Paper 211

A Taste of SAS New Enhancements

Jiang Jin
EDP Contract Services

Abstract

A SAS® coding job can be done differently by various SAS programmers in different levels. Those who catch SAS new enhancements can always produce more efficient SAS codes.

The purpose of this paper is to present the flexibility of SAS New Enhancements by a group of examples which include using SYMPUT ROUTINE, SAS %SYSFUNC( ), host variables from PROC SQL, and CALL EXECUTE( ). It demonstrates the advantages and the differences between SAS New Enhancements and normal SAS coding technique.

Introduction

One SAS job sounds even very easy to a SAS novice:

Data set ONE has variable PATIENT, AA, BB, CC, DD, EE, FF, GG, HH

PATIENT	AA	BB	CC	DD	EE	FF	GG	HH
01	1	2	3	4	5	6	7	8
02	1	2	3	4	5	6	7	8
03	1	2	3	4	5	6	7	8
04	1	2	3	4	5	6	7	8

Using PROC REPORT to generate listings including PATIENT and one other variable at a time, the output would look like:

Listing for Variable AA

PATIENT	AA
01	1
02	1
03	1
04	1

Listing for Variable BB

PATIENT	BB
01	2
02	2
03	2
04	2

……

Listing for Variable HH

PATIENT	HH
01	8
02	8
03	8
04	8

From novices to veterans, six methods can be used as follows

Method 1. Novices

This is not a tough job for new SAS programmers.

```
options formdlim=' ' nodate  nonumber;
 proc report data=one nowd headline headdskip;
 col patient aa;
 define patient /display ;
 define aa /order;
 title '  Listing for Variable AA ';
 run;
 proc report data=one nowd headline headskip;
  col patient bb;
  define patient /display ;
  define bb /order;
  title "  Listing for Variable BB ";
   run;
```

One could use PROC REPORT eight times for all those variables. However, the problem becomes very obvious that the code is too long, and it could turn out to be a pain if there are too many variables in the data.

Method 2. Starting with MACRO.

```
%macro report2(var=);
proc report data=one nowd headline headskip;
  col patient &var;
  define patient /display ;
  define &var /order;
  title "  Listing for Variable &var ";
  run;
%mend report2;

%report2(var=AA); %report2(var=BB);
%report2(var=CC); %report2(var=DD);
%report2(var=EE); %report2(var=FF);
%report2(var=GG); %report2(var=HH);
```

Method 1 to Method 2 is a big jump. SAS Macro Language is another system of SAS. Using SAS macros, it will save a lot of time if it is a routine job or the task needs to be repeated several times. SAS macros are being used every day, and they can be applied in Data steps, Procs, SQL, SCL, etc. To explore SAS Macro, many references can be found from SAS Institute or SAS user books.
In method 2 Typing and calling %report2 for each variable are still tedious.

Method 3. Using CALL SYMPUT( )

```
proc contents data=one(drop=patient)
out=two(keep=name) noprint; run;

data _null_;
set two end=eof;
call symput("var"||left(put(_n_, 2.)),  name);
 if eof then
 call symput('num', _n_);
 run;

%macro report3;
%local i;
  %do i=1 %to &num.;
proc report data=one nowd  headline headskip;
 col patient &&var&i;
 define patient /display ;
 define &&var&i /order;
 title "  Listing for Variable  &&var&i ";
  run;
 %end;
 %mend report3;
 %report3;
```

PROC CONTENTS outputs variables needed to report as values of NAME.
CALL SYMPUT( ) transfers these values to &var1 - &var8.
Eight PROC REPORT are submitted automatically in the loop of REPORT3.

In many applications, macro variables are assigned based on: 1) data values stored in files (SAS data sets or external files) or 2) programming logic or computed values.
The SYMPUT ROUTINE is one of the most commonly used methods creating macro variables and assigning them values.
Combining the features of SYMPUT ROUTINE and _n_ to generate a sequence of macro variables is a nice tip for many SAS users.

Method 4 Host variables from PROC SQL

```
%macro report4;
 proc sql noprint;
 select name into: variable  separated by ","
 from  two;
 quit;
 %let varr="&variable";
 %let i=1 ;
%do %while ( %scan(&varr, &i, ",") ~=
                                  %str() );
 %let var=%scan(&varr, &i, ",");
 proc report data=one nowd  headline headskip;
   col patient &var;
   define patient /display ;
   define &var /order;
    itle "  Listing for Variable &var ";
  run;
%let  i=%eval(&i+1);
%end;
%mend report4;  %report4;
```

PROC SQL is widely accepted by SAS programmers all over the world, especially for those with some database experiences. Tons of papers and references can be found to show the advantage of PROC SQL.
PROC SQL's ability to assign the whole observations of a variable to a single macro variable benefits SAS programmers who use PROC SQL.
The results of the assigning is &var = AA,BB,CC,DD,EE,FF,GG,HH.
PROC REPORT is submitted eight times by the loop in %report4.

Method 5. Using %SYSFUNC( )

```
%macro report5;
%let
dsid=%sysfunc(open(one(drop=patient), i));
%let varlist=;
  %do i = 1 %to
       %sysfunc(attrn(&dsid, nvars));
    %let varlist=&varlist
      %sysfunc(varname(&dsid, &i));
  %end;
 %let i=1 ;
 %do %while ( %scan(&varlist, &i, ",") ~=
                                    %str() );
%let var=%scan(&varlist, &i, ",");
 proc report data=one nowd  headline headskip;
  col patient &var;
  define patient /display ;
  define &var /order;
  title "  Listing for Variable &var ";
  run;
%let  i=%eval(&i+1);
%end;
%mend report5;
%report5;
```

SAS macro function %SYSFUNC and %QSYSFUNC are relatively new to a lot of SAS users. The complete reference can be found in SAS web site:
http://www.sas.com/service/techsup/intro.html

The difference of this method from the other ones is without referring output from PROC CONTENTS. The macro variable

&varlist=AA BB CC DD EE FF GG HH

is generated by a loop of %SYSFUNC. This is an alternative way of getting host variable from PROC SQL.

Method 6. Using CALL EXECUTE( )

```
%macro report6(var=);
 proc report data=one nowd headline headskip;
 col patient &var;
 define patient /display ;
 define &var /order;
 title "  Listing for Variable &var ";
 run;
%mend report6;
data _null_;
set two;
call execute('%report6 ( var = '|| name ||')'); run;
```

CALL EXECUTE is another relatively new DATA Step function which interacts with the SAS Macro Facility. Character data can be sent to the macro facility for immediate macro execution during the execution of the DATA Step. Completed reference can be found in Technical Report P-222 Change and Enhancement to Base SAS Software.
SAS executes %report6 at each step of SET TWO, and performs PROC REPORT for AA, BB, CC, DD, EE, FF, GG, and HH.

Conclusion

Examples have been given to show the flexibility of SAS New Enhancements in coding and potential benefit it provides to us. Comparing to first 3 methods, method4, method5, and method6 are relatively new to many SAS users. Further comparison of efficiency between regular methods and methods using SAS NE could expand our discussion in much more details. For instance computer CPU time used by these method could be an index of the comparisons.

SAS continually provides its enhancements. We can always take the advantages of SAS New Enhancements by paying our attention to it.
The problem presented in the paper can be solved by many other ways.
The author can be contacted by mail

Jiang Jin
134 Pepper Drive
Collegeville, PA 19426

or e-mail

jiang_2_jin@sbphrd.com |
danjj@worldnet.att.net.

Reference

SAS Institute, Inc., SAS Technical Report P-222, Changes and Enhancements to Base SAS Software, Release 6.07, Cary, NC.

SAS Guide to the SQL Procedure, Usage and Reference, Version 6, First Edition.