values of the overall F-tests for every categorical variables are obtained from the ODS output and saved in a data set, namely 'out'. Next, all the variables in the 'forceInVar' are removed from the data 'out'. The p-values are sorted among the rest of variables. The maximal p-value is compared to the staying criterion 'alpha'. If the maximal p-value is greater than 'alpha', the corresponding variable is eliminated from the 'varlist', and 'nvar' is subtracted by one, else the backward elimination

procedure ends. The loops continue until either 'nvar' equals to zero or the highest p-value is less than or equal to 'alpha'.

The list of categorical variables is not updated during the backward elimination procedure, since extra categorical variables in the CLASS statement of PROC SURVEYREG does not have any effects on the parameter estimates or the significance tests.

The parameter estimates in the final model are saved in the data set 'estimate'. Note that if no significant variables are chosen, the last variable to be dropped is retained in the data set 'estimate'.

**EXAMPLE OF MACRO USE**

In the data set 'example', there are 17 variables and 50 observations. The variables include Y, X1-X3, Z1-Z10, weight, strata and cluster. Y is the outcome, X1-X3 are known predictors, and Z1-Z10 are 10 potential explanatory variables. X1, Z3, Z4 and Z9 are categorical variables. The goal of the study is to identify if any variables among Z1-Z10 are important explanatory variables to Y. Additionally, we want to get estimates for X1-X3 after adjusting for other explanatory variables for Y, but without losing too many degrees of freedom.

First, download and save the %backward macro definition in your system. In your SAS program, specify this statement to define the %backward macro and make it available for use:

```
%inc "<location of your file containing the backward macro>";
```

Next, define the 'forceInVar', 'varlist' and 'catVarlist' by using %LET macro function.

```
%let forceInVar = X1 X2 X3;
%let varlist = Z1 Z2 Z3 Z4 Z5 Z6 Z7 Z8 Z9 Z10;
%let catVarlist = X1 Z3 Z4 Z9;
```

Finally, call the macro:

```
%backward (example, &forceInVar, &varlist, &catVarlist, Y, weight, strata, cluster, 0.1);
```

**CONCLUSION**

The %backward macro is useful when there are many potential predictors. The programming time is short. The macro screens relatively important explanatory variables quickly. Moreover, the backward macro is short and easily understandable. It can also be easily extended to the backward selection procedure of any other statistical regression procedure, for example, the SURVEYLOGISTIC procedure.

**MACRO CODE**

The complete SAS® code for the backward selection macro is provided below:

```
%macro backward (dataset, forceInVar, varlist, catVarlist, outcome, weight, strata, cluster, alpha);

%let nvar = 0;
%do %while (%scan (&varlist, %eval(&nvar+1), %str( )) NE);
   %let nvar = %eval (&nvar+1);
%end;
%let nForceInVar = 0;
%do %while (%scan (&forceInVar, %eval(&nForceInVar + 1), %str( )) NE);
   %let nForceInVar = %eval(&nForceInVar + 1);
   %let forceInVar&nForceInVar = %scan(&forceInVar, &nForceInVar);
%end;

%do %while (&nvar > 0);
   proc surveyreg data = &dataset;
      weight &weight;
      strata &strata;
      cluster &cluster;
      class &catVarlist;
```

```
    model &outcome = &forceInVar &varlist /solution adjrsq;
    ods output Effects = out;
    ods output ParameterEstimates = estimate;
  run;

  data result; set out;
    if Effect in ('Model', 'Intercept', %do i=1 %to %eval(&nForceInVar-1);
                                          %unquote(%str(%'&&forceInvar&i%')),
                                        %end;
                                       %unquote(%str(%'&&forceInVar&nForceInVar%'))) then delete;
    keep Effect ProbF;
  run;
  proc sort data = result;
    by descending ProbF;
  run;

  data result; set result;
    if _N_ = 1 then do;
      call symput('pvalue',ProbF);
      if ProbF > &alpha then delete;
    end;
    drop ProbF;
  run;

  %if &pvalue < &alpha %then %goto exit;

  proc sql noprint;
  select Effect into :varlist separated by ' ' from result;
  quit;

  %let nvar = %eval(&nvar-1);

%end;

%exit: %mend backward;
```

**A REVIEW OF MACRO FEATURES USED IN MACRO BACKWARD**

**Creating Macro Variables**

There are several ways to create macro variables. In this macro, we use %LET, CALL SYMPUT, and PROC SQL to create macro variables.

- Use %LET to produce a macro variable, and give it a value.
  EXAMPLE 1:
  ```
  %let nvar = 0;
  ***display the value of the macro variable in the LOG***
  %put &nvar;
  0
  ```

- Use CALL SYMPUT to store the value of a given variable from a data set.
  EXAMPLE 2:
  ```
  data result;
    input Effect $ ProbF;
    datalines;
    age   0.501
    BMI 0.001
    sex   0.002
    ;
  ```

```
run;
data result; set result;
    if _N_ = 1 then CALL SYMPUT('pvalue', ProbF);
run;
***display the value of the macro variable in the LOG***
%put &pvalue;
0.501
```

- Use SELECT INTO in PROC SQL to store the values of a variable from a data set as the text of a macro variable

```
Continued on EXAMPLE 2:
PROC SQL noprint;
select Effect into :varlist separated by ' ' from result;
quit;
***display the value of the macro variable in the LOG***
%put &varlist;
age BMI sex
```

**Functions Used in the Macro**

- Use %SCAN function to search an argument and return the nth word.

```
Continued on EXAMPLE 2:
%let var1 = %scan(&varlist, 1, %str( ));
***display the value of the macro variable in the LOG***
%put &var1;
age
```

- Use %STR function to specify a blank as the delimiter in %SCAN function or for character strings that contain a quotation mark.

```
Continued on EXAMPLE 2:
%let var2 = %str(%'&var1%');
***display the value of the macro variable in the LOG***
%put &var2;
'age'
```

- Use %UNQUOTE function to unmask the value of a macro variable that is assigned using a macro quoting function. If the value is not unmasked, when the variable is referenced, it produces error message.

```
Continued on EXAMPLE 2:
*** The following program generates error messages in the LOG because the value of var2 is still masked when it reaches the SAS® compiler. ***
data result; set result;
   if Effect = &var2 then delete;
run;

***This version of the program runs correctly***
data result; set result;
   if Effect =%unquote(&var2) then delete;
run;
```

- Use %EVAL function to evaluate integer arithmetic.

```
Continued on EXAMPLE 1:
%let nvar = %eval(&nvar + 1);
***display the value of the macro variable in the LOG***
%put &nvar;
1
```

**ACKNOWLEDGEMENT**
The authors acknowledge Dr. David Garabrant, Dr. Al Franzblau, Dr. Peter Adriaens, Dr. Avery Demond, Dr. James Lepkowski and the University of Michigan Dioxin Exposure Study for supporting the project and Kathy Welch for her continued SAS® assistance.

**REFERENCES**

1. *SAS® Macro Language: Reference, Version 8*, Cary, NC: SAS Institute Inc.
2. Mike S. Zdeb, *Creating Macro Variables Via PROC SQL*, NESUG, Washington, DC 1999

**CONTACT INFORMATION**
Please send your comments and questions to:

Qixuan Chen
The Department of Biostatistics
School of Public Health
University of Michigan
1420 Washington Heights
Ann Arbor, MI 48109-2029
Email: qixuan@umich.edu