A Recursive SAS Macro Technique and its Application to Statistics

Conference Paper · August 2003

CITATIONS
0

READS
300

1 author:

Yohji Itoh
A2 Healthcare
56 PUBLICATIONS **2,875** CITATIONS

SEE PROFILE

SUGI-J 2003, 31 August 2003

A Recursive SAS Macro Technique and its Application to Statistics

Yohji Itoh
Statistics & Programming Department, AstraZeneca K.K., Osaka, Japan

ABSTRACT

The recursive call of modules is an important function for iterative computation and it is avalilable in some programming languages. However, the SAS System does not provide the recursive call facility, thus users need to take other approaches. Here a new technique called “recursive macro call” is proposed. This technique utilizes CALL EXECUTE routine, thus the explanation about it is given in the first place, and then the recursive macro technique is derived. This technique is very useful especially for statistical iterative computation because it enables us to use powerful SAS procedures iteratively. As an example of its application to statistics, the power-of-the-mean model using PROC MIXED is illustrated.

Key words: recursive call, macro, CALL EXECUTE

1. INTRODUCTION

In some programming languages, such as Pascal and PL/I, a recursive call of subroutines is available. The recursive call technique enables us to write a subroutine that can call itself and sometimes makes iterative processing very easy. However, the SAS System does not provide a recursive call facility, thus users need to take other approaches. Benjamin (1999) proposed "pseudo-recursive" SAS macros, by which he simulated a "run-time stack" for recursive processing adopted by other languages, but his approach is very complicated and needs special knowledge, so it is difficult to use for general programmers.

Here, a new approach for a recursive macro call is proposed. It is very simple and does not require any special knowledge about the recursive subroutine call.

First, a general problem that SASA macros have will be presented and the use of "CALL EXECUTE" will be explained as a tool for covering the defect. Next, based on this technique, the "recursive macro call" is derived. An example of its application to iterative computation in statistics is presented, i.e. the power-of-the-mean model utilizing PROC MIXED.

2. CALL EXECUTE

CALL EXECUTE is a DATA step call routine and it is well described by Riba (1997). It is used as shown in the following program. An argument to be specified for CALL EXECUTE should be a character string consisted of SAS statements and it can be either of a character constant or a character variable.

Program 1

```
data …;
   …
   call execute('SAS statemants');
   …
run;
```

The knowledge of the process flow of CALL EXECUTE is important for understanding the "recursive macro", so it is explained in this section. Before explaining it, however, it is worth explaining the process flow of usual SAS programs.

Figure 1 illustrates the flow of the execution of a usual SAS program. When program statements are "submitted", they are not compiled immediately by the SAS System. Instead they are first stored in a "program stack". When a RUN statement or the next step is detected, the SAS statements stored in the program stack are compiled and then executed.

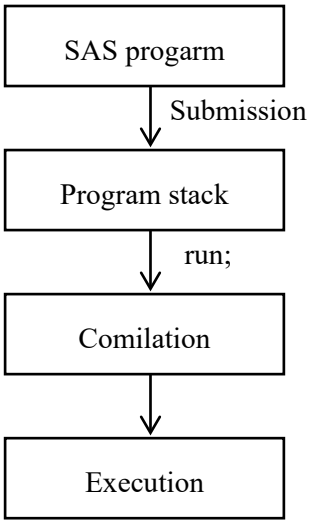

Figure 1. Process flow of usual SAS progorams

The knowledge of the process flow of CALL EXECUTE is important for understanding the "recursive macro call", so it is explained in this section. Before explaining it, however, the process flow of usual SAS programs is explained first.

Figure 2 illustrates the flow of the execution of a SAS program that includes CALL EXECUTE statement. Until the start of the execution of the program, the process is the same as a usual program that does not include CALL EXECUTE. When the CALL EXECUTE is executed, its argument ('abc' in Figure 2) is stored in the program stack. When the execution of the program is finished, the control is moved to the program statements stored in the program stack, and then they are compiled and executed.

An interesting feature of this processing is that the program statements generated by CALL EXECUTE are not compiled until the execution of the DATA step is completed. This feature is very important when you want to execute a macro conditionally using CALL EXECUTE. This topic is discussed later.

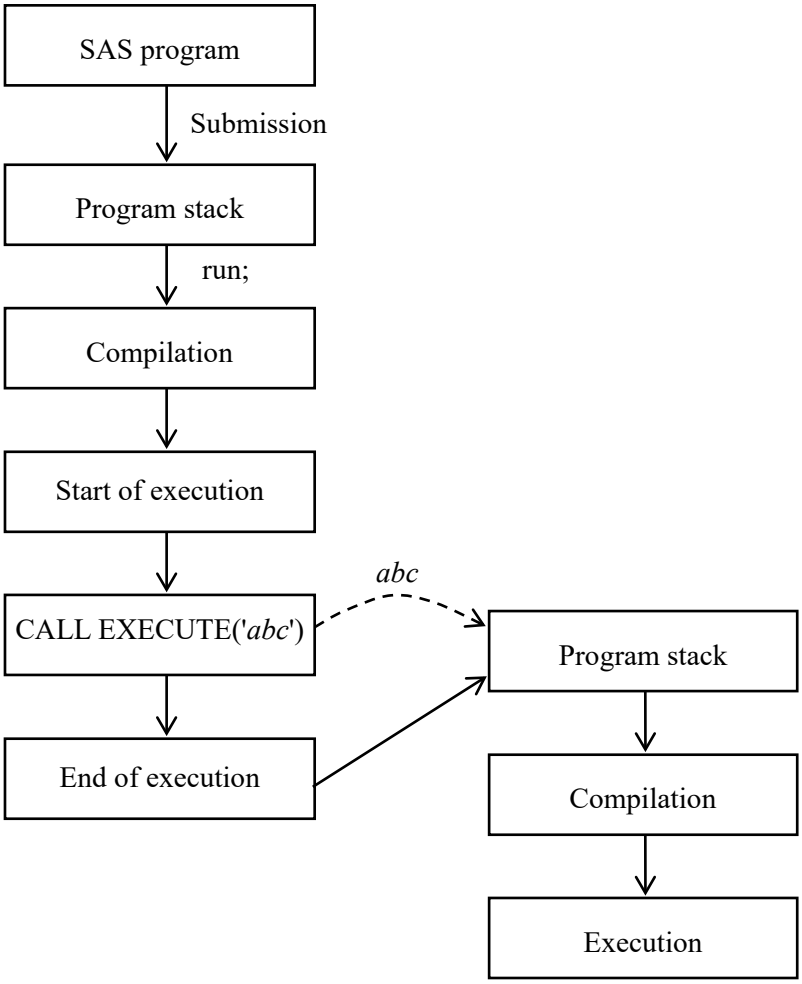

Figure 2. Process flow of SAS programs including CALL EXECUTE

3. PROBLEM OF MACRO PROCESSING

When the following program is submitted, one may expect that macro "%macrox" will be resolved depending on the result of the condition of the IF statement, that is, it will be resolved using argument "a" if the condition is TRUE, or resolved using argument "b" if it is FALSE.

Program 2

```
data ...;
  ...
   if (condition) then %macrox(a);
                  else %macrox(b);
  ...
run;
```

However, this will not occur, because the SAS macro processor resolves the macro before the execution of the DATA step as shown Figure 3. Therefore, the macros are already been resolved at the start of the execution of the DATA step. So in general the SAS macro resolution cannot be changed depending on the results of program execution as described by Riba (1997).

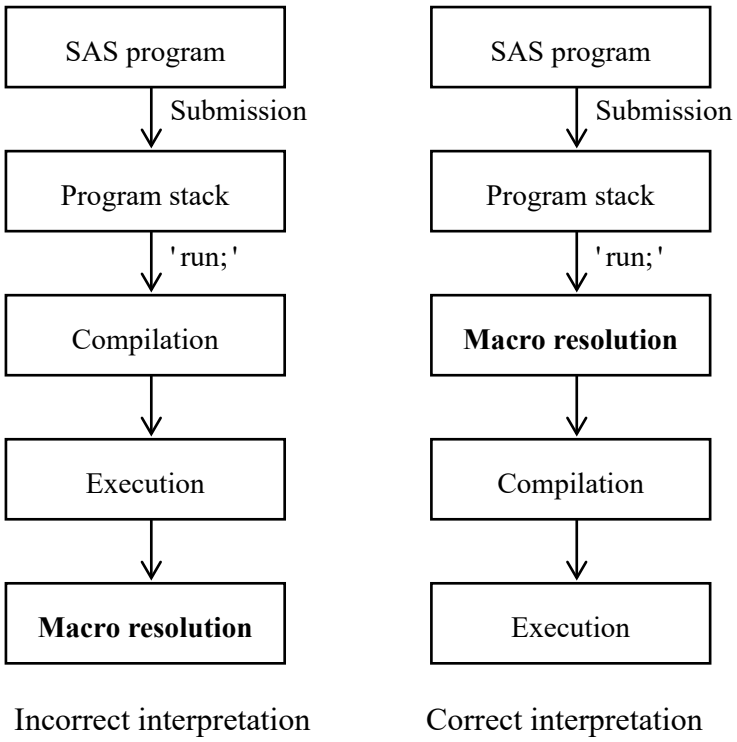

Figure 3. Interpretation of process flow of Program 2

4. MACRO INVOCATION BY CALL EXECUTE

If we use CALL EXECUTE, we can change macro resolution depending on the results of program execution. In the following program, text '%macrox(a)' will be stored in the program stack if the condition is TRUE, and '%macrox(b)' will be stored in the program stack if the condition is FALSE. These texts stored in the program stack will not be compiled until the execution of this DATA step is finished. So in this case, we can control the macro resolution depending on the condition of the IF statement.

Program 3

```
data ...;
   ...
   if (condition) then call execute('%macrox(a);');
                  else call execute('%macrox(b);');
   ...
run;
```

5. RECURSEIVE MACRO CALL

If we extend this idea, we can construct a macro program that can call itself. Program 4 illustrates this idea. In this program, macro %mcrx calls itself by CALL EXECUTE. When the execution of the macro is finished, the macro itself called by CALL EXECUTE will be compiled and executed, so the macro program can be executed recursively. If the condition of the IF statement is FALSE, the macro will not be called, so the process will finish.

Similar service of recursive processing is available in other languages like PL/I and PASCAL, and it is called "recursive call" of subroutines. Although such service is not available in the original SAS system, it is possible even in the SAS System if we use CALL EXECUTE. We call the method described here "recursive macro call" after other computer languages.

Program 4

```
%macro mcrx;
   ...
  data ...;
    ...
    if (condition) then call execute('%mcrx;');
    ...
  run;
  ...
%mend;

%mcrx;
```

6. APPLICATION OF RECURSIVE MACRO TO STATISTICS - Power-of-the-mean models

6.1 What is “Power-of-the-mean models”?

We will show an application of the recursive macro call to a statistical problem, the power-of-the-mean model (Carroll & Ruppert, 1988, Littell et al., 1996). In this model, the error variance of an observation is assumed to be proportional to the power of the expected valued of the observation, that is, if we assume a linear model, the variance of the i-th individual is expressed as:

$$\sigma_{e_i}^2 = \sigma^2 \mid \mathbf{x}_i' \boldsymbol{\beta} \mid^{\theta}$$

where σ^2 is a variance parameter,

$\mathbf{x}_i'$ is the i-th row of design matrix $\mathbf{X}$,

$\boldsymbol{\beta}$ is a vector of fixed effects, and

θ is a power coefficient to be estimated.

In SAS, the power-of-the-mean model can be specified by LOCAL=POM option in REPEATED statement in PROC MIXED (Littell et al., 1996).

REPEATED /LOCAL=POM(*SAS datasets*);

In the parentheses after POM, we should specify a SAS dataset in which some values of fixed effects, like regression coefficients, are included. Thus, in this model, the values of the fixed effects are assumed to be known. In practice, however, we usually have to estimate them from data. Once we obtain the estimates, we can apply the power-of-the-mean model using the estimates to compute new estimates for the fixed effects and the power coefficient. Furthermore, we can use these estimates to compute new estimates again, and if we repeat this process, we may obtain more reliable estimates.

6.2 NUMERICAL EXAMPLE

To illustrate the power-of-the-mean model, we use the data given by the following program:

Program 5

```
data doseres;
   input dose @;
   do i=1 to 10;
      input res @;
      output;
   end;
   keep dose res;
cards;
    1   9.2   6.8  10.0  12.4   9.2  11.6  10.9   7.1  12.9   5.9
    2  23.9  24.8  23.9  19.6  18.3  12.7  10.7  18.4  17.1  21.1
    3  26.0  22.5  36.9  27.8  29.0  30.8  23.3  39.8  29.9  16.1
    4  44.6  47.7  30.2  55.4  18.8  40.0  39.4  55.5  28.4  38.0
;
run;
```

This data consists of two variables DOSE and RES. We would like to know the dependence of RES on DOSE. The graph below shows the relationship of the two variables. The graph suggests that the regression of RES on DOSE may be linear, but the variance of RES increases depending on DOSE. These features of data distribution suggest that the power-of-the-mean model may fit the data.

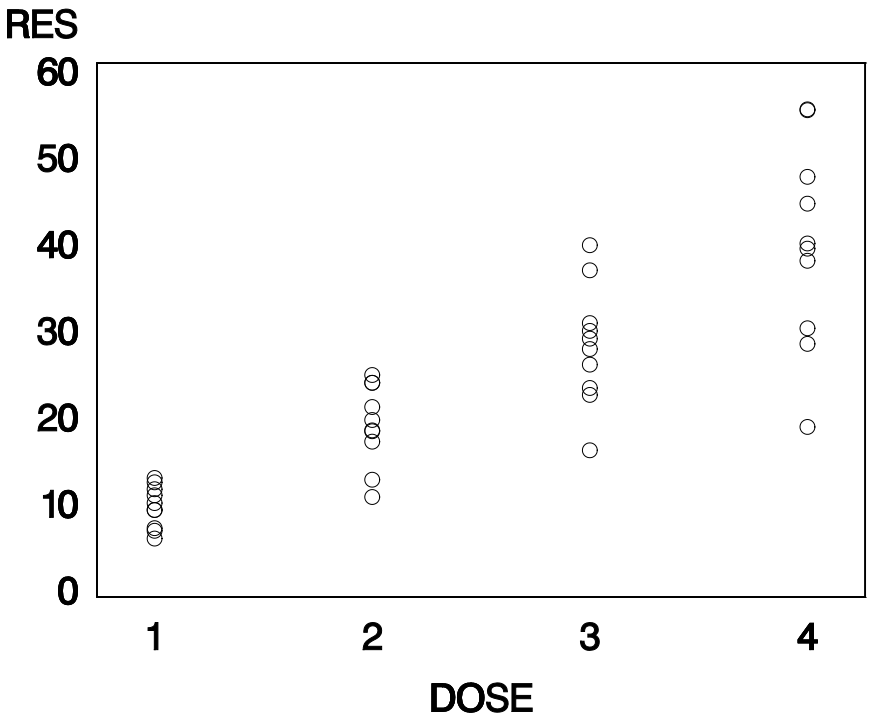

Figure 4. Plots of numerical example for the. Power-of-the-mean model

6.3 SAS PROGRAM FOR ITERATIVE CALCULATION

Program 6 shows a basic idea of analyzing this data by the power-of-the-mean model using PROC MIXED. This program consists of two steps.

Program 6

```
ods output solutionf=solf1;
proc mixed data=doseres;
   model res=dose / s;
run;

ods output solutionf=solf2;
proc mixed data=doseres;
   model res=dose / s;
   repeated / local=pom(solf1);
run;
...
```

The first step is a preliminary step that provides the estimates of regression parameters based on a regression model with the homogeneous variance assumption and these estimates will be stored in dataset "solf1."

In the next step, "repeated / local=pom(solf1)" is specified, so estimation is done based on the power-of-the-mean model. In this calculation, the estimates obtained in the first step are used to compute new estimates that will be stored in dataset SOLF2. Furthermore, we can execute PROC MIXED again using these estimates, if we want. If we repeat these steps and the regression parameter values converge, finally we will get the estimates that we want.

However, it is tedious to repeat these steps manually, so we hope that the iteration can be conducted automatically. This automatic iterative computation is possible if we use the technique described in the previous section. Program 7 shown below enables us to compute the power-of-the-mean model automatically.

Program 7

```
%let reg=1;

%macro pom(first);
  ods output solutionf=solf2;
  proc mixed data=doseres;
    model res=dose / s;
    %if &first^=1 %then repeated/ local=pom(solf1);;
  run;
  data solf1;
    set solf2;
    if effect='dose'
       and abs(estimate - &reg)>1e-8 then do;
       call symput('reg',left(put(estimate,e17.10)));
       call execute('%pom();');
    end;
  run;
%mend;

%pom(1);
```

The first %LET statement gives the initial value of 1 to the macro variable "reg" that stores the estimate of regression coefficient tentatively. This macro variable will be used for the judgment of convergence in a later step.

The next is macro program "pom" that consists of two steps of PROC MIXED and DATA step.

Macro program %pom has macro parameter "first." When it has the value of 1, it means the first cycle of the iteration. In that case, the condition of

the %IF statement inserted into the MIXED procddure is not TRUE, so the REPEATED statement is not specified, then the computation will be done based on a homogeneous variance model. If the value of &fist is not 1, “repeated/ local=pom(solf1)” will be specified, then the computation based on the power-of-the-mean model will be done. The resulting estimates of regression parameters will be stored in dataset “solf2.”

In the next data step, the convergence of the regression coefficient is judged. The new estimate of the regression coefficient is read from data set “solf2,” and then it is compared with the previous value stored as macro variable “reg.” If the absolute value of the difference between them is larger than 10^{-8}, then the convergence is considered not to be reached and the preparation for the next cycle will be done, i.e. replacement of the macro variable by the new regression coefficient and a recursive call of the macro by "call execute('%pom();');". In this case, the argument is not specified for %POM, so it means that the power-of-the-mean model is specified in the next cycle. If the difference of the new regression coefficient and the previous one is not greater than 10^{-8}, the convergence is assumed to be reached. In this case, the macro is not called, so the iteration will be stopped.

The results of iteration are shown in Table 1. The third column in this table represents the change in the regression coefficient between two consecutive cycles. The regression coefficient converged at the 6th cycle where the change is less than 10-8. The last column represents the estimated power coefficient, which also seems to have converged.

The last line is for the initial call of macro “pom.”

The results of iteration are shown in Table 1. The third column in this table represents the change in the regression coefficient between two consecutive cycles. The regression coefficient converged at the 6th cycle where the change is less than 10−8. The last column represents the estimated power coefficient, which also seems to have converged.

Table 1. Iteration history of numerical example of power-of-the-mean model

Iteration	Regression coefficeint	Change	Power coefficient (θ)
1	9.9760000000	-	-
2	9.6919601675	-0.2840398325	2.0816386164
3	9.6898237032	-0.0021364643	2.1587574892
4	9.6898171411	-0.0000065621	2.1590014405
5	9.6898171209	-0.0000000202	2.1590021894
6	9.6898171209	-0.0000000000	2.1590021917

7. GENERAL ALGORITHM FOR ITERATIVE COMPUTATIONS

The algorithm for the iterative computation presented in the previous section can be expressed in a general form schematically as Program 8.

In the iterative computations using recursive macros as shown here, we need to use a macro variable for controlling the iteration; we need a %LET statement for the initialization of it, and CALL SYMPUT for the renewal of it in each cycle.

Program 8

```
%let macro variable = ...;                          (1)

%macro macro name;
  ...
  data ...;
    ...
    if (more iterations needed? ) then do;          (2)
      call symput('macro variable',variable);       (3)
      call execute('macro name');                   (4)
    end;
  run;
%mend;

%macro name;                                        (5)
```

Note:

(1) Initialization of a macro variable
(2) Judgment for convergence by referring the macro variable
(3) Replacing the macro variable by the new result
(4) Recursive macro call
(5) Initial macro call

8. CONCLUSION

In this paper, the recursive macro using CALL EXECUE was presented. It is a powerful technique especially for iterative computations in statistics.

Iterative computations are also possible by a DO loop of DATA-step programming or IML, but we have to write all program codes for computing algorithms. So, the iteration by the DO loop is restricted to cases where algorithms are simple.

On the other hand, the recursive macro technique can utilize powerful SAS procedures. This point is the most useful aspect of this technique. For example, PROC MIXED was used in the example in

this paper. In statistics, iterative computations are necessary in various problems, i.e. estimations in nonlinear models, the EM algorithm in incomplete data (Dempster et al., 1977), etc. Some of them may be computed by iterative execution of existing SAS procedures, then the recursive macro technique is very useful for these cases.

References

Benjamin, W. E. Jr. (1999), A Pseudo-Recursive SAS Macro, Observation, 07MAY1999, obswww18. (http:// support.sas.com/documentation/periodicals/obs/obswww18/index.html)

Carroll, R. J., Ruppert, D. (1988), Transformation and weighting in regression, Chapman & Hall

Dempster, A. P., Laird, N. M. and Rubin, D. B. (1977), Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society, Series B 39, 1-38.

Littell, R. C., Milliken, G. A., Stroup,W. W., Wolfinger, R. D. (1996), SAS System for Mixed Models, SAS Institute Inc., Cary, NC

Riba, S. D., (1997), Self-modifying SAS programs: a DATA step interface. Obsevation, 07SEP1997, obswww03 (http://support.sas.com/documentation/periodicals/obs/obswww03/toc.html)