Run your reports through that last loop to standardize the presentation attributes

Niraj J. Pandya, Element Technologies Inc., NJ

ABSTRACT

Post Processing of the report could be considered as the last step of generating listings and analysis reports of clinical data reporting in pharmaceutical industry and is the essential aspect of better presentable reports. In general, these macros developed in SAS® software are very flexible for repeated use and can work with varying needs of report presentation. Post processor macros generally take care of different formatting, display, and other presentation attributes associated with the report. There are various methods to generate and invoke the post processor macros. This paper discusses such methods to create, and effectively use the post processor macros in variety of reporting needs.

The approaches suggested here include usage of global macro variables by resolving such variables in the post processor macro and development of post processor macro based on the report presentation needs. The paper also talks about aspects like flexibility, user control, and effectiveness of the post processor macro usage. This paper concludes with further recommendations about development of effective post processor macros.

INTRODUCTION

Simple it may sound to create basic summary reports and listings from SAS datasets containing clinical trials data, but when it comes to submission of these reports to regulatory authorities, every minor detail even from presentation point of view is important and sometimes getting same reporting attributes on each report across an organization becomes a daunting task. For this purpose, standardization of reporting attributes across an organization can be the best solution and a post processing macro taking care of such attributes is a vital part of any such standardization efforts. This paper is targeted for clinical trials data reporting and deals with different formatting and display issues and other various needs of presenting the clinical data in the form of tables and listings. The techniques mentioned here might be also useful for the presentation of other type of data (i.e. Sales and marketing, Manufacturing, Finance, Information Services, Human Resources etc.). This paper is most useful for the beginners who are novice in clinical data reporting using SAS.

POST PROCESSING

Post processing is a technique in which the text output report file from a reporting SAS program or macro program, is used as input to another macro program and, in turn, passes through different stages to get its different attributes like page size, line size, page numbers, titles, footnotes, headline, footline etc. processed in the more acceptable and presentable form to the regulatory authorities. The secondary macro program stated above is called a Post Processing Macro and is mainly designed to provide cosmetic changes to the final output of the report-writing macro program. The user can write one macro which can take care of all the display attributes or can write several separate macros to take care of different things and then call these macros according to the requirement of the report.

Post processor macros generally use many SAS system functions, data steps, procedures and can call other macros. Using global macro variables, passing them in to post processing macro and resolving those ads flexibility and user control. The user can also create blank reports with same formatting attributes as of other successful reports when no subject meets reporting requirements and can add message in the blank report about this using post processor macro. Having such post processing macros in place can save lots of time of several programmers they could have wasted in the efforts of formatting the report, it adds flexibility and repeated use of it saves resources which justifies the time and other resources invested by the company or organization in developing such macros. Use of such macros also standardized the format of reports across the company.

INPUT AND OUTPUT

Any SAS procedure that generates some sort of output text report file (i.e. PROC REPORT, PROC TABULATE, PROC FREQ, PROC MEANS etc.) can be used as input for Post Processing. This can be specified by passing the parameters as INFILE. The output file of the macro can be specified by passing the parameter as OUTFILE which can be used in file statement of the data step that writes the output with PRINT option.

```
%macro postproc (INFILE =, OUTFILE =);
```

In case of input file does not exist, you can put a message using system function FILEEXIST.

```
%if not %sysfunc (FILEEXIST (&INFILE)) %then

Put "NOTE: File Not Found";
```

HEADLINE

Headline is used to separate the main analysis report or listing from the titles section of the report. To add a headline below the last title and above the column headings, you need to get the last title number in the report. When there is no title defined earlier you can suppress the headline by passing the parameter HEADLINE in the macro definition and setting its value equal to 0 which will be checked in the macro later and headline will be suppressed. This way it gives you the control of whether to have a headline in the report or not.

The last title number can be retrieved using the SASHELP directory. Once you get the number of last title, you need to read the input file line by line which includes titles and footnotes specified earlier and find out the length of the longest line in the report and store it in a macro variable. Using this macro variable you can output a line below the last title.

```
proc sql;
     Select max (NUMBER) into: TITLLAST
     from sashelp.vtitle
     where TYPE ='T' and compress (TEXT) ^='';
quit;
```

The above code will look in VTITLE dataset in the SASHELP directory and will find out the maximum value of NUMBER variable where TYPE is for title and TEXT of the title is not missing and will return the value in &TITLLAST macro variable.

When no title is defined, headline can be suppressed using the following condition in the addition to the above code.

```
%if (^&sqlobs) %then %do;
        %let HEADLINE =0;
%end;
```

READ IN TEXT REPORT FILE

Using SAS data steps you can read the text report file and create a dataset which contains few variables which can be used to determine the page size, line size, maximum length of the line etc. These variables will be used to insert a headline or footline at some specific point of the report. While reading the text report file, a new page can be identified by reading form feed character which can be easily read by using hex literal in SAS. A hex literal in SAS is any of the 16 hex characters (0-9 and A-F) in quotes followed by an x. The hexadecimal value for form feed or new page character is '0c'x.

While reading the text report file, in the first pass you can calculate and store different attributes in variables and then in the second run you can output the report file with headline and footline. Let us say page numbers will be stored in PAGENUM. The number of lines will be stored in NLINES. The length of line will be stored in LLINE. The length of the longest line will be stored in LOFLLINE. The content of each line will be stored in LINE. You can define one global macro variable &LINESIZE and specify the line size there so that it can be used by resolving this macro variable in the macro later on.
The following code will read the input text report file and will calculate other variables.

```
data pass1;

      length PAGENUM NLINES LLINE 8 LINE $200;
      retain PAGENUM 0 LLINE 0 NLINES 0;

      infile "&INFILE" end=eof length = &LINESIZE;
      input @1 LINE;
```

```
        if LINE = '0c'x or _N_ = 1 then do;
            PAGENUM+1;
            NLINES = 0;
            If LINE =: '0c'x then LINE = substr (LINE, 2);
        end;

        NLINES+1;
        LLINE = length (LINE);

run;
```

In the above code, each line's content will be read and stored in LINE until end of file is reached. Each line's number in that particular page will be assigned and it will be stored in NLINES and length of string in each line which will be stored in LLINE. When encountering a new page, PAGENUM will increase by 1 and line number for that particular page will start again from 0.

The following code will find the maximum of the line length which was stored in LLINE in the previous code example and it will put its value in a macro variable &LOFLLINE.

```
proc sql ;
    select max(LLINE) into: LOFLLINE from pass1;
quit;
```

INSERT HEADLINE

Now you have the length of the longest line in the report and you have the last title number from previously explained code. By comparing the last title number with the line number of each page you can find the exact position where a headline has to be inserted in each page of the report.

The following code will insert the headline in the report.

```
data pass2;
     set pass1;
     by PAGENUM;

     %if &HEADLINE = 1 %then %do;

       if (NLINES = &TITLLAST) then do;
          output;

          LINE = repeat ('-', &LOFLLINE);
          output;
          return;
       end;
     %end;

run;
```

The above code will check if the HEADLINE parameter is on by checking its value equal to 1. If condition is satisfied, a LINE of character '-' with the length of longest line in the report (&LOFLLINE) will be output after last title in the dataset by PAGENUM which will be output as final report in turn.

FOOTLINE

Footline is required to separate the main data report or listing from the footnotes. To add a footline above the first footnote and below the last line of data report or listing, you need to get the first footnote in the report as a variable for comparison.

When there is no footnote defined earlier, you can suppress the footline by passing the parameter FOOTLINE in the macro definition and setting its value equal to 0 which will be checked in the macro later and footline will be suppressed. This way it gives you the control of whether to have footnote in the report or not.

The first footnote can be retrieved using the SASHELP directory in a form of a character variable. Once you get the string of the first footnote, you need to read the input file line by line which includes titles and footnotes specified earlier and find out the length of the longest line in the report (as explained in earlier section) and store it in a macro variable (&LOFLLINE). Using this macro variable you can output a line above the first footnote.

```
proc sql;
    select trim (TEXT) into :FOOT1
      from sashelp.vtitle
      where TYPE ='F' and NUMBER = 1;
quit;
```

The above code will read from the VTITLE dataset in the SASHELP directory and will store the string of TEXT variable in the macro variable &FOOT1 where TYPE is for footnote and NUMBER of the footnote is 1.

When no footnote is defined, a footline can be suppressed using following condition in the addition to the above code.

```
%if (^&sqlobs) %then %do;
        %let FOOTLINE =0;
%end;
```

INSERT FOOTLINE

Now you have the length of the longest line in the report and you have the text of first footnote from previously explained code. By comparing the value of macro variable &FOOT1 with the value of LINE variable from pass1 dataset, you can find the exact position where footline has to be inserted in each page of the report.

The following code will insert the footline in the report.

```
data pass2;
     set pass1;
     by PAGENUM;

     %if &FOOTLINE = 1 %then %do;

       if LINE = "&FOOT1" then do;

          LINE = repeat ('-', &LOFLLINE);
          output;
          LINE = "&FOOT1";
       end;
    %end;

run;
```

The code above will check if the FOOTLINE parameter is on by checking if its value is equal to 1. If the condition is satisfied, a LINE of character '-' with the length of longest line in the report (&LOFLLINE) will be output before the first footnote in the dataset by PAGENUM which will be output as final report in turn.

PAGE NUMBERS

In clinical data reporting, page numbers are often required to be presented in the form of “PAGE m of n” or “PAGE m/n” either in the title or footnote. This helps a lot as it indicates how many pages the reviewer should expect and if someone is not getting all the pages, he will be able to identify that he has not received the whole report.

Using below mentioned method, you can add “PAGE m of n” kind of page numbers in the first title. For this, you need to use NONUMBER option to suppress the default page numbers. At the same time, the default date and time stamp of report generation has to be suppressed from first title using the NODATE option and alternatively these can be presented in the last footnote of the report using SAS system variables like &SYSDATE, &SYSTIME and &SYSDAY.

As you have already counted the total number of pages and stored it in the PAGENUM variable while reading the input text report file, you can get the maximum number of pages in the report and then you can present it in the form of “PAGE m of n”.

```
proc sql;
     Select max (PAGENUM) into: MAXPAGE
     from pass1;
quit;
```

The code above will calculate the maximum of PAGENUM from dataset pass1 and then store it in the macro variable &MAXPAGE.

Now you have to find the text of the first title and store it in a macro variable so that you can compare it later with the value of LINE variable defined earlier in pass1 dataset.

```
proc sql;
    select trim (TEXT) into: TITL1
      from sashelp.vtitle
      where TYPE ='T' and NUMBER = 1;
quit;
```

The code above will read from the VTITLE dataset in the SASHELP directory and will store the first title’s string from TEXT variable in to macro variable &TITL1.

Now that you have required information available, you can compare two variables containing the value of first title and when that condition is satisfied, you can enter the page number information in the dataset as a text value of variable LINE.

```
data pass2;
     set pass1;
     by PAGENUM;

   if LINE = "&TITL1" then do;

   LINE = right ("PAGE"||compress (PAGENUM) ||"of"||compress (&MAXPAGE) ;
   output;
   LINE = "&TITL1";
   end;

run;
```

The code above will compare the first title’s text from macro variable &TITL1 with the LINE variable and when condition is satisfied it will enter the text ‘PAGE m of n” which is right aligned in the report.

OUTPUT THE REPORT

At last, with the use of _NULL_ dataset and FILE statement with PRINT option, you can print the report with headlines and footlines where required and also change the format of page numbers which will be used in the finished report.

```
data _NULL_;
     set pass2;
     by PAGENUM;
     file &OUTFILE notitles print;

run;
```

ALTERNATIVE METHODS

Apart from the techniques mentioned so far in this paper, you can also draw a headline and footline when using PROC REPORT for generating a report. You can use COMPUTE BEFORE and COMPUTE AFTER statements respectively in PROC REPORT to draw a dashed line before the first raw of report and after the last raw of report. But when variables in PROC REPORT are used as ID variables, then this method may not work properly. Also every time you write a PROC REPORT, you have to compute the length of line manually. This method will also not work with any other procedures that generate report output such as PROC TABULATE, PROC FREQ, PROC MEANS etc. If you are creating RTF outputs, page numbers in the form of 'PAGE m of n" or "PAGE m/n" can be easily added by adding escape sequence using ODS ESCAPECHAR and escape character functions like {pageof}, {thispage} and {lastpage}. But this will not work with any other kind of report destinations.

CONCLUSION

In the reporting of clinical study data, many times the presentation available from the default SAS settings is not acceptable or not enough and you need to modify it. Effective presentation of data is of equal importance as the data being presented in an analysis report or listing. Most of the time it is very time consuming for a novice programmer to make a simple change in the attributes of report and so it is in the interest of the company or organization to have post processing macros in place which will save time and at the same time help in keeping all reports in a standard format across the company or organization.

There are many other aspects of the reporting that can be standardized, such as a standard footnote consisting of the company's name, its copyright and a confidentiality statement, a specific font size and font style in the report, color of the text, background color of the report, type of the output (PDF, HTML, RTF, PRINTER) etc. which can be controlled with such macros, but these are out of the scope of this paper. It is time consuming to write a post processing macro but well worth the effort, as it saves lots of time and other resources with repeated usage.

CONTACT INFORMATION

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

Name: NIRAJ J PANDYA
Phone: 201-936-5826
E-mail: npandya@elementtechnologies.com