Paper 198-2013

Using CALL SYMPUT to Generate Dynamic Columns in Reports

Suwen Li, Everest Research Services Inc., Markham, ON.
Sai Ma, inVentiv Health Clinical, Burlington, ON
Regan Li, Hoffmann-La Roche Ltd., Mississauga, ON
Bob Lan, Everest Research Services Inc., Markham, ON.

ABSTRACT

When creating reports, we often want to make the report respond dynamically to data. If the headers and the number of columns in the report are unknown, it is helpful when they change dynamically depending on the data. As a powerful SAS® procedure, PROC TABULATE can produce dynamic results in most cases. This paper describes how to use the CALL SYMPUT routine and PROC REPORT to generate dynamic columns in reports in cases where PROC TABULATE does not yield the desired results.

INTRODUCTION

PROC REPORT and PROC TABULATE are two of the most common SAS procedures used to create reports. The biggest advantage of PROC TABULATE against PROC REPORT is its ability to adjust reports to data. If a new value of a classification variable is added to the data, PROC TABULATE will adjust the report automatically. The new category can be added in the report as a new column. An appropriate column header also can appear if the category variable is formatted.

This allows the SAS code to remain applicable even if the categories in the data are changed. Unlike PROC TABULATE, PROC REPORT is lacking that capability. When the categories are changed, the SAS codes must be changed manually in COLUMN and DEFINE statements in PROC REPORT. However, the disadvantage of PROC TABLUATE is it can only generate standard summary tables. It lacks capability to produce non-standard reports or reports with a complicated layout, which, in contrast, is the strength of PROC REPORT. Combined with CALL SYMPUT, we can use PROC REPORT to generate dynamic columns in reports similar to PROC TABULATE but for very complicated tables. This paper will introduce CALL SYMPUT routine and discuss how to use PROC REPORT generate reports dynamically adjusting to data.

CALL SYMPUT ROUTINE

CALL SYMPUT is a DATA step call routine that assigns a value produced in a DATA step to a macro variable. The syntax is CALL SYMPUT (macro-variable, value) [1]. The macro variable takes three forms. The first is a character string that is a character string enclosed in quotation marks. For example, the statement *call symput('new','testing')* will assign the character string “testing” to a macro variable NEW. The second type of macro variable form is the name of a character variable in a DATA step. It will create a series of macro variable names from that variable’s values. For example, in the *call symput(position,player)* statement, both POSTION and PLAYER are variable names. In each data iteration of DATA step, a macro variable is created from the value of POSTION, and the value of that macro variable is determined by the value of the second variable PLAYER. The third form is a character expression that produces a series of macro variables. For example, the *call symput ('POS'||left(_n_), position)* will create a series of macro variables &POS1, &POS2, etc. The value of each macro variable is the value of POSITION corresponding to the number of times the DATA step has iterated. Thus, CALL SYMPUT is a dynamic DATA step interface tool to assign variable values in DATA steps to macro variable. We will discuss how to use this feature to produce flexible tables and listings.

PROC TABULATE

There are a lot of papers discussing the features and usage of PROC TABULATE. The example below simply illustrates the advantage of PROC TABULATE in generating reports automatically adjusting to data.

```
proc tabulate data=popl;
  class treatc;
  var age;
  table age*(mean std), treatc;
run;
```

```
 ____________________________________________________________________
|                              |                TREATC                |
|                              |______________________________________|
|                              |     ABC     |     DEF     |  PLACEBO   |
|______________________________+____________+____________+____________|
|Age              |Mean        |       37.11|       37.31|        36.8|
|                 |____________+____________+____________+____________|
|                 |Std         |       12.41|       12.12|       13.12|
 ____________________________________________________________________
```

The column headers are from the values of the variable TREATC defined in the CLASS statement. When the values of TREATC are changed, the column headers and number in SAS output will change automatically to reflect the change to treatment group in new data.

DEMOGRAPHIC SUMMARY TABLE BY TREATMENT

The demographic summary table is a very common table in clinical report. The template of this kind of table is always completely the same, but the columns always change depending on the study treatment group.

Because the report layout is complicated, it needs a few steps of calculations. Thus, PROC TABULATE cannot deal with this type of summary table. Instead, PROC MEANS, PROC FREQ and PROC REPORT are usually used together to create it. After simple calculation, the statistics of age and frequency distribution of gender are printed in below.

```
CLA    CLAC        TREAT  TREATC   _MEANC  _STDC  _MEDIANC  _MINMAXC      _NC
 1   Age (yrs)       0    Placebo   36.8   13.1     37.8     21, 52       14
 1   Age (yrs)       1    ABC       37.1   12.4     35.0     20, 58       14
 1   Age (yrs)       2    DEF       37.3   12.1     37.0     20, 58       20


CLA    CLAC     SUBCLA  SUBCLAC    TREAT  TREATC    COUNTC
 3   Sex, n(%)   12     Male        0     Placebo    3 (75.0)
 3   Sex, n(%)   12     Male        1     ABC        7 (50.0)
 3   Sex, n(%)   12     Male        2     DEF        8 (40.0)
 3   Sex, n(%)   13     Female      0     Placebo    1 (25.0)
 3   Sex, n(%)   13     Female      1     ABC        7 (50.0)
 3   Sex, n(%)   13     Female      2     DEF       12 (60.0)
```

The summarized results are then transposed using PROC TRANSPOSE.

```
 CLA  CLAC          _NAME_      _0              _1              _2
 1   Age (yrs)      _MEANC       36.8            37.1            37.3
 1   Age (yrs)      _STDC        13.1            12.4            12.1
 1   Age (yrs)      _MEDIANC     37.8            35.0            37.0
 1   Age (yrs)      _MINMAXC     21, 52          20, 58          20, 58
 1   Age (yrs)      _NC           4              14              20


 CLA CLAC       SUBCLA SUBCLAC      _0           _1            _2
 3   Sex, n(%)  12     Male        3 (75.0)     7 (50.0)      8 (40.0)
 3   Sex, n(%)  13     Female      1 (25.0)     7 (50.0)     12 (60.0)
```

The typical SAS program for this report is

```
proc report data=report2 nowd missing headline headskip spacing=3 split='@' ;
 column ('__' cla clac subcla _0 _1 _2      );
 define cla      /group order=internal noprint;
 define clac     /group order noprint;
 define subcla /order=internal width=54 ' '  flow left f=subcla. spacing=0;
 define _0       / width=16            flow "Placebo@(N=%left(&t1))";
 define _1       / width=16            flow "ABC@(N=%LEFT(&T2))";
 define _2       / width=16            flow "DEF@(N=%left(&t3))";
 break after clac / skip;
 compute before clac;
   line @1 clac $ left;
 endcomp;
run;
```

The program then can be used in different studies. The only thing that needs to be changed is the column headers which represent the different treatment groups. The below code, however, can dynamically generate the summary table and can be used in different studies without any update.

We used CALL SYMPUT to assign treatment groups to a series of macro variables and modify the above program so that it can be copied for use in different studies and update the report automatically without any SAS code modification.

```
proc sort data=out3 out=treat nodupkey;
  by treat;
run;

data _null_;
 set treat end=last;
 call symput('Treat'||left(put(_n_,best.)), left(put(treat,best.)));
 call symput('Treatc'||left(put(_n_,best.)), treatc );
 if last then do;
      call symput('total', left(put(_n_,best.)));
  end;
run;
```

The _null_ DATA step creates a series of macro variables. The character decode of the treatments are assigned to macro variables &TREATC1, &TREATC2, and &TREATC3. The numbers of the treatments are assigned to macro variables &TREAT1, &TREAT2, and &TREAT3. The total number of treatment is assigned to the macro variable &TOTAL. The below macro then will create the demographic summary table after the summarized data is transposed.

```
%macro report;
%if &total=2 %then %do;
  %let cwidth=39;
%end;

%if &total=3 %then %do;
  %let cwidth=26;
%end;

proc report data=report2 nowd missing headline headskip spacing=0 split='@' ;
  column ('__' cla clac subcla
          %do i=1 %to &total;
             _&&treat&i
          %end;
          );
   define cla      /group order=internal noprint;
   define clac     /group order noprint;
   define subcla /order=internal width=54 ' '  flow left f=subcla. spacing=0;

   %do j=1 %to &total;
     define _&&treat&j  / width=&cwidth   flow "&&treatc&j@(N=%left(&&t&j))";
   %end;

   break after clac / skip;
   compute before clac;
     line @1 clac $ left;
   endcomp;
run;

%mend;
%report;
```

```
________________________________________________________
                    Placebo       ABC           DEF
                    (N=4)         (N=14)        (N=20)

__________________________________________________________

Age (yrs)
    Mean              36.8          37.1          37.3
    SD                13.1          12.4          12.1
    Median            37.8          35.0          37.0
    Min, Max          21, 52        20, 58        20, 58
    n                  4            14            20

Sex, n(%)
    Male               3 (75.0)      7 (50.0)      8 (40.0)
    Female             1 (25.0)      7 (50.0)     12 (60.0)
```

CLINICAL LABORATORY RESULTS LISTING

A more complicated table in clinical study report is the list of laboratory data[2]. As laboratory results generally have a very large dataset, the listing table of laboratory data always has hundreds or thousands of pages. Thus, it is not easy for reviewer to obtain informative findings. To have an informative and neat list of laboratory results, test dates and visit per subject ID are requested to be presented in column headings. The desirable table is presented on page 6. The difficulty of this kind of layout is different subject IDs may have different test dates and some subjects may miss a few visits. Thus, the column will change by subjects and pages. For example, one subject may have only 5 columns, and one subject may have as many as 23 columns. Thus, the report will have only 5 columns in some page, but some may have columns more or less than 5. The column headings will change by subjects and pages also. Furthermore, the number of columns for each subject is unknown. It may change when the dataset is updated.

We used CALL SYMPUT, macro Do-loops, and PROC REPORT to generate the listing.

Below is a sample printout of the lab data ADLB.

```
SUBJID   PARMGRP     PARAM                  ADT        LBSTRESU    LBSTRESC

001     CHEMISTRY   Protein, Total     2011-08-11     g/L          77.0
001     CHEMISTRY   Bilirubin, Total   2011-08-11     umol/L       1.81
001     CHEMISTRY   Triglycerides      2011-08-11     mmol/L       2.21
```

The program generating the final report is as follows:

```
proc sort data=adlb ;
  by subjid parmgrp adt;
run;
```

Only keep unique visit date for each lab test category and subject in dataset LBDT

```
proc sort data=adlb out=lbdt(keep=subjid parmgrp adt) nodupkey;
  by subjid parmgrp adt;
run;
```

Create a series of macro variables and assign each subject’s visit date in the dataset to those macro variables; create macro variables for each subject’s maximum number of visits; create a macro variable for total number of lab test groups for all subjects.

```
data lbdt2;
   set lbdt end=last;
   by subjid parmgrp adt;
   retain parmgrp2 subjid2;
   if first.parmgrp then seq=.;
   seq+1;

   if _n_=1 then do;
       grpseq=1;
       parmgrp2=parmgrp;
       subjid2=subjid;
   end;
   if parmgrp2^=parmgrp or subjid2^=subjid then do;
      grpseq+1;
      parmgrp2=parmgrp;
```

```
         subjid2=subjid;
      end;
      call symput('dt'||compress(put(grpseq,best.)||put(seq,best.)),
      put(adt,yymmdd10.));

      if last.parmgrp then do;
          call symput('tot'||compress(put(grpseq,best.)), put(seq,best.));
      end;
      if last then do;
         call symput('lastgrp', put(grpseq,best.));
      end;
      drop parmgrp2 subjid2;
   run;
```

Merge the sequence number back to original lab test dataset

```
   data adlb2;
      merge adlb lbdt2;
      by subjid parmgrp adt;
      param='  '||trim(left(param));
   run;

   proc sort data=adlb2;
     by subjid parmgrp grpseq param lbstresu;
   run;
```

Transpose the data and specify the sequence number which represents each visit date as variable names in the output dataset.

```
   proc transpose data=adlb2 out=adlb3;
     by subjid parmgrp grpseq param lbstresu;
     id seq;
     var lbstresc;
   run;
```

Generate lab test listings

```
   %macro lb;
   %do i=1 %to &lastgrp;
   proc report data=adlb3 nowd headskip headline split='!';
     where grpseq=&i;
     column ('__' subjid parmgrp param lbstresu
       %do j=1 %to &&tot&i;
           _&j
        %end;
         );
      define subjid/group noprint;
      define parmgrp/group noprint;
      define param/group order width=36 flow left id  "Clinical Laboratory Group!
   Parameter" spacing=0;
      define lbstresu/group order width=08 flow left id  'Unit' spacing=0;
      %do j=1 %to &&tot&i;
         define _&j/"&&dt&i&j" width=20;
       %end;
      compute before parmgrp ;
        length c0d $100;
        c0d=trim(left(parmgrp));
         line @01 c0d  $100.;
      endcomp;

      compute before _page_;
         line 132*'_';
         line @1 'Patient ID:' @13 subjid $132.;
       endcomp;
    run;
   %end;
   %mend;
   %lb;
```

Example of the output

```
______________________________________________________________________
Patient ID: 001
_______________________________________________________________________
Clinical Laboratory Group
  Parameter                         Unit   2011-08-11 2011-09-12
                                            (Visit 2)  (Visit 3)

_______________________________________________________________________

CHEMISTRY
  Alanine Aminotransferase (SGPT)   U/L       25         32
  Albumin                           g/L       48         46
  Alkaline Phosphatase              U/L       74         73
   .
   .
   .
_______________________________________________________________________
Patient ID: 002
_______________________________________________________________________
Clinical Laboratory Group
  Parameter                         Unit   2011-07-21 2011-08-12 2011-10-12
                                            (Visit 1)  (Visit 2)  (Visit 4)

_______________________________________________________________________

CHEMISTRY
  Alanine Aminotransferase (SGPT)   U/L       25         32         47
  Albumin                           g/L       48         46         50
  Alkaline Phosphatase              U/L       74         73         80
```

By using CALL SYMPUT, the column headers can dynamically change by subjects.

CONCLUSION

An advantage of PROC REPORT is its ability to define the columns and present results in a report after data manipulation when producing complex reports. Combined with the use of CALL SYMPUT, it can create dynamic columns depending on data in those complex reports which PROC TABLUATE cannot handle.

REFERENCES

[1] SAS(R) 9.2 Macro Language: Reference

[2] Sai Ma. Generate Informative Clinical Laboratory Results Listing, PharmaSUG, 2011.

CONTACT INFORMATION

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

Suwen Li
Everest Clinical Research Services Inc.
675 Cochrane Drive
Suite 408, East Tower
Markham, Ontario, Canada L3R 0B8
(905) 752-5253
suwen.li@ecrscorp.com

sas® Certified Advanced Programmer