Paper CC07

Using Compute Blocks in Proc Report

Sharon Dunn, Sepracor Inc., Marlborough, MA

ABSTRACT

Proc Report is widely used to generate tables and listings. By incorporating the use of compute blocks within the proc report, we can utilize data null capability to get around some common challenges. For anyone still using a version of SAS 8.1 or earlier, there is a 10 line text limitation for titles and footnotes. Compute blocks will allow for as many lines of text as needed, as well as letting you create page specific titles and footnotes. They can also be a very useful way of specifying varying category titles within the report body, particularly if the data dictates which titles should be used. Another advantage of using compute blocks is to create the line 'Number of Patients Dosed' without having to add an observation to your dataset. You simply calculate macro variables for these N's and use a compute block to output the information into the report.

INTRODUCTION

Compute blocks in proc report can be very powerful tools. They enable you to do many things you might not have been able to do in the version of SAS you are using. They are particularly handy when the data dictates how the titles, footnotes or other information within the body of the report needs to be displayed. The objective of this paper is to demonstrate both the efficiency achieved in reporting and the power of the Compute Block. Through simple data manipulation, you are able to get the data into a format that is conducive to using these techniques.

10 LINE TEXT LIMITATION

If you have not made the upgrade to SAS 8.2 then you are still in a bind when you need to display more than 10 lines of text in either your titles or footnotes of your report. To effectively get around this you can use a compute block. The first step in doing this is to create a page variable on your dataset which will be read into the report. Typically this is done by counting the lines of the dataset and then assigning a page based on those lines or on a subgroup.

```
*** COUNT PAGES*** ;
data all;  set all end=EOF ;
by groupn;
   lines + 1 ;
   if lines > 11 or first.groupn then do ;
      page + 1 ;
      lines = 0 ;
   end ;
   if EOF then call symput('totpages',left(put(page,3.))) ;
run ;
```

Once you have a page variable you are able to implement the compute block logic. The following is an example of more than 10 lines of footnotes:

```
COMPUTE AFTER PAGE;
   length TEXT1 TEXT2 TEXT3 TEXT4 TEXT5 TEXT6
          TEXT7 TEXT8 TEXT9 TEXT10 TEXT11 $ &linesize ;
   TEXT1  = "footnote 1    ";
   TEXT2  = "footnote 2    ";
   TEXT3  = "footnote 3    ";
   TEXT4 =  "footnote 4    ";
   TEXT5 =  "footnote 5    ";
   TEXT6 =  "footnote 6    ";
   TEXT7 =  "footnote 7    ";
   TEXT8 =  "footnote 8    ";
   TEXT9 =  "footnote 9    ";
   TEXT10=  "footnote 10    ";
   TEXT11=  "footnote 11    ";
   TEXT12= "Source code: %trim(&source)";
   TEXT13= "Date generated: &sysdate9.";
   LINE @1 &linesize*'_' ;
   LINE @1 TEXT1 $&linesize.. ;
   LINE @1 TEXT2 $&linesize.. ;
   LINE @1 TEXT3 $&linesize.. ;
   LINE @1 TEXT4 $&linesize.. ;
   LINE @1 TEXT5 $&linesize.. ;
   LINE @1 TEXT6 $&linesize.. ;
   LINE @1 TEXT7 $&linesize.. ;
   LINE @1 TEXT8 $&linesize.. ;
   LINE @1 TEXT9 $&linesize.. ;
   LINE @1 TEXT10 $&linesize.. ;
   LINE @1 TEXT11 $&linesize.. ;
   LINE @1 &linesize*' ' ;
   LINE @1 TEXT12 $60. @82 TEXT13 $25. ;
ENDCOMP;
```

VARYING TITLES AND FOOTNOTES BY PAGE
The above code can also be applied for titles, however it is not as simple a task since if you opt to create your titles in a compute block you must also create your column headers the same way. Again, you would have to create a page variable to define Before Page. What is probably even more useful is the ability to create page specific titles or footnotes. The logic is the same for both. Essentially the code, which illustrates the use of page specific titles, would look something like this:

```
COMPUTE BEFORE PAGE;
   length TEXT1 TEXT2 TEXT3 $ &linesize ;
IF PAGE=1 THEN DO;
   TEXT1 = "TITLE1 FOR PAGE 1";
   TEXT2 = "TITLE2 FOR PAGE 1";
   TEXT3 = "          ";
END;
ELSE DO;
   TEXT1 = "TITLE1 FOR PAGES OTHER THAN 1";
   TEXT2 = "          ";
END;
   LINE @1 TEXT1 $&linesize.. ;
   LINE @1 TEXT2 $&linesize.. ;
   LINE @1 TEXT3 $&linesize.. ;
   LINE @1 &linesize*'_' ;
ENDCOMP;
```

VARYING CATEGORY TITLES
Often there is a situation when it is important to be able to specify different group titles within the report without having to create an observation on the input dataset. I have found this particularly helpful when a variable on a dataset contains 2 distinct types of information. For example, if you had a variable which contained information regarding inclusion and exclusion criteria for a study. We wanted to be able to present the inclusion and exclusion criteria for a study by treatment group. Because there was one field, which contained the information as to whether the criteria were inclusion or exclusion, it was simpler to create a category variable and use a compute block to present the data easily. The first step was to create the category variable:

```
proc sort data=incexc.inclexcl(keep=group iecode answer)
               out=inc(keep= group iecode) nodupkeys;
   by group iecode;
   where substr(iecode,1,1) in ('I');
   format iecode $iecode.;
run;
proc sort data=incexc.inclexcl(keep=group iecode answer)
               out=exc(keep=group iecode) nodupkeys;
   by group iecode;
   where substr(iecode,1,1)='E';
   format iecode $iecode.;
run;
data all; set inc(in=in1)  exc(in=in2);
   if in1 then cat=1;
   else if in2 then cat=2;
run;
```

The next step is to run the proc report. Notice that by computing before category (cat), You can create the text that describes the data, which will be presented beneath it. If the category =1 then you will be presenting the inclusion criteria and if the category =2 then you will be presenting the exclusion criteria.

```
proc report data=all nowd split='*' center headskip missing spacing=2 ;
   columns groupn cat iecode;
   by groupn ;
define groupn    / order   order=internal noprint ;
define cat      / order   order=internal noprint ;
define iecode / order order=internal left width=107 flow
               ‘Admission Eligibility Criteria*__' ;
BREAK AFTER CAT / SKIP;
BREAK AFTER GROUPN / PAGE;
COMPUTE BEFORE CAT;
   if cat=1 then TEXT1 = "Inclusion Criteria";
   else if cat=2 then TEXT1 = "Exclusion Criteria";
   TEXT2 = "------------------";
   LINE @1 TEXT1 $&linesize..;
   LINE @1 TEXT2 $&linesize..;
ENDCOMP;
COMPUTE AFTER GROUPN;
   TEXT3 = "Source code: %trim(&source)";
   TEXT4 = "Date generated: &sysdate9.";
   LINE @1 &linesize*'_' ;
   LINE @1 &linesize*' ' ;
   LINE @1 TEXT3 $70. @82 TEXT4 $25. ;
ENDCOMP;
run ;
```

NUMBER OF PATIENTS DOSED

Almost every table I create has some reference to the population being evaluated. This is usually either the number of subjects randomized or dosed. It can also be just those subjects evaluable for efficacy. Often those numbers are overall and by treatment groups. I have found it very efficient and easy to simply calculate the number of subjects using code, which creates macro variables for each group, and then using a compute block in my proc report to add in the text and the number of subjects. I find this to be much more appealing than adding an observation to my report dataset with the information in it. This first requires calculating the number of subjects.

```
***CALCULATING THE NUMBER OF SUBJECTS****;
proc sort data=perpat; by pksubj; run;
data _null_; set perpat;
   by pksubj;
   if first.pksubj then _pt = 0;
      _pt + 1;
   if last.pksubj then call symput('n_' || left(pksubj), put(_pt,4.));
run;
```

Above I have calculated the number of subjects and stored them in macro variables defined as &n_1 and &n_2. I will use these within a compute block in the proc report to create this line at the beginning of my report, which specifies my denominator. This is a ‘Compute Before’. The code for the report is only a few lines but saves a step in the data processing.

```
COMPUTE BEFORE;
   TEXTA="Number of Pharmacokinetic Subjects";
   TEXTB="   ";
   LINE @2 TEXTA $&linesize.. @60 "&N_1" @92 "&N_2";
   LINE @2 TEXTB $&linesize.. ;
ENDCOMP;
```

The Compute before will put the text ‘Number of Pharmacokinetic Subjects’ on the top line in the body of the report with the number of subjects for each group. Below is an example of the output.

```
    Neopterin (ng/mL) Levels for Pharmacokinetic Subjects


                              Dose group1      Dose Group 2
_____________________________________________________________
Number of PK Subjects             23               25
```

PROGRAM SOURCE AND DATE STAMP

All of our reports need to have a source and date stamp for various reasons. One reason is the audit trail necessary for the FDA. Other reasons include the importance of being able to find the program which created the table. Sometimes this is not as obvious as it should be, so we have taken to creating this stamp at the bottom of all our output. The stamp identifies the program which created the output, as well as the date it was created. This is very important when we get to the point of creating the clinical study report. Because there are times when data needs to be updated or cleaned, often we find ourselves re-running programs to create the tables for our report. It then becomes crucial to know that all the tables have been run on the latest version of the data. If you have a date on your table, which is prior to the date on your data for that table, then you know you have a problem. This can actually be done in a number of ways but I find that compute blocks are a safe and easy way to accomplish this. Below I have included the code to demonstrate the ease with which this is achieved:

```
                    BREAK AFTER GROUPN / PAGE;
                    COMPUTE AFTER GROUPN;
                       TEXT3 = "Source code: %trim(&source)";
                       TEXT4 = "Date generated: &sysdate9.";
                       LINE @1 &linesize*'_' ;
                       LINE @1 &linesize*'¯' ;
                       LINE @1 TEXT3 $70. @82 TEXT4 $25. ;
                    ENDCOMP;
```

Whatever variable you use to page break after, is the variable you will use in the compute after. The example above is very similar to the footnote examples previously presented. This code only requires that you specify the source.

CONCLUSION

These are just a few examples of how compute blocks can make all our table programming lives much easier. They can strip away the need for additional data step processing, as well as get around many of the limitations of proc report. Because it may be necessary for us to conform to the particular process within our work environment, it is useful to know that within every process there is a silver cloud. Hopefully this paper achieved the goal of demonstrating the efficiency and power of using compute blocks in proc report.

CONTACT INFORMATION

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

Sharon Dunn
Sepracor Inc.
84 Waterford Drive
Marlborough, MA 01752
Work Phone: (508)-357-7331
Fax: (508)-357-7586
Email: Sharon.Dunn@Sepracor.com
Web: