PhUSE 2007

Paper CS03

Preliminary data process for PROC REPORT issue

Antoine Brisacier, Sanofi-aventis, Chilly-Mazarin, France

ABSTRACT

Have you ever experienced difficulties when producing a SAS® PROC REPORT summary or listing having a gap between results from a same block of ID/BY variables, using FLOW option, as below:

Parameter	Day	Time	Mean
Diastolic Blood Pressure (mmHg)	Day1	14:00	73.5
		15:00	72.1
		16:00	79.4
Heart rate (bpm)	Day1	14:00	63.2
		15:00	60.3
		16:00	68.1
Systolic Blood Pressure (mmHg)	Day1	14:00	115.2
		15:00	120.5
		16:00	118.3

Fig. 1

But would it be nicer to have the results gathered within the same block of ID/BY variables, even if the output from PROC REPORT is correct?

Parameter	Day	Time	Mean
Diastolic Blood Pressure (mmHg)	Day1	14:00	73.5
		15:00	72.1
		16:00	79.4
Heart rate (bpm)	Day1	14:00	63.2
		15:00	60.3
		16:00	68.1
Systolic Blood Pressure (mmHg)	Day1	14:00	115.2
		15:00	120.5
		16:00	118.3

Fig. 2

This paper will present SAS techniques along to the PROC REPORT's use to overcome this issue and to get the above output.

INTRODUCTION

SAS® PROC REPORT is often used to report statistical results or listings in pharmaceutical industry. It has been an alternative procedure to the DATA _NULL_ step, for being flexible and more user-friendly. However, in some particular cases, the use of the PROC REPORT cannot produce easily what we expect but there are techniques to get around our issues.

In the issue presented in this paper, we will focus on how to prepare the data in the input dataset for the PROC REPORT. First, it will be to include split characters in variables using FLOW option in the PROC REPORT variables definition according to column width. Then from those raw data, new variables will be created to display the variables as needed. Those two steps could be done with some SAS macro techniques.

INSERT SPLIT CHARACTER IN A VARIABLE ACCORDING TO COLUMN WIDTH

The FLOW option in the DEFINE statement of the PROC REPORT allows to wrap the value of the character variable in its own column (by default at a blank according to the columns width). The FLOW option can also be controlled with a split character. This is exactly what we are looking for. How to control the wrapping of the character variable and be able to know when the value split to the next line.

Therefore, our first step will be to insert a split character within the value of a character variable, according to the column WIDTH from DEFINE statement of the PROC REPORT.

Through all this paper we will use the following input data

```
data DSIN;
input test $ 1-31 day $ 33-37 time $ 39-43 result 45-49;
cards;
Systolic Blood Pressure (mmHg)  Day1  14:00 115.2
Systolic Blood Pressure (mmHg)  Day1  15:00 120.5
Systolic Blood Pressure (mmHg)  Day1  16:00 118.3
Diastolic Blood Pressure (mmHg) Day1  14:00 73.5
Diastolic Blood Pressure (mmHg) Day1  15:00 72.1
Diastolic Blood Pressure (mmHg) Day1  16:00 79.4
Heart rate (bpm)                Day1  14:00 63.2
Heart rate (bpm)                Day1  15:00 60.3
Heart rate (bpm)                Day1  16:00 68.1
;
run;
```

The insertion of a split character in a character variable could be achieved with the macro %VARSPLIT (*cf appendix 1)* having in parameter the width of the column, and the specific split character (e.g. "#" in our example) that will be defined at a later step in the PROC REPORT.

This %VARSPLIT macro program calculates for each observation of the variable its length. Then, with a do loop statement on every single character of the character string, it defines the position of each blank character (or eventually split character already in the raw value) as a potential breaking point to insert a split character. The split character will finally be added to the last potential breaking point found before to reach the n^{th} character (where *n* is the width of the column from PROC REPORT). The following character will be consider as a new starting point, and the process is done again and again until the last character of the string is reached.

In our example

```
%varsplit(dsin=DSIN, dsout=DSOUT1, var=test, width=14, split=#);
```

The execution of this macro give us the following results for the variable TEST :

```
Systolic Blood#Pressure#(mmHg)
Diastolic#Blood Pressure#(mmHg)
Heart rate#(bpm)
```

If a basic PROC REPORT is done on the output dataset DSOUT1 as below the result will be the same as presented in *Figure 1* (*cf previous page*).

```
proc report data=DSOUT1 headline spacing=2 split="#" nowindows headskip;
   column test day time result;
   define test   / id order flow width=14 "Parameter";
   define day    / id order flow width=5  "Day";
   define time   / id order flow width=5  "Time";
   define result / display                "Mean";
   break after test / skip;
run;
```

But the split characters "#" added in the variable TEST will be useful for the next step and how to fill the gap.

HOW TO FILL THE GAP

The second step is to do a pre-processing on the input data by creating extra variables associated to each variable defined as ID in the PROC REPORT. It means that for each ID variable, an additional variable will be created to be used as DISPLAY variable in the final PROC REPORT.

Each character value of the variable could now be divided into *n* part according to the split character placed in the previous step within each character string.

```
proc sort data=DSOUT1;
   by test day time;
run;
```

After having sorted the input dataset, our dataset will looks like this :

```
TEST                            FIRST.TEST LAST.TEST DAY  TIME   RESULT
------------------------------- ---------- --------- ---- ------ ------
Diastolic#Blood Pressure#(mmHg)      1          0     Day1  14:00 73.5
Diastolic#Blood Pressure#(mmHg)      0          0     Day1  15:00 72.1
Diastolic#Blood Pressure#(mmHg)      0          1     Day1  16:00 79.4
```

And we will use FIRST.variable and LAST.variable informations and also the *n* part of the string separated by the split character in order to obtain the following dataset:

```
TEST                            _TEST            DAY  TIME   RESULT
------------------------------- ---------------- ---- ------ ------
Diastolic#Blood Pressure#(mmHg) Diastolic        Day1  14:00 73.5
Diastolic#Blood Pressure#(mmHg) Blood Pressure   Day1  15:00 72.1
Diastolic#Blood Pressure#(mmHg) (mmHg)           Day1  16:00 79.4
```

It consists to divide the *n* parts of the string within the *k* observations from a same BY variable value. In our example the value of the variable TEST is “Diastolic#Blood Pressure#(mmHg)” has three observations and divided in three parts which means that the first part will remain on first observation, the second part on the second observation and so on.

```
data dsout2 ;
   set dsout1 ;
   by test day time;

   length _test $200;
   retain cnt 0;
   if first.test then do;
      cnt=1;
   end;

   if not last.test then do;
      _test=scan(test,cnt,"#");
      cnt=cnt+1;
   end;

   else if last.test then do;
      if index(test,'#')>0 then do;
         do while (scan(test,cnt,"#") ne "");
            if _test="" then _test=scan(test,cnt,"#");
            else             _test=trim(left(_test))||"#"||scan(test,cnt,"#");
            cnt=cnt+1;
         end;
      end;
      else do;
         if first.test then _test=test;
      end;
   end;
   _test=trim(left(_test));

run;
```

Those data steps, made for variable TEST in the above example should also be done to all variables declared in the ID variable in the PROC REPORT. A macro process could be done easily, and the macro %VREPORT is presented in the *appendix 2* of this paper.

```
data DSOUT2;
   set DSOUT1;
   by test day time;
   count=_N_;
   %vreport(test,cnt1);
   %vreport(day ,cnt2);
   %vreport(time,cnt3);

run;
```

	test	day	time	result	count	_test	cnt1	_day	cnt2	_time	cnt3
1	Diastolic#Blood Pressure#(mmHg)	Day1	14:00	73.5	1	Diastolic	2	Day1	2	14:00	1
2	Diastolic#Blood Pressure#(mmHg)	Day1	15:00	72.1	2	Blood Pressure	3		3	15:00	1
3	Diastolic#Blood Pressure#(mmHg)	Day1	16:00	79.4	3	(mmHg)	4		3	16:00	1
4	Heart rate#(bpm)	Day1	14:00	63.2	4	Heart rate	2	Day1	2	14:00	1
5	Heart rate#(bpm)	Day1	15:00	60.3	5	(bpm)	3		3	15:00	1
6	Heart rate#(bpm)	Day1	16:00	68.1	6		3		3	16:00	1
7	Systolic Blood#Pressure#(mmHg)	Day1	14:00	115.2	7	Systolic Blood	2	Day1	2	14:00	1
8	Systolic Blood#Pressure#(mmHg)	Day1	15:00	120.5	8	Pressure	3		3	15:00	1
9	Systolic Blood#Pressure#(mmHg)	Day1	16:00	118.3	9	(mmHg)	4		3	16:00	1

Fig. 3 : Dataset DSOUT2

Once those variables have been created, the PROC REPORT will look like this, where original variables could be declared as sorting ID variables non printed (when used for example in BREAK AFTER statement), and new variables defined as DISPLAY variables. Note also that the variable COUNT=_N_ created in the dataset DSOUT2 will be used to sort observations as appropriate.

```
proc report data=DSOUT2 headline spacing=2 split="#" nowindows headskip missing;
   column test count _test _day _time result;
   define test   / id order noprint;
   define count  / id order noprint;
   define _test  / display flow width=14 "Parameter";
   define _day   / display flow width=5  "Day";
   define _time  / display flow width=5  "Time";
   define result / display               "Mean";

   break after test / skip;
run;
```

The final output displayed will correspond to the one presented in *Figure 2* (cf *page 1*).

CONCLUSION

Those pre-processing steps done before a PROC REPORT are not straight forward at their first use, but once the macros are available it could help a lot. It also permit to avoid the use of a more complicated data _NULL_ steps which are usually less user-friendly to report results or listing.

CONTACT INFORMATION

Antoine BRISACIER
sanofi-aventis recherche & developpement
1 avenue Pierre Brossolette
91383 CHILLY-MAZARIN cedex
Work Phone: +33 (0)1 69 79 16 74
Fax: +33 (0)1 69 79 71 65
Email: Antoine.Brisacier@sanofi-aventis.com
Web: www.sanofi-aventis.com

APPENDIX 1

SAS code of macro %VARSPLIT use to insert split character in the values of a character variable according to column width define in the PROC REPORT.

```
%macro varsplit(dsin=,dsout=,var=,width=,split=#);
   %LOCAL dsin dsout var width split;

   data &dsout (drop=break posinit posterm line vartemp _vvar j);
      set &dsin;
      length break posinit posterm line 4 vartemp _vvar  $200;

       break=.; posinit=1; posterm=&width; line=1; _vvar='';

       do j=1 to length(&var);

          if substr(&var,j  ,1) in (' ')  then break=j;

          if j=posterm+1 and break ne . then do;
             vartemp=substr(&var,posinit,(break-posinit));
             if line=1 then _vvar=trim(left(vartemp));
             else           _vvar=trim(left(_vvar))||"&split"||trim(left(vartemp));
             posinit=break+1;
             posterm=break+&width;
             break=.;
             line+1;
          end;

          else if j=posterm+1 and break=. then do;
             vartemp=substr(&var,posinit,(&width));
             if line=1 then _vvar=trim(left(vartemp));
             else           _vvar=trim(left(_vvar))||"&split"||trim(left(vartemp));
             posinit=j;
             posterm=j+&width-1;
             break=.;
             line+1;
          end;

         else if substr(&var,j,1) in ("&split") then do;
             vartemp=substr(&var,posinit,(j-posinit));
             if line=1 then _vvar=trim(left(vartemp));
             else           _vvar=trim(left(_vvar))||"&split"||trim(left(vartemp));
             posinit=j+1;
             posterm=j+&width;
             break=.;
             line+1;
          end;

          if j<length(&var) then do;
             if substr(&var,j+1,1) in (' ',"&split") then break=j+1;
          end;

          if j=length(&var) then do;
             vartemp=substr(&var,posinit,(j+1-posinit));
             if line=1 then _vvar=trim(left(vartemp));
             else           _vvar=trim(left(_vvar))||"&split"||trim(left(vartemp));
          end;

       end;

       &var=left(_vvar);
   run;

%mend varsplit;
```

APPENDIX 2

SAS code of macro %VREPORT to create extra variables to be displayed in the final PROC REPORT.

```
%macro vreport(var,cnt);

  options mprint;
      length _&var $200;
      retain &cnt 0;
      if first.&var then do;
         &cnt=1;
      end;

      if not last.&var then do;
         _&var=scan(&var,&cnt,"#");
         &cnt=&cnt+1;
      end;
      else if last.&var then do;
         if index(&var,'#')>0 then do;
            do while (scan(&var,&cnt,"#") ne "");
               if _&var="" then _&var=scan(&var,&cnt,"#");
               else             _&var=trim(left(_&var))||"#"||scan(&var,&cnt,"#");
               &cnt=&cnt+1;
            end;
         end;
         else do;
            if first.&var then _&var=&var;
         end;
      end;
      _&var=trim(left(_&var));


        options nomprint;
%mend;
```