Paper TT13

Getting More from the Compute Block to Enhance Table and Listing Output for Clinical Trials Reporting

David W. Carr, ICON Clinical Research, Redwood City, CA

ABSTRACT

The REPORT procedure has become a popular tool of choice by a number of SAS programming groups in the pharmaceutical research industry for producing tables and listings (TLs). One component of PROC REPORT that can be utilized to great advantage in creating and enhancing TL output is the Compute Block. Among its many capabilities, the Compute Block can be used to create titles and footnotes, to create column headers, and to insert text strings into various sections of the report. The purpose of this paper is to present some examples, with relevant discussion points, of applying the Compute Block to enhance or help produce desired program output that follows the requirements outlined in a study's Statistical Analysis Plan (SAP). This presentation is based on SAS version 8.2 or above, is not limited to any particular operating system, and is intended for intermediate SAS programmers who have some familiarity with the REPORT procedure.

KEYWORDS: SAS, PROC REPORT, COMPUTE BLOCK

INTRODUCTION

The Compute Block in PROC REPORT has a multitude of capabilities, many of which can be utilized to help create the desired result in TL output. The programmer can employ Compute Block processing to create titles, footnotes, column headers, and identifying text as well as produce free text within various areas of a specific report. The following sections provide examples of Compute Block code that accomplish just such tasks – tasks the clinical trials SAS programmer might typically encounter from one day to the next.

Note for the example reports provided, the following assumptions apply:

- The line size (LS=) is 85 characters.
- The macro variable &RPTLINE has a pre-created value of underscore ('_') times the line size (i.e. '_' * 85).
- The macro variable &SOURCE has a system generated value of user name + directory path + date and time.

In most cases, each example will consist of sample report output (TL), the Compute Block code to help produce the output, and relevant points of discussion. (NOTE: As mentioned previously, the Compute Block has many capabilities, however an exhaustive discussion of all of these is beyond the scope of this paper. Therefore, we have limited the context to examples that apply to producing typical TL output within the SAS environment.)

A REVIEW OF BASIC COMPUTE BLOCK SYNTAX

Although the purpose of this paper is not to provide instruction on Compute Block syntax, a brief review of basic syntax and a discussion of some inherent related concepts is useful. Compute Block syntax would most commonly take on a form similar to this:

```
compute before|after <variable name>|  |_page_;
   line @n <variable name> + format.|"text"|"macro variable(s)";
endcomp;
```

This syntax includes 3 primary sections, all of which will be on display in the examples that follow:

1. the COMPUTE statement with either BEFORE or AFTER
2. at least one LINE statement
3. the ENDCOMP statement

Note the following concepts related to Compute Block syntax. These are important to keep in mind when examining the examples included hereafter.

- Any variable specified in the COMPUTE statement (e.g. 'COMPUTE BEFORE TRTCODE;') must be designated in the REPORT definition as either a GROUP or ORDER variable.
- Any text produced from within a 'COMPUTE BEFORE;' block would be displayed at the beginning of the report body output (i.e. page 1) *but after* any column headers or any titles explicitly defined in the TITLE statement.
- Any text produced from within a 'COMPUTE AFTER;' block would be displayed at the end of the report body output (i.e. last page) *but before* any footnotes explicitly defined in the FOOTNOTE statement.
- The LINE statement should always indicate the position at which the specified text is to be placed (e.g. 'LINE @12 "SAMPLE TEXT";').
- Any variable (character or numeric) specified in the LINE statement must be accompanied by a format (e.g. 'LINE @1 TRTCODE TRTFMT.;').
- _PAGE_ is an internal PROC REPORT variable that identifies the boundaries for report body output on each page:
    1. Any text produced from within a 'COMPUTE BEFORE _PAGE_;' block would be displayed at the top of the report body output on each page *but after* any titles explicitly defined in the TITLE statement.
    2. Any text produced from within a 'COMPUTE AFTER _PAGE_;' block would be displayed at the bottom of the report body output on each page *but before* any footnotes explicitly defined in the FOOTNOTE statement.

CREATING TITLES AND FOOTNOTES

The Compute Block can be used to apply titles and footnotes to TLs very easily. Suppose we wanted to do so for the following adverse event table:

GENERIC DRUG COMPANY
PROTOCOL: XX-2008

Table 1
Summary of Adverse Events
Safety Population

System Organ Class Preferred Term	ARM I (N=100)	ARM II (N=100)
Number (%) of Subjects Reporting AEs	32 (32.0%)	41 (41.0%)
Cardiac disorders	5 (5.0%)	6 (6.0%)
Palpitations	3 (3.0%)	4 (4.0%)
Tachycardia	2 (2.0%)	2 (2.0%)
Ear and labyrinth disorders	8 (8.0%)	10 (10.0%)
Vertigo	5 (5.0%)	7 (7.0%)
. . . . .		

NOTE: Includes all subjects who received study drug.
Source: programmer1/pub/studies/training/ae.sas Mar 6, 2008 12:42

The code (bolded) to create these titles and footnotes would look something like this:

```
proc report data=final center missing headline headskip nowd split='|' spacing=2;
   column ord subord col1-col3;
   define ord     / order  order=internal  noprint;
   define subord  / order  order=internal  noprint;
   define col1    /        flow  width=56  "System Organ Class|   Preferred Term ";
   . . . . . .

   compute before _page_;
      line @1 "GENERIC DRUG COMPANY";
      line @1 "PROTOCOL XX-2008";
      line @40 "Table 1";
      line @31 "Summary of Adverse Events";
      line @35 "Safety Population";
      line @1 "&rptline";
   endcomp;
   compute after _page_;
      line @1 "&rptline";
      line @1 "NOTE: Includes all subjects who received study drug.";
      line @1 "&source";
   endcomp;
run;
```

DISCUSSION

Titles are generated in the 'COMPUTE BEFORE _PAGE_;' block while footnotes are created from within the 'COMPUTE AFTER _PAGE_' block. Remember that any text produced from within a 'COMPUTE BEFORE _PAGE_;' block would be displayed at the top of the report body output on each page *but after* any titles explicitly defined in the TITLE statement (which in this case doesn't apply). In contrast, recall that any text produced from within a 'COMPUTE AFTER _PAGE_;' block would be displayed at the bottom of the report body output on each page *but before* any footnotes explicitly defined in the FOOTNOTE statement (which also doesn't apply here).

There can be potential advantages to using Compute Block processing to create titles and footnotes. These include the following:

- All report output (i.e. titles, footnotes and body) can be created within the REPORT procedure.
- The programmer can circumvent the 10-title and 10-footnote limits inherent when using the TITLE and FOOTNOTE statements respectively (i.e. more than 10 of each can be produced).

Of course potential advantages should also be weighed against these considerations when using the Compute Block for creation of titles and footnotes:

- Centering titles (or footnotes if required) can be clumsy (since the number of characters in each title must be compared against the line size) and may require additional macro processing to be more user-friendly.
- As a general rule, use of a utility macro to apply titles and footnotes is preferred, particularly if there are 10 or fewer of each.

CREATING COLUMN HEADERS

In addition to producing titles and footnotes, the Compute Block can be used to create the column headers in a report. Assume that the efficacy table on the following page is to be produced in PROC REPORT and that the titles and footnotes are created via traditional TITLE and FOOTNOTE statements respectively.

GENERIC DRUG COMPANY
PROTOCOL: XX-2008

Table 2
Summary of Progression-Free Survival (PFS) vs. Long-Term Survival (LTS)
Efficacy Population

	ARM I		ARM II	
	PFS	LTS	PFS	LTS
MONTHS				
n	188	188	183	183
Mean	22.1	36.8	21.2	37.9
Median	16.1	32.8	15.8	33.4
75% Time	46.9	53.4	44.6	54.7
Std Error of the Mean	1.245	2.505	1.362	2.114

NOTE: All subjects who received at least 4 immunizations are included.
Source: programmer1/pub/studies/training/pfs_lts.sas Mar 6, 2008 12:42

The code necessary to create the column headers in the above report might resemble that provided below:

```
proc report data=final center missing noheader nowd spacing=2;
   . . . . . .

   compute before _page_;
      line @1 "&rptline";
      line @1 " ";
      line @46 "ARM I"  @66 "ARM II";
      line @41 "----------------"  @61 "----------------";
      line @41 "PFS"  @51 "LTS"  @61 "PFS"  @71 "LST";
      line @1 "&rptline";
      line @1 " ";
   endcomp;
   . . . . . .
run;
```

DISCUSSION

Note the following relevant points regarding the preceding SAS code:

- As mentioned previously, any text created from within a 'COMPUTE BEFORE _PAGE_' block displays on each page at the top of the report body output *but after* any titles created explicitly with the TITLE statement.
- In comparison to the previous example, the PROC REPORT options HEADLINE and HEADSKIP have been replaced by the NOHEADER option. NOHEADER should almost always be used when producing column headers within a Compute Block as this option suppresses the REPORT procedure from automatically creating column headers based on the attributes of each variable included in the report.
- As with creating centered titles, using Compute Block processing to produce column headers can be somewhat awkward in that the programmer must know the precise character location for each header element in order to specify them in the LINE statement (e.g. 'LINE @46 "ARM I"').
- Although it is typically advisable to not create column headers from within a Compute Block, this is a good example of where doing so might be useful since the dashed line ('----------') between parent headers (e.g. 'ARM I') and individual column headers can be problematic when trying to produce from the COLUMN statement. [NOTE: The COLUMN statement would work very well if the line is created using the underscore symbol ('_'), however if a sponsor insists on the dashed line for aesthetic reasons, this example demonstrates a potentially better option.]

INSERTION OF OPEN TEXT

Another useful function of the Compute Block is to insert open text into a report. Open text could appear at the beginning or end of a report, at the top or bottom of each report page, or at the beginning or end of a set of related observations. For instance, the text 'MONTHS' that appears at the top of the report body in the previous example could be produced from the SAS code that follows (assuming titles and column headers are produced in the typical fashion).

```
compute before;
   line @1 "MONTHS";
endcomp;
```

As discussed previously, any text produced from within a 'COMPUTE BEFORE' block would be displayed at the beginning of the report body output (i.e. page 1) *but after* any column headers or any titles explicitly defined in the TITLE statement.

Something similarly useful could be done with a 'COMPUTE AFTER' block to apply a p-value at the end of a table (in this example the p-value has been previously defined as a macro variable).

```
compute after;
   line @10 "P-value from Log-Rank Test"  @80 "&pval";
endcomp;
```

Recall also that, in contrast to a 'COMPUTE BEFORE' block, any text produced from within a 'COMPUTE AFTER' block would be displayed at the end of the report body output (i.e. last page) *but before* any footnotes explicitly defined in the FOOTNOTE statement.

INSERTION OF TEXT TO IDENTIFY GROUPS OF OBSERVATIONS

Another case where the Compute Block can be used effectively involves insertion of text prior to a set of related report observations that identifies how those records are related. Consider the following demographics listing:

GENERIC DRUG COMPANY
PROTOCOL: XX-2008

Listing 3
Demographic Characteristics
Intent-to-Treat Population

Subject	Treatment	Age (yrs)	Gender	Race
Site Number 01				
010001	ARM I	82	Male	White
010002	ARM II	66	Female	White
010003	ARM II	69	Female	White
010004	ARM I	70	Male	Hispanic or Latino
010005	ARM II	64	Male	Black
010006	ARM I	68	Female	White
Site Number 02				
020001	ARM II	70	Male	Black
020002	ARM II	47	Female	Hispanic
. . . . .				

NOTE: Patients who failed screening are not included.
Source: programmer1/pub/studies/training/demo.sas Mar 6, 2008 12:42

In this example, site information is displayed prior to the listing of each subject enrolled at the site. The code to accomplish such a task would look similar to this:

```
 proc report data=demo center missing headline headskip nowd split='|' spacing=2;
   column siteno ptid trtgrp age sex race;
   define siteno / group  order=internal  noprint;
   define ptid   /                width=18  "Subject";
   define trtgrp /                width=18  "Treatment";
   define age    /        center  width=8   "Age|(yrs)";
   define sex    /                width=8   "Gender";
   define race   /                width=30  "Race";

   break after siteno / skip;
   compute before siteno;
      line @1 'Site Number ' siteno $10.;
   endcomp;
run;
```

DISCUSSION

This method for inserting text is more efficient and requires less SAS code than the alternative of creating dummy rows in the dataset used to populate the listing. In addition, the following are important points to be made concerning this particular SAS code:

- As previously discussed, the variable SITENO in this instance must be identified in the REPORT definition as either a GROUP or ORDER variable. Note too that no column for site is actually needed (although it is included in the report definition) and thus the NOPRINT option is employed.
- Since the character variable SITENO occurs in the LINE statement, a format (in this case, $10.) must be applied. Also, note that the insertion of site-identifying text occurs only once in the report (before the first observation within that site group) and does not occur at the top of subsequent report pages if the group of site records spans more than one page.

CONCLUSION

The Compute Block is a very useful PROC REPORT tool that can be used to create titles and footnotes or to insert text into specific areas of a report. Programmers wishing to utilize Compute Block processing must be familiar with certain rules and concepts inherent in the REPORT procedure (e.g. specifying variables as either GROUP or ORDER; knowing where text is placed by default from certain BEFORE and AFTER blocks) in order to employ the tool properly. Further information about Compute Block processing, as well as other aspects of PROC REPORT, can be found in the SAS Procedures Guide.

ACKNOWLEDGEMENTS

I would like to extend a special thanks to the statistical programming management team at ICON for their ongoing support of progress, innovation and professional development, and to my wife Laurey, without whose unwavering support none of this would be possible.

CONTACT INFORMATION

The author can be contacted at:

David W. Carr
ICON Clinical Research
555 Twin Dolphin Drive, Suite 400
Redwood City, CA 94065
Work Phone: 215.616.4966
E-mail: david.carr@iconplc.com