PharmaSUG 2015 - Paper TT16

PROC PRINT - the Granddaddy of all Procedures, Enhanced and Still Going Strong!

David Franklin, TheProgrammersCabin.com, Litchfield, NH

ABSTRACT

The PRINT procedure, or PROC PRINT, has been around since SAS® first began and is considered one of the granddaddy procedures. Although this procedure has been replaced in part by the REPORT procedure, there is still a lot you can do with it.

This paper looks at first a simple dump of data, then dresses it up with the use of statements like the BY and ID statements to publication ready output. Next, output is cranked up a notch to demonstrate how PROC PRINT enhancements can be used to produce HTML (with graphics and links), RTF and PDF (with bookmarks). Along the way the paper will also touch on techniques for post processing to make your output more alive.

INTRODUCTION

PROC PRINT can trace its lineage back to the first version of SAS to be commercially released. In recent years, with the advent of PROC REPORT and the ODS system, the PRINT procedure has been pushed to the back of the reporting procedures to where it is considered as no more than being able to do a basic data dump. However, PROC PRINT has been enhanced and refined to where it is still a procedure worth knowing and having an understanding of.

FIRST SOME DATA

For the purposes of this paper, the following data is used to produce the output:

```
data Exposure;
 length subjid $4 exstdt exdose 8 exdoseu $2;
 format exstdt date9.;
 infile cards;
 input subjid $ exstdt yymmdd10. exdose;
 exdoseu='mg';
cards;
0001 2015-01-03 100
0001 2015-01-10 100
0001 2015-01-17 100
0002 2015-01-04 100
0002 2015-01-11 50
0002 2015-01-18 100
0003 2015-02-01 200
0003 2015-02-08 100
0003 2015-02-16 50
;
run;
```

For all the programs, the following code is used:

```
** Sort our data for the PRINT procedure. Note use of DESCENDING option
   to sort in decreasing year value, customer wants latest dose first.;
proc sort data=Exposure;
 by subjid DESCENDING exstdt exdose;
run;

** Set up standard titles and footnotes;
title1 "Subject Exposure Data";
footnote1 "Note: Subject 0002 data is under review due to problem at the site.";
footnote2 "Source: EXPOSURE";
```

A SIMPLE DATA DUMP

First, a simple basic PRINT procedure call that “dumps” the data:

```
proc print data=Exposure;
   var subjid exstdt exdose exdoseu;
   title2 "(Basic Output)";
run;
```

which results in the output given below:

```
                    Subject Exposure Data
                        (Basic Output)

        Obs    subjid       exstdt    exdose    exdoseu

         1      0001     17JAN2015      100       mg
         2      0001     10JAN2015      100       mg
         3      0001     03JAN2015      100       mg
         4      0002     18JAN2015      100       mg
         5      0002     11JAN2015       50       mg
         6      0002     04JAN2015      100       mg
         7      0003     16FEB2015       50       mg
         8      0003     08FEB2015      100       mg
         9      0003     01FEB2015      200       mg

            Note: Subject 0002 data is under review.
                        Source: EXPOSURE
```

There are no formats, except what is already defined in the dataset and no labels for the variables – this is a basic as it gets.

NOOBS AND LABEL OPTIONS

The NOOBS and LABEL options make the output a little more presentable by removing the OBS variable and allowing for column labels that are not just variable names. The following example demonstrates this

```
proc print data=Exposure noobs label;
   var subjid exstdt exdose exdoseu;
   label subjid='Subject'
         exstdt='Date of Treatment'
         exdose='Treatment Dose'
         exdoseu='Treatment Unit';
   title2 "(Something a Little Better)";
run;
```

with the following output given below:

```
                    Subject Exposure Data
                  (Something a Little Better)

                     Date of     Treatment    Treatment
        Subject     Treatment        Dose         Unit

         0001      17JAN2015       100          mg
         0001      10JAN2015       100          mg
         0001      03JAN2015       100          mg
         0002      18JAN2015       100          mg
         0002      11JAN2015        50          mg
         0002      04JAN2015       100          mg
         0003      16FEB2015        50          mg
         0003      08FEB2015       100          mg
         0003      01FEB2015       200          mg

            Note: Subject 0002 data is under review.
                        Source: EXPOSURE
```

Just by adding these two simple options the output is a little better, but we can still do more.

BY AND ID STATEMENTS

The BY and ID statements allow for a “grouping” of the output. The BY statement itself produces a separate section of the report for each BY group, while the ID statement identifies observations by using the formatted values of the

variables that you list instead of by using observation numbers.

```
options byline;
proc print data=Exposure label;
   by subjid;
   id subjid;
   var exstdt exdose exdoseu;
   label subjid='Subject'
         exstdt='Date of Treatment'
         exdose='Treatment Dose'
         exdoseu='Treatment Unit';
   title2 "(Something even better, with BY and ID)";
run;
```

results in the output given below:

```
                    Subject Exposure Data
               (Something even better, with BY and ID)

                     Date of     Treatment    Treatment
            Subject  Treatment      Dose         Unit

             0001    17JAN2015       100          mg
                     10JAN2015       100          mg
                     03JAN2015       100          mg

             0002    18JAN2015       100          mg
                     11JAN2015        50          mg
                     04JAN2015       100          mg

             0003    16FEB2015        50          mg
                     08FEB2015       100          mg
                     01FEB2015       200          mg

               Note: Subject 0002 data is under review.
                           Source: EXPOSURE
```

In the PROC PRINT call, a NOOBS is implied so that option is not required. The variable in the ID statement is used as a "grouping" variable and must also be included in the BY statement -- this is what makes the SUBJID variable appear as a grouping variable and have a spacing (blank line) between each group. Like before, the LABEL statement and option is used to make user friendly labels appear rather than the variable names. It is important to note that the BYLINE option must be on for this to work although the usual BY line processing does not appear in the listing the usual way.

Unlike the REPORT procedure, a line underneath the column headings does not appear in this mode, but does appear when using ODS destinations as we stall see later. Given all the advancements with PROC PRINT it would be nice to have that option outside ODS.

In order for a line to appear underneath the column headings, some post-processing needs to be done in the following order:

- Make a line underneath the column headings – does not need to be all the way across except for the first and last variables (see below of an example). I find it useful to use a character not used elsewhere to set the underline character at this stage.
- When the file is read in after the PROC PRINT call, if the line carries the characters for used for the line, find the first and last positions.
- Fill the text between the first and last characters of the line with the character that is going to be the line character.

Using the previous example the use of this method will be shown. It must be noted that the output must be saved in a file, usually using the PRINTTO procedure.

```
proc printto print='Y:\Reference\Telegraph\PharmaSUG2015\POSTPRC1.txt' new;
run;
proc print data=Exposure label split='!';
   by subjid;
   id subjid;
   var exstdt exdose exdoseu;
   label subjid='Subject!£££££'
         exstdt='Date of!Treatment!£££££'
         exdose='Treatment!Dose!£££££'
```

```
          exdoseu='Treatment!Unit!£££££';
   title2 "(Something even better, with what looks like the HEADLINE "
          "option in PROC REPORT)";
run;
proc printto;
run;
```

Note that the use of the SPLIT option that will force a line break in the column header. I have also used the '£' character for the underline – you will see shortly why.

Next is the post-processing step.

```
data _null_;
   length _txt $255;
   infile 'Y:\Reference\Telegraph\PharmaSUG2015\POSTPRC1.txt'
          sharebuffers length=len;
   file 'Y:\Reference\Telegraph\PharmaSUG2015\POSTPRC1.txt';
   input _txt $varying255. len;
   if index(_txt,'£') then do;
      _startpos= index(_txt,'£');
      do _k=_startpos to len;
         if substr(_txt,_k)='£' then _endpos=_k;
      end;
      substr(_txt,_startpos,_endpos)=repeat('-',_endpos-_startpos);
   end;
   put _txt $varying255. len;
run;
```

Because the INFILE statement uses the SHAREBUFFERS option, the INFILE and FILE statements must use the same file name. Also used is the LEN option in the INFILE statement to tell how long the incoming sting is and use this in the INPUT statement.

Only if a '£' is found in the incoming line, will the next step proceed, where we will first find the first and last position of the '£' characters in the string, and then populate that string filling in the gaps with the replacement character, which in this case is the '-' character.

At the end of the datastep, all the lines, whether replaced or not with new text, is output to the file. Below is the new output file:

```
                          Subject Exposure Data
(Something even better, with what looks like the HEADLINE option in PROC REPORT)

                          Date of     Treatment    Treatment
               Subject    Treatment       Dose         Unit
               ---------------------------------------------

                0001      17JAN2015       100          mg
                          10JAN2015       100          mg
                          03JAN2015       100          mg

                0002      18JAN2015       100          mg
                          11JAN2015        50          mg
                          04JAN2015       100          mg

                0003      16FEB2015        50          mg
                          08FEB2015       100          mg
                          01FEB2015       200          mg

                  Note: Subject 0002 data is under review.
                             Source: EXPOSURE
```

This is actually a very tricky post-processing task to do so care is advised.

SUM STATEMENT AND N OPTION

The SUM statement allows for a summary of numbers to be produced in the listing and is useful when subgroup and total summaries are requested, while the N option prints the number of observations.

```
proc print data=Exposure label n='Subject Count = ' 'Total Count = ';
   by subjid;
   id subjid;
   var exstdt exdose exdoseu;
   label subjid='Subject'
         exstdt='Date of Treatment'
         exdose='Treatment Dose'
         exdoseu='Treatment Unit';
   sum exdose;
   title2 "(Output using the SUM Statement)";
run;
```

The SUM statement tells the procedure what to use to sum the variables while the N statement, as used here, will put out the number of observations found in each BY variable and give the label ‘Subject Count’ and ‘Total Count’ for the total number of observations. Below is the new output:

```
                    Subject Exposure Data
                  (Output using the SUM Statement)

                     Date of     Treatment    Treatment
               Subject    Treatment       Dose        Unit

                0001      17JAN2015       100          mg
                          10JAN2015       100          mg
                          03JAN2015       100          mg
               -------                  ---------
                0001                      300

                     Subject Count = 3


                0002      18JAN2015       100          mg
                          11JAN2015        50          mg
                          04JAN2015       100          mg
               -------                  ---------
                0002                      250

                     Subject Count = 3


                0003      16FEB2015        50          mg
                          08FEB2015       100          mg
                          01FEB2015       200          mg
               -------                  ---------
                0003                      350
                                         =========
                                          900

               Subject Count = 3
                 Total Count = 9


          Note: Subject 0002 data is under review.
                       Source: EXPOSURE
```

The Subject Count and Total Count are not useful on the output but is presented for illustrative purposes. Other options are available in the PROC PRINT procedure that I have rarely seen used, including the UNIFORM DOUBLE and ROUND options, and I encourage you to read these up in the documentation and try them.

HTML, RTF AND PDF OUTPUT USING ODS

ODS has allowed the output from many procedures to be collected and reported in formats that go well beyond the text format that SAS originally had. In this paper I will just have a quick look at the three output formats that SAS provides, e.g. HTML, RTF and PDF. In each case a custom style called CUSTOM defined using a PROC TEMPLATE call is used (the use and description of PROC TEMPLATE is beyond the scope of this paper, but there are a large number of references that can be accessed that discuss the PROC TEMPLATE procedure).

First, lets look at HTML.

```
ods listing close;
ods html file='Y:\Reference\Telegraph\PharmaSUG2015\htmlexample.html' style=custom;
proc print data=Exposure label split='!';
   by subjid;
   id subjid /style=[cellwidth=0.75in just=c];
   var exstdt exdose exdoseu /style=[cellwidth=0.75in just=c];
   label subjid='Subject'
         exstdt='Date!of Treatment'
         exdose='Treatment!Dose'
         exdoseu='Treatment!Unit';
   title2 "(HTML Output, Using Custom Template)";
run;
ods html close;
ods listing;
run;
```

The first point of note is the ODS HTML statement – here the output file destination and style reference is made.

The second major difference is the use of the two VAR statements. Inside ODS using the PRINT procedure, the relevant STYLE= options must follow the variables they apply to, however the next variable or set of variable must follow in later VAR statement calls. In the example above, the SUBJID variable has a cell width of 0.75 inches and the column centered, and the EXSTDT, EXDOSE and EXDOSEU variable also has the column width set to 0.75 inch width. The rest of the code is as seen before except the output (see below) is in Times Roman format as defined in the TEMPLATE procedure definition that defined the style.

Below is the output produced using the ODS HTML output destination:

Subject Exposure Data
(HTML Output, Using Custom Template)

Subject	Date of Treatment	Treatment Dose	Treatment Unit
0001	17JAN2015	100	mg
	10JAN2015	100	mg
	03JAN2015	100	mg
0002	18JAN2015	100	mg
	11JAN2015	50	mg
	04JAN2015	100	mg
0003	16FEB2015	50	mg
	08FEB2015	100	mg
	01FEB2015	200	mg

Note: Subject 0002 data is under review.
Source: EXPOSURE

It must be noted here that the use of the STYLE= options have been around since ODS started. The STYLE locations for the PRINT procedure are:

- BYLABEL -- the label for the BY variable on the line containing the SUM totals
- DATA -- the cells of all columns
- GRANDTOTAL -- the SUM line containing the grand totals for the whole report
- HEADER -- all column headings
- N -- N= table and contents
- OBS -- the data in the OBS column
- OBSHEADER -- the header of the OBS column
- TABLE -- the structural part of the report - that is, the underlying table used to set things like the width of the border and the space between cells
- TOTAL -- the SUM line containing totals for each BY group

Style attributes that are available include:

- BORDERWIDTH=
- CELLHEIGHT=
- CELLPADDING=
- CELLSPACING=
- CELLWIDTH=
- FONT=
- FONT_FACE=

- FONT_SIZE=
- FONT_STYLE=
- FONT_WEIGHT=
- JUST=
- POSTIMAGE=
- PRETEXT=
- RULES=
- VJUST=

There are others – refer to the PROC PRINT documentation.

The RTF code is the same except for the ODS RTF statement as given below with the relevant output:

```
ods rtf file='Y:\Reference\Telegraph\PharmaSUG2015\rtfexample.rtf' style=custom;
```

Subject Exposure Data
(RTF Output, Using Custom Template)

Subject	Date of Treatment	Treatment Dose	Treatment Unit
0001	17JAN2015	100	mg
	10JAN2015	100	mg
	03JAN2015	100	mg
0002	18JAN2015	100	mg
	11JAN2015	50	mg
	04JAN2015	100	mg
0003	16FEB2015	50	mg
	08FEB2015	100	mg
	01FEB2015	200	mg

Note: Subject 0002 data is under review.
Source: EXPOSURE

The PDF code is also the same except for the ODS PDF statement as given below with the relevant output:

```
ods pdf file='Y:\Reference\Telegraph\PharmaSUG2015\pdfexample.pdf' style=custom;
```

Subject Exposure Data
(PDF Output, Using Custom Template)

Subject	Date of Treatment	Treatment Dose	Treatment Unit
0001	17JAN2015	100	mg
	10JAN2015	100	mg
	03JAN2015	100	mg
0002	18JAN2015	100	mg
	11JAN2015	50	mg
	04JAN2015	100	mg
0003	16FEB2015	50	mg
	08FEB2015	100	mg
	01FEB2015	200	mg

In the example above the footnote is not displayed but is in the output.

In the ODS examples there is no blank line between the countries. In SAS version 9.2 and beyond, there is an option called BLANKLINE that specifies the number of observations to be read until a blank line appears.

It is very interesting to note that a line underneath the column headers appears in the ODS output destinations, something that had to be done via post processing when it is put to the output file without ODS destinations.

ADDING LINKS AND GRAPHICS IN HTML

Adding a link and a graphic to an HTML file is done using the PREIMAGE and PREHTML STYLE options in the PROC PRINT statement, as shown below:

```
ods listing close;
ods html file='Y:\Reference\Telegraph\PharmaSUG2015\htmlexamplejpg.html'
         style=custom;
proc print data=Exposure label split='!'
   STYLE=[PREIMAGE='Y:\Reference\Telegraph\PharmaSUG2015\border.jpg'
   PREHTML='<a href="http://en.wikipedia.org/wiki/Clinical_trial">Link to more about
           Clinical Trials</a>'];
   by subjid;
   id subjid /style=[cellwidth=0.75in just=c];
   var exstdt exdose exdoseu /style=[cellwidth=0.75in just=c];
   label subjid='Subject'
         exstdt='Date of Treatment'
         exdose='Treatment Dose'
         exdoseu='Treatment Unit';
   title2 "(HTML Output with Graphic and Link, Using Custom Template)";
run;
ods html close;
ods listing;
run;
```

In the example the link was make to an external URL but the link could quite easily be made to other file. Refer to the output below for an example of the new output:

Subject Exposure Data
(HTML Output with Graphic and Link, Using Custom Template)

Link to more about Clinical Trials

Subject	Date of Treatment	Treatment Dose	Treatment Unit
0001	17JAN2015	100	mg
	10JAN2015	100	mg
	03JAN2015	100	mg
0002	18JAN2015	100	mg
	11JAN2015	50	mg
	04JAN2015	100	mg
0003	16FEB2015	50	mg
	08FEB2015	100	mg
	01FEB2015	200	mg

Note: Subject 0002 data is under review.
Source: EXPOSURE

ADDING A GRAPHIC IN RTF AND PDF

Importing a graphic into RTF uses the same style PREIMAGE option in the PROC PRINT statement, as shown below:

```
ods listing close;
ods rtf file='Y:\Reference\Telegraph\PharmaSUG2015\rtfexamplejpg.rtf' style=custom;
proc print data=Exposure label split='!'
      STYLE=[PREIMAGE='Y:\Reference\Telegraph\PharmaSUG2015\border.jpg'];
   by subjid;
   id subjid /style=[cellwidth=0.75in just=c];
   var exstdt exdose exdoseu /style=[cellwidth=0.75in just=c];
   label subjid='Subject'
         exstdt='Date of Treatment'
         exdose='Treatment Dose'
         exdoseu='Treatment Unit';
   title2 "(RTF Output with Graphic, Using Custom Template)";
run;
ods rtf close;
ods listing;
run;
```

with the following output:

Subject Exposure Data
(RTF Output with Graphic, Using Custom Template)

Subject	Date of Treatment	Treatment Dose	Treatment Unit
0001	17JAN2015	100	mg
	10JAN2015	100	mg
	03JAN2015	100	mg
0002	18JAN2015	100	mg
	11JAN2015	50	mg
	04JAN2015	100	mg
0003	16FEB2015	50	mg
	08FEB2015	100	mg
	01FEB2015	200	mg

The footer will appear in the output but is removed to made the image clear to read.

Just by changing the RTF destination to a PDF destination will also bring in the same graphic into a PDF created output file.

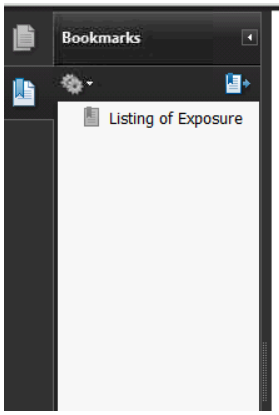
BOOKMARKS AND PDF

Bookmarks are created automatically when creating a PDF file using ODS PDF, however the bookmarks use default value which make the text look awful. There is a trick to modifying the bookmarks as the following example demonstrates:

```
ods listing close;
ods pdf file='Y:\Reference\Telegraph\PharmaSUG2015\pdfexamplelink.pdf' style=custom;
ods escapechar="!";
ods proclabel="Listing of Exposure";
proc print data=Exposure label split='!' contents='';
   by subjid;
   id subjid /style=[cellwidth=0.75in just=c];
   var exstdt exdose exdoseu /style=[cellwidth=0.75in just=c];
   label subjid='Subject'
         exstdt='Date of!Treatment'
         exdose='Treatment!Dose'
         exdoseu='Treatment!Unit';
   title2 "(PDF Output with Bookmarks, Using Custom Template)";
run;
ods pdf close;
ods listing;
run;
```

The ODS PROCLABEL statement replaces the default text associated with the bookmark at the first level – this should be done before the PRINT procedure is run.

The second item is the CONTENTS option in the PROC PRINT statement – in this case the text is set to missing so that the label for the second level is missing. See below for the output:



Subject Exposure Data
(PDF Output with Bookmarks, Using Custom Template)

Subject	Date of Treatment	Treatment Dose	Treatment Unit
0001	17JAN2015	100	mg
	10JAN2015	100	mg
	03JAN2015	100	mg
0002	18JAN2015	100	mg
	11JAN2015	50	mg
	04JAN2015	100	mg
0003	16FEB2015	50	mg
	08FEB2015	100	mg
	01FEB2015	200	mg

When creating separate output for a single PDF file using ODS PDF it is useful to run the ODS PROCLABEL statement before each report is produced so that the bookmarks are clearly set up at the ODS PDF CLOSE statement, i.e. the PDF file is generated.

CONCLUSION

The PRINT procedure is not just something used to “dump” data from a dataset to a text listing file. During the tour we have seen how simple things like NOOBS and LABEL options, and BY and ID statements make the output more presentable. An example of how we can post process a file was also shown. Next on our journey was a brief introduction into how ODS HTML/RTF/PDF has enhanced the PRINT procedure. The PRINT procedure may be one of the oldest procedures in the SAS stable but it still has life today as a serious option when considering what procedure to use to produce your output.

REFERENCES

SAS Institute Inc. 2006. Base SAS® 9.1.3 Procedures Guide, Second Edition, Volumes 1, 2, 3, and 4. Cary, NC: SAS Institute Inc.

CONTACT INFORMATION

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

Name: David Franklin
Cell Phone: 603-275-6809
E-mail: dfranklin@TheProgrammersCabin.com