Paper 079-2008

A Step-by-Step Introduction to PROC REPORT

David Lewandowski, Thomson Healthcare, Evanston, IL

ABSTRACT

Have you read the description of PROC REPORT in the SAS® manual and been left scratching your head wondering where to start? Then, here's a step by step introduction. It walks through the basics, one feature at a time, so you can see each one's impact on the report. When you're finished, you will already know how to create ninety percent of the reports you need. And more importantly, you will have a context so you can go back to the manual and learn the advanced features.

INTRODUCTION

The syntax of PROC REPORT is different from all the other procedures and many of us found the manual less than helpful in learning its unique statements. Most of us turned to a colleague and asked him to explain PROC REPORT. We got a brief introduction and a set of sample code that we've been modifying ever since. This paper will give you the same thing—a simple introduction and lots of sample code to enhance as needed.

STEP 1: CREATE A SMALL DATASET FOR REPORTING

To start off, let's create a dataset to use for reporting—say monthly wine sales by zip code and county. Here's the program and the resulting listing:

```
data mnthly_sales;
  length zip $ 5 cty $ 8 var $ 10;
  input zip $ cty $ var $ sales;
  label zip="Zip Code"
        cty="County"
        var="Variety"
        sales="Monthly Sales";
  datalines;
52423 Scott Merlot 186.
52423 Scott Chardonnay 156.61
52423 Scott Zinfandel 35.5
52423 Scott Merlot 55.3
52388 Scott Merlot 122.89
52388 Scott Chardonnay 78.22
52388 Scott Zinfandel 15.4
52200 Adams Merlot 385.51
52200 Adams Chardonnay 246
52200 Adams Zinfandel 151.1
52200 Adams Chardonnay 76.24
52199 Adams Merlot 233.03
52199 Adams Chardonnay 185.22
52199 Adams Zinfandel 95.84
;

proc print data=mnthly_sales;
  title "Raw Data";
run;
```

Raw Data

Obs	ZIP	CTY	VAR	SALES
1	52423	Scott	Merlot	186.00
2	52423	Scott	Chardonnay	156.61
3	52423	Scott	Zinfandel	35.50
4	52423	Scott	Merlot	55.30
5	52388	Scott	Merlot	122.89
6	52388	Scott	Chardonnay	78.22
7	52388	Scott	Zinfandel	15.40
8	52200	Adams	Merlot	385.51
9	52200	Adams	Chardonnay	246.00
10	52200	Adams	Zinfandel	151.10
11	52200	Adams	Chardonnay	76.24
12	52199	Adams	Merlot	233.03
13	52199	Adams	Chardonnay	185.22
14	52199	Adams	Zinfandel	95.84

SYNTAX

Next, let me describe PROC REPORT's syntax. The COLUMN statement is used to list each report column. Each column, in turn, has a DEFINE statement that describes how that column is created and formatted. You use the TITLE statement to specify the title at the top of each page.

```
PROC REPORT DATA=datasetname <options>;
  TITLE title text;
  COLUMN variable list and column specifications;
  DEFINE column / define type and column attributes;
  DEFINE column / define type and column attributes;
  ...
RUN;
```

STEP 2: A SIMPLE REPORT

Putting the data and a basic PROC REPORT together…

```
proc report data=mnthly_sales nofs;
  title1 "Simple Report";
  column cty zip var sales;
  define cty / display;
  define zip / display;
  define var / display;
  define sales / display;
run;
```

```
                Simple Report

             Zip                      Monthly
County       Code    Variety           Sales
Scott        52423   Merlot              186
Scott        52423   Chardonnay       156.61
Scott        52423   Zinfandel          35.5
Scott        52423   Merlot             55.3
Scott        52388   Merlot           122.89
Scott        52388   Chardonnay        78.22
Scott        52388   Zinfandel          15.4
Adams        52200   Merlot           385.51
Adams        52200   Chardonnay          246
Adams        52200   Zinfandel         151.1
Adams        52200   Chardonnay        76.24
Adams        52199   Merlot           233.03
Adams        52199   Chardonnay       185.22
Adams        52199   Zinfandel         95.84
```

If you compare "Simple Report" to "Raw Data" created in Step 1, you will notice a few differences.

- Simple Report doesn't have an OBS column
- the variables are listed in their order in the column statement
- the column headers are the labels not the variable names
- the column headers are adjusted to the column width, not the other way around

By the way, nofs is used to turn off the procedure's interactive features.

STEP 3: SOME FORMATTING OPTIONS

On the define statement, you can specify the formatting options for that column. "format" applies the standard SAS formats to the column, "width" sets the column width, "flow" wraps the text within the width you specified, and "noprint" suppresses printing that column. You can replace the label as the column heading by specifying the new heading in quotes. The slash (i.e. "/") is the line break symbol used to force the heading to wrap lines. The PROC REPORT option "headline" adds a line after the column headings and the "headskip" option adds the blank line. Let's add formatting to Simple Report.

```
proc report data=mnthly_sales nofs
            headline headskip;
  title1 "Simple Formatted Report";
  column cty zip var sales;
  define cty / display width=6 'County/Name';
  define zip / display;
  define var / display;
  define sales / display format=6.2 width=10;
run;
```

```
          Simple Formatted Report

County  Zip                        Monthly
Name    Code    Variety             Sales
-------------------------------------------

Scott   52423   Merlot            186.00
Scott   52423   Chardonnay        156.61
Scott   52423   Zinfandel          35.50
Scott   52423   Merlot             55.30
Scott   52388   Merlot            122.89
Scott   52388   Chardonnay         78.22
Scott   52388   Zinfandel          15.40
Adams   52200   Merlot            385.51
Adams   52200   Chardonnay        246.00
Adams   52200   Zinfandel         151.10
Adams   52200   Chardonnay         76.24
Adams   52199   Merlot            233.03
Adams   52199   Chardonnay        185.22
Adams   52199   Zinfandel          95.84
```

STEP 4: THE ORDER DEFINE TYPE

In a DEFINE statement, the word after the slash specifies the define type for that column. The valid define types are DISPLAY, ORDER, GROUP, ANALYSIS, ACROSS and COMPUTED. Up to now, we've been using the DISPLAY define type. Now, let's use each of the other five types in turn. The ORDER define type specifies the column used to sort the report.

```
proc report data=mnthly_sales nofs
            headline headskip;
  title1 "Ordered Report (Order Type)";
  column cty zip var sales;
  define cty / order width=6 'County/Name';
  define zip / display;
  define var / display;
  define sales / display format=6.2 width=10;
run;
```

```
      Ordered Report (Order Type)

County  Zip                     Monthly
Name    Code   Variety            Sales
-------------------------------------

Adams   52200  Merlot            385.51
        52200  Chardonnay        246.00
        52200  Zinfandel         151.10
        52200  Chardonnay         76.24
        52199  Merlot            233.03
        52199  Chardonnay        185.22
        52199  Zinfandel          95.84
Scott   52423  Merlot            186.00
        52423  Chardonnay        156.61
        52423  Zinfandel          35.50
        52423  Merlot             55.30
        52388  Merlot            122.89
        52388  Chardonnay         78.22
        52388  Zinfandel          15.40
```

Notice how cty, the ordered column, doesn't repeat in each row, only when it changes.

STEP 5: THE GROUP DEFINE TYPE

The GROUP define type consolidates all the observations with the same unique combination of grouped variables. You can specify the order of the rows within the group by using the ORDER= option of the DEFINE statement. In this case they are ordered by descending frequency of var.

```
proc report data=mnthly_sales nofs
            headline headskip;
  title1 "Grouped Report (Group Type)";
  column cty zip var sales;
  define cty / group width=6 'County/Name';
  define zip / group;
  define var / group order=freq descending;
  define sales / display format=6.2 width=10;
run;
```

```
      Grouped Report (Group Type)

County  Zip                     Monthly
Name    Code   Variety            Sales
-------------------------------------

Adams   52199  Merlot            233.03
               Chardonnay        185.22
               Zinfandel          95.84
        52200  Merlot            385.51
               Chardonnay        246.00
                                  76.24
               Zinfandel         151.10
Scott   52388  Merlot            122.89
               Chardonnay         78.22
               Zinfandel          15.40
        52423  Merlot            186.00
                                  55.30
               Chardonnay        156.61
               Zinfandel          35.50
```

You probably noticed that the GROUP define type isn't very helpful—unless it is used with the ANALYSIS define type.

STEP 6: THE ANALYSIS DEFINE TYPE

The ANALYSIS define type lets you specify for that column any of the statistics used in PROC MEANS, SUMMARY and UNIVARIATE. The statistics are calculated for the group you defined.

```
proc report data=mnthly_sales nofs
            headline headskip;
  title1 "Summed Groups Rept (Analysis Type)";
  column cty zip sales;
  define cty / group width=6 'County/Name';
  define zip / group;
  define sales / analysis sum
                 format=6.2 width=10;
run;
```

```
 Summed Groups Rept (Analysis Type)

     County  Zip       Monthly
     Name    Code        Sales
     -------------------------

     Adams   52199      514.09
             52200      858.85
     Scott   52388      216.51
             52423      433.41
```

STEP 7: MULTIPLE STATISTICS ON THE SAME COLUMN

If you want to calculate more than one statistic on the same column, you create an alias in the COLUMN statement.

```
proc report data=mnthly_sales nofs
            headline headskip;
  title1 "Report with Multiple Statistics";
  column cty zip sales sales=mean_sales;
  define cty / group width=6 'County/Name';
  define zip / group;
  define sales / analysis sum
                 format=6.2 width=10 'Sum';
  define mean_sales / analysis mean
       format=6.2 width=10 'Mean of/Sales';
run;
```

```
     Report with Multiple Statistics

County  Zip                    Mean of
Name    Code          Sum       Sales
-------------------------------------

Adams   52199      514.09      171.36
        52200      858.85      214.71
Scott   52388      216.51       72.17
        52423      433.41      108.35
```

STEP 8: THE ACROSS DEFINE TYPE

If you want a report where all the unique values of a variable have their own column, you use the ACROSS define type. It's the easiest way to create a cross-tab. In this case, we're going to make the Merlot, Chardonnay and Zinfandel values of the var variable their own columns. Besides setting the define type of var to ACROSS, we also specify sales as the variable nested under var. This is done in the COLUMN statement by separating the ACROSS variable from the nested variable with a comma. In this case var,sales. If you use dashes as the first and last characters in the ACROSS column header, they span all the columns (it works for : = _ .* + too).

```
proc report data=mnthly_sales nofs headline headskip;
  title1 "Cross Tab Report (Across Type)";
  column cty zip var,sales;
  define cty / group width=6 'County/Name';
  define zip / group;
  define var / across order=freq descending '- Grape Variety -';
  define sales / analysis sum format=6.2 width=10 'Revenue';
run;
```

```
                        Cross Tab Report (Across Type)

                                 --------- Grape Variety ----------
                County  Zip        Merlot  Chardonnay   Zinfandel
                Name    Code      Revenue     Revenue     Revenue
                -------------------------------------------------

                Adams   52199      233.03      185.22       95.84
                        52200      385.51      322.24      151.10
                Scott   52388      122.89       78.22       15.40
                        52423      241.30      156.61       35.50
```

STEP 9: BREAK AND RBREAK STATEMENTS

The BREAK statement adds summaries (subtotals in this case) every time the group column(s) change. You can specify whether the break occurs before or after the group. The RBREAK statement gives you grand totals.

```
proc report data=mnthly_sales nofs headline headskip;
  title1 "Report with Breaks";
  column cty zip var,sales;
  define cty / group width=6 'County/Name';
  define zip / group;
  define var / across order=freq descending '- Grape Variety -';
  define sales / analysis sum format=6.2 width=10 'Revenue';
  break after cty / ol skip summarize suppress;
  rbreak after / dol skip summarize;
run;
```

```
                              Report with Breaks

                          --------- Grape Variety ----------
                 County  Zip        Merlot  Chardonnay   Zinfandel
                 Name    Code      Revenue     Revenue     Revenue
                 ------------------------------------------------

                 Adams   52199      233.03      185.22       95.84
                         52200      385.51      322.24      151.10
                          ----------  ----------  ----------
                             618.54      507.46      246.94

                 Scott   52388      122.89       78.22       15.40
                         52423      241.30      156.61       35.50
                          ----------  ----------  ----------
                             364.19      234.83       50.90

                          ==========  ==========  ==========
                             982.73      742.29      297.84
```

There are many options for the BREAK and RBREAK statements:

OL	overline	DOL	double overline
UL	underline	DUL	double underline
summarize	summarize each group	skip	skip a line after the break
suppress	don't repeat the break variable on the summary line		

STEP 10: THE COMPUTED DEFINE TYPE AND COMPUTE BLOCK

You can also compute your own values from the other data in your report. These computed columns have a COMPUTED define type. Besides the DEFINE statement for each computed column, you need to write a COMPUTE block which starts with a COMPUTE statement and ends with ENDCOMPUTE. Let's add a row total variable, called row_sum, to the report. row_sum is the sum of all the analytic variables in the row. You can use the automatically defined _C#_ variables to make it easier. For example, _C3_ is the value in third column. Here's how it works.

```
proc report data=mnthly_sales nofs headline headskip;
  title1 "Report with Row Sums (Computed Type)";
  column cty zip var,sales row_sum;
  define cty / group width=6 'County/Name';
  define zip / group;
  define var / across order=freq descending '- Grape Variety -';
  define sales / analysis sum format=6.2 width=10 'Revenue';
  define row_sum / computed format=comma10.2 'Total';
  break after cty / ol skip summarize suppress;
  rbreak after / dol skip summarize;
  compute row_sum;
    row_sum = sum(_C3_,_C4_,_C5_,_C6_,_C7_,_C8_);
  endcompute;
run;
```

Report with Row Sums (Computed Type)

County Name	Zip Code	Grape Variety: Merlot Revenue	Chardonnay Revenue	Zinfandel Revenue	Total
Adams	52199	233.03	185.22	95.84	514.09
	52200	385.51	322.24	151.10	858.85
		618.54	507.46	246.94	1,372.94
Scott	52388	122.89	78.22	15.40	216.51
	52423	241.30	156.61	35.50	433.41
		364.19	234.83	50.90	649.92
		982.73	742.29	297.84	2,022.86

If you don't know how many columns your ACROSS define type will create, just use the maximum possible _C#_ variables—the extras don't hurt. A COMPUTE block can contain, more or less, everything allowed in a DATA step including macro variables and %include. And remember, you can also reference any report item from within the block.

CONCLUSION

See how far you've come—just one step at a time. Now you can build most of the reports you need using just these few elements.

PROC REPORT does have more advanced features--use the manual to fill in those finer points. I suggest looking at the COLUMN statement. It has several capabilities that I haven't mentioned. The COMPUTE block is very powerful. But use it with caution. Often your programs will be easier to understand and maintain if you make your calculations in a DATA step before PROC REPORT rather than getting fancy in a COMPUTE block.

RECOMMENDED READING

Carpenter, Arthur. "PROC REPORT Basics: Getting Started with the Primary Statements" http://www.caloxy.com/papers/65_HOW07.pdf.

Carpenter, Arthur. 2007. *Carpenter's Complete Guide to the SAS® REPORT Procedure*. Cary, NC: SAS Institute Inc.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. I can be reached at:

David Lewandowski
Thomson Healthcare
1007 Church Street
Suite 700
Evanston, IL 60201
dave.lewandowski@thomson.com
www.thomsonhealthcare.com