

Where Proc FREQ ends and Proc REPORT begins

Thushan Wijesinghe, LexisNexis, Miamisburg, OH

ABSTRACT

The paper describes the applications of Proc REPORT in real business situations. In many cases we need to cleanse the data and then run it through a statistical procedure (PROC RANK is used in this case for segmentation purposes) before any reports can be prepared.

The paper will detail how to create a dashboard type report using Proc REPORT with SAS® ODS. The paper will start with the basics of the REPORT PROCEDURE such as how to create a report header and how to change the report's appearance (font size, color, etc).

The final section of the paper will exhibit how to use some of advanced functions of the REPORT PROCEDURE such as the COMPUTE option, and using ANALYSIS, SUM and ACROSS variables with the DEFINE statement. In addition the paper will illustrate the use of the COMPUTE statement for traffic lighting (to color individuals cells within the report based on predefined criteria).

INTRODUCTION

The purpose of the paper is to effectively demonstrate the various capabilities of Report PROCEDURE. When used in conjunction with statistical methodology and efficient data cleansing techniques, the Report PROCEDURE can be a powerful tool to aid business/marketing decision making.

The paper considers a Bank which issues credit cards to customers. The Bank is interested in tracking the customers' use of card (in terms of the # of transactions) over time. To accomplish this, we first build a data set of users from the last year (2004) and then segment them into quartiles. The paper walks through the steps to build a dashboard which will run monthly. The dashboard will track the change in customer behavior (moving to higher or lower segments) and perform calculations on average use.

DATA PREPARATION

If the activity (# of transactions carried out) for 2004 is defined by activity2004, then we could rank the population using a PROC RANK statement:

```
proc rank data=mwsug.paper
  groups = 4 out=mwsug.paper;
  var activity2004;
  ranks rank2004;
run;
```

Use a groups=4 statement to segment the data into quartiles (segments=10 will be deciles).

To view details on the distribution of transactions use a PROC UNIVARIATE statement¹:

```
proc univariate data=mwsug.paper normal plot;
  var activity2004;
  title "Univariate procedure for user activity 2004";
  histogram activity2004 / normal(color=black w=3
                                percents=20 40 60 80 midpercents)
                        cfill = green
                        cframe = ligr
                        name='MyPlot';
  inset n normal(ksdpval) /
        pos = ne format = 6.3;
run;
```

The UNIVARIATE statement will also display a graph of the data distribution.

Using the above PROC RANK we can see how each observation in the population falls into either one of the four quartiles or is tagged as a non-user. We use the data step to identify each user with the correct segment for 2004

based on activity level.

```
data mwsug.paper;
  set mwsug.paper;
  format segmt2004 $22.;
  if rank2004= . then segmt2004 = 'Non user';
  else if rank2004= 0 then segmt2004= '1-25 pctl';
  else if rank2004= 1 then segmt2004= '26-50 pctl';
  else if rank2004= 2 then segmt2004= '51-75 pctl';
  else if rank2004= 3 then segmt2004= '75+ pctl';
run;
```

The same could be done for the 2005 user activity. The segment variable for 2005 will be called *segmt2005*. We need to keep the quartile cut-offs at the same value to see the movement of users within segments (from 2004 to 2005).

WHERE PROC FREQ ENDS

We could create a basic matrix exhibiting the movement of users within the segments using the PROC FREQ as:

```
proc freq data=mwsug.paper;
  title "Movements within segments";
  tables segmt2004*segmt2005 /out=freq_temp nocol norow nocum nopercnt;
run;
```

To show additional details such as calculation of means, percentages and color coding for positive/negative movements, we need a procedure such as Proc REPORT.

USING PROC REPORT

REPORT PROCEDURE can be used to create a “dashboard” type of report to display customer data. The code can be written inside a SAS ODS statement so that the final output will be rendered as a PDF or a RTF file.

SAMPLE OF DATA SET USED

Given below is a small sample of the data used for this paper. This will help the reader to better understand how the REPORT procedure is used to manipulate and summarize data.

active_ user	activity2004	activity2005	channel	rank2004	rank2005
0	18	4	Dir Mail	3	2
0	6	6	Dir Mail	2	2
0	18	8	Telephonic	3	2
0	.	.	Email/Web	4	4
0	.	.	Email/Web	5	5
0	2	1	Telephonic	1	4
0	16	9	Dir Mail	3	2
0	.	.	Email/Web	4	4

segmt2004	segmt2005	yrs_in_2005	post_ move	negt_ move	new_non_ post
75+ pctl	51-75 pctl	Yr06	0	1	0
51-75 pctl	51-75 pctl	Yr06	0	0	0
75+ pctl	51-75 pctl	Yr7+	0	1	0
New user	Inactive	Yr7+	0	0	0
Non user	Non user	Yr06	0	0	0
26-50 pctl	Inactive	Yr06	0	0	0
75+ pctl	51-75 pctl	Yr06	0	1	0
New user	Inactive	Yr06	0	0	0

Variables “post_move”, “negt_move” and “new_non_post” are binary variables. e.g. “post_move” will have a value of 1 if a user moved up at least one segment from 2004 to 2005 (say from 26-50 to 51-75 segment). Similarly “new_non_post” will have a value of 1 if the user was a non user or new user from 2004 and has now moved to a user segment. Rank2004 and Rank2005 give numerical ranks to the segments of 2004 and 2005 respectively.

USING ODS AND CREATING A REPORT HEADER

The shell code for SAS ODS is:

```
ods listing close;
title; footnote;
options orientation=portrait symbolgen date number ls=120 pageno=1;
```

```
ods rtf file="&path\Dashboard for &yy. &mm..rtf"
      startpage=no;
ods escapechar='~';
ods noptitle;
```

```
PROC REPORT statements
....
....
```

```
quit;
ods pdf close;
ods listing;
```

Where *&path*, *&yy* and *&mm* are macro variables referring to the destination of the RTF report, current year and current month respectively.

We can use the following macro declarations at the beginning of the code:

```
%let mm = 08;
%let yy = 2005;
```

Then use the following code to store the year and month in the form we need:

```
%let y=%sysfunc(substr(&yy,3,2));
%let m=%sysfunc(mdy(&mm,01,&y),monname9.);
```

A header for the dashboard can be created using PROC REPORT as:

```
/*Report title + Confidential header*/
proc report data      = makeline nowd
      colwidth      = 25 center
      style(report)={font_face=Tahoma}
      style(header)={background=blue
      foreground=white
      bordercolor=white
      font_face=Tahoma font_size=11pt}

      style(column)={background=red
      foreground=white
      font_weight=bold
      bordercolor=white
      font_face=Tahoma font_size=9pt};

      column ("ABC Bank Ltd" line );
      define line / display "Dashboard for Credit Card [CC] customer
transactions -&m. &yy.";
      define line / display "Produced by Customer Analytics Dept.";

run;
```

As the above code displays, the REPORT procedure allows the user to change many attributes of the dashboard header such as color, font size, font weight, etc.

```
data makeline;
      length line $100;
      line =
      "ABC Bank Ltd. - Confidential Information";
run;
```

The *makeline* statement above will create the last line of the header – the confidentiality statement.

CREATING THE FIRST SUMMARY REPORT

In the first summary report of the dashboard, we are interested in displaying the number of users by the marketing channel used (channel) and the number of years the user has been with the bank (Yrs_in_2005).

```
/*Marketing channel for customer population and additional metrics - years with Bank*/
proc report data = mwsug.paper nowd
      headline headskip split = '*' colwidth = 10
      style(report)={font_face=Tahoma}
      style(header)={background=gray
```

```

        foreground=white
        font_weight=bold
        font_face=Tahoma font_size=8pt}
style(column)={background=white
        foreground=black
        font_face=Tahoma font_size=8pt}
style(summary)={font_weight=bold} ;

```

DEFINING COLUMNS AND THE POWER OF “GROUP” & “ACROSS” VARIABLES

In the next section of code, we identify the columns that will be displayed (see code below). The heading for the table is given quotes. The “*” is used to identify the columns that will be shown under the header. To include all the variables in the data set in the same order just omit the “column” statement.

Immediately following would be the individual column definitions. In the Report PROCEDURE a column can take one of several types, i.e. DISPLAY, ORDER, ACROSS, GROUP and ANALYSIS. A GROUP variable, groups all observations in a single value of the variable to one row. In this example, it will summarize the data for each unique value of the “channel” variable.

ACROSS variables is a powerful feature in the Report PROCEDURE. This option will create a column for each unique value of the ACROSS variable (“Yrs_in_2005” in this example), thus relieving the need to create additional variables. The actual output (the dashboard) is shown in the last page. Please refer to table 1 in the output for the layout of columns and rows.

To display any variables that are not part of the original data set, we need to use the COMPUTE or ANALYSIS option. In this example we are interested in viewing the total for each channel (the sum of all the groups in “Yrs_in_2005” variable). So we define “Total” as a computed variable. We can also specify the display title for each variable after the ‘/’ and variable type (see code below).

```

column channel ("Channel for user population by Yrs_in_2005*" Yrs_in_2005
total);
define channel / group 'Channel' ;
define Yrs_in_2005 / across 'Yrs_in_2005' ;
define Total / computed 'Total' format=comma8.0;

rbreak after / dol summarize ;
compute after;
    channel = 'Total';
endcomp;

compute Total;
    Total = sum (_c2_, _c3_, _c4_, _c5_, _c6_, _c7_, _c8_, _c9_, _c10_);
endcomp;

```

The RBREAK statement computes the summary while the AFTER option places the break lines at the end of the report. So the block of code with the RBREAK statement will compute totals for all the channels by each value of “Yrs_in_2005” variable.

The final block of code with the COMPUTE statement will add columns 2 to 10. The columns are referred to by the location, e.g. _c2_ is column 2 in the report. There are 4 ways to refer to report items in a COMPUTE block. See the SAS documentation on Report PROCEDURE² for additional details.

TRAFFIC LIGHTING A ROW (OR A COLUMN)

We could use a combination of the COMPUTE statement and the CALL DEFINE statement to change properties of a particular column or row. In the code below we use these two statements to change the background color of the row where channel = ‘Email/Web’ to “khaki” color³ and change the font style to *italics*. The CALL DEFINE statement is only valid inside a COMPUTE statement. Similar type of coding can be used to change the attributes of a column.

```

compute channel;
    if channel = 'Email/Web' then do;
        call define (_row_, "style",
                    "style = [background = khaki font_style = italic]");
    end;
endcomp;

run;
/*End of the 1st block of PROC REPORT*/

```

PREPARING A SEGMENT MIGRATION GRID

In the 2nd section of the report, we are interested in summarizing the data and showing the movement of credit card users between transaction segments of 2004 and 2005 (using the variables segmt2004 and segmt2005). Using the resultant table we should be able to track how users belonging to a certain segment in 2004 are now distributed within the 2005 segments. Please refer to table 2 in the output shown in the last page of this paper.

We will use similar code as shown above to prepare the headers for the report.

```
/*Transaction segment migration grid - 2nd block of PROC REPORT*/  
proc report data = mwsug.paper nowd  
  headline headskip split = '*' colwidth = 10  
  style(report)={font_face=Tahoma}  
  style(header)={background=gray  
    foreground=white  
    font_weight=bold  
    font_face=Tahoma font_size=8pt}  
  style(column)={background=white  
    foreground=black  
    font_face=Tahoma font_size=8pt}  
  style(summary)={font_weight=bold  
    font_face=Tahoma} ;
```

For this table we need to show only two variables segmt2004 and segmt2005. The first variable is specified as a GROUP type and the second as ACROSS type. We will use "Total", a COMPUTE variable to sum the table. The use of the RBREAK and the COMPUTE statements are same as shown above. In this table, segmt2005 has only 6 values and they will be listed from the 2nd column to the 7th column. Hence the COMPUTE total statement will sum columns 2 to 7 using the `_cn_` notation (where "n" is the column number).

```
column segmt2004 segmt2005 Total ;  
define segmt2004 / group "Segment 2004 for CC" ;  
define segmt2005 / across "Segment 2005 for CC" ;  
define Total / computed 'Total Population' format=comma8.0 ;  
  
rbreak after / dol summarize ;  
compute after ;  
  segmt2004 = 'Total' ;  
endcomp ;  
  
compute Total ;  
  Total = sum (_c2_, _c3_, _c4_, _c5_, _c6_, _c7_) ;  
endcomp ;
```

ADVANCED TRAFFIC LIGHTING WITHIN THE REPORT PROCEDURE

Assume that for extra clarity, we wanted to color the background of each cell showing a positive movement of an existing user with "light green". Similarly we want to identify negative movements of existing users with "light yellow" and positive movements of non and new users with "pale green". In addition we want to color cells with inactive users with "gray" and leave cells with unchanged users segments white (the default background).

In order to do this we need to write code which will change the properties at the individual cell level and not at the "row" or "column" level. This could be done by opening up a COMPUTE block for the segmt2005 variable. Then we can specify the background color for each column when the row has a fixed value. In the first block of code, we are specifying the colors when segmt 2004 = '1-25 pctl' or the first row of the table.

The code specifies to change the background color of column 3 (`_c3_`) to color #AAFFAA when the row value of segmt2004 is '1-25 pctl'. This cell (2nd row, 3rd column in the matrix) will show the number of user moving from '1-25 pctl' in 2004 to '26-50 pctl' in 2005. Similarly the `_c6_` value of the same row is set to #CCCCC.

Additional conditions can be combined with the IF statement within the COMPUTE block as shown in the second block of code below. As a result `_c7_` of the first row will be changed to background color #FFFF99 only if it meets the additional criteria of `&s0. > 0` (&s0 is a macro value declared at the beginning of the code).

```
compute segmt2005 ;  
  if segmt2004 = '1-25 pctl' then  
    do ;  
      call define ('_c3_', "style",  
        "style = [background = #AAFFAA]");  
      call define ('_c4_', "style",  
        "style = [background = #AAFFAA]");
```

```

        call define ('_c5_', "style",
                    "style = [background = #AAFFAA]");
        call define ('_c6_', "style",
                    "style = [background = #CCCCCC]");
    end;

/* Additional code to color cells 40 and 50 based on &s0 value */
if (segmt2004 = '1-25 pctl' and &s0. > 0) then
    call define ('_c7_', "style",
                "style = [background = #FFFF99]");

    if segmt2004 = '26-50 pctl' then
        do;
            call define ('_c4_', "style",
                        "style = [background = #AAFFAA]");
            call define ('_c5_', "style",
                        "style = [background = #AAFFAA]");
            call define ('_c2_', "style",
                        "style = [background = #FFFF99]");
            call define ('_c6_', "style",
                        "style = [background = #CCCCCC]");
        end;

        Code for remaining segments (same as above)....

    endcomp;
run;
/*End 2nd block of PROC REPORT*/

```

Similar CALL DEFINE statements could be written for the remaining values of segmt2004 variable until we complete the desired color assignments.

USING HEXADECIMAL VALUES FOR COLORS

Values such as #CCCCCC (gray) and #FFFF99 (light yellow) refer to colors using the hexadecimal notation for RGB values. This notation helps to use the exact color needed. There many resources on the web^{4,5,6} which will aid finding the correct hexadecimal RGB values for each color.

CALCULATIONS USING ANALYSIS VARIABLES

In the final section of the report we are interested in displaying the data using two selected variables and also in performing additional calculations using the existing variables in the data set.

The initial code will be the same for the header of the report.

```

/*Yrs_in_2005 vs. Segment2005 - 3rd block of PROC REPORT*/
proc report data = mwsug.paper nowd
    headline headskip split = '*' colwidth = 10
    style(report)   = {font_face=Tahoma}
    style(header)   = {background=gray
                        foreground=white
                        font_weight=bold
                        font_face=Tahoma font_size=8pt}
    style(column)   = {background=white
                        foreground=black
                        font_face=Tahoma font_size=8pt}
    style(summary)  = {font_weight=bold} ;

    column          Yrs_in_2005 segmt2005
                    ("Overall (CC - last 12 mths)" Total activity2005 Avg_trans
                     activity2005=Pct_trans)
                    ('Migration*' new_non_post post_move negt_move)
                    ('Special*' active_user) ;

```

The columns of interest are specified in order. Since we specified the SPLIT = option at the top (using '*'), we can now specify the group headers for the variables selected: e.g. 'Migration' will appear on top of new_non_post, post_move and negt_move variables. Please refer to table 3 in the output shown in the last page of this paper.

At the next stage, we define the variables of interest. Yrs_in_2005 will now be the row variable and we assign it as a GROUP variable. Segmt2005 is specified as an ACROSS variable (will take column values). So in the simplest form

this will be a table between Yrs_in_2005 and Segmt2005. We can sum up the values of columns 2 to 7 and call it variable 'Total Pop'.

```
define Yrs_in_2005 / group 'Yrs_in_2005' ;
define segmt2005 / across "Segment 2005 for CC";

define Total / computed 'Total Pop' format=comma8.0;
```

At the next stage we define the ANALYSIS variables. The sum of variable activity2005 will be shown under the label 'Total trans'. We need to remember that the table is already defined in terms of two main variables 'Yrs_in_2005' and Segmt2005. Since Yrs_in_2005 is the row variable, the summation of activity2005 will be shown for each unique value of Yrs_in_2005 and will appear in the column next to "Total Pop".

Similarly, 'new_non_post', 'post_move' and 'negt_move' will sum up the number of users with value of '1' for each category in the row variable.

Variable 'avg_trans' will calculate the average number of transactions for each value in the 'Yrs_in_2005' variable. The average is calculated by dividing the total number of transactions each group carried out (column 9) by the number of users in each group (sum of columns 2 to 6). In the Report PROCEDURE we can specify this by column numbers as `_c9_ / sum(_c2_,_c3_,_c4_,_c5_,_c6_)`. We can specify the format for these columns as shown in the code below.

The 'pct_trans' variable is assigned PCTSUM type and this option will calculate the percentage of transactions carried out by each user group in variable Yrs_in_2005. Next we will calculate the row totals using RBREAK statement with a COMPUTE block as before.

```
define activity2005 / analysis sum 'Total Trans' format=comma10.0;
define Avg_trans / computed 'Avg Trans' format=comma8.1;
define Pct_trans / pctsum 'Pct Trans' format=percent8.1;

define new_non_post / analysis sum 'New_Non (+)ve' format=comma6.0;
define post_move / analysis sum 'Exist (+)ve' format=comma6.0;
define negt_move / analysis sum 'Exist (-)ve' format=comma6.0;

define active_user / analysis sum 'Active' format=comma6.0;

compute Total;
    Total = sum (_c2_,_c3_,_c4_,_c5_,_c6_,_c7_);
endcomp;

compute Avg_trans;
    Avg_trans = _c9_ / sum(_c2_,_c3_,_c4_,_c5_,_c6_) ;
endcomp;

rbreak after / dol summarize ;
compute after;
    Yrs_in_2005 = 'Total';
endcomp;
```

In the final section of the code, we aim to color code certain columns with pre-determined colors. We will use a COMPUTE block with a CALL DEFINE statement to change the background color of each column. We refer to column numbers by the `_cn_` notation as before.

```
compute new_non_post;
    call define ('_c12_', "style",
                "style = [background = #E1FFE1]");
endcomp;

compute post_move;
    call define ('_c13_', "style",
                "style = [background = #AAFFAA]");
endcomp;

compute negt_move;
    call define ('_c14_', "style",
                "style = [background = #FFFF99]");
endcomp;

compute segmt2005;
```

```

call define ('_c6_', "style",
            "style = [background = #CCCCCC]");

endcomp;

run;
/*End of 3rd block of PROC REPORT*/

```

The above block of code completes the coding for the dashboard. This should be followed by the back-end of the ODS shell code.

CONCLUSION

The paper looks at features available in the Report PROCEDURE, illustrates the use of GROUP and ACROSS variables to summarize data and use of ANALYSIS type variables to carry out calculations. The areas covered include using SAS ODS for data presentation such as producing RTF reports and using "Traffic Lighting".

The paper exhibits the use of some of the powerful features of the Report PROCEDURE. It is a great tool which combines reporting with analytical capabilities.

REFERENCES

1. A handbook of Statistical Analyses using SAS (2nd edition) – Geoff Der and Brian S. Everitt (2002)
2. Proc REPORT documentation in Base SAS 9.1.3 Procedure Guide – SAS Publishing (2004)
3. SAS with Style: Creating your own ODS template – Lauren Haworth
4. Base SAS – ODS FAQ and concepts (Colors concept) -
http://support.sas.com/rnd/base/topics/templateFAQ/Template_colors.html
5. HTML color helper - <http://www.pangloss.com/seidel/ClrHlpr/mixer.cgi>
6. Netscape (6x6x6) Color Palette map - <http://the-light.com/colclick.html>

CONTACT INFORMATION

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

Thushan Wijesinghe
LexisNexis
9443 Springboro Pike
Miamisburg, OH 45342
937-865-6800 x55234
thushan.wijesinghe@lexisnexis.com
www.lexisnexis.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.

ABC Bank Ltd
Dashboard for Credit Card [CC] customer transactions -August 2005 Produced by Customer Analytics Dept.
ABC Bank Ltd. - Confidential Information

Channel for user population by Yrs_in_2005									
Yrs_in_2005									
Channel	U	Yr01	Yr02	Yr03	Yr04	Yr05	Yr06	Yr7+	Total
Dir Mail	57	.	31	13	59	128	401	93	782
Email/Web	158	3	7	5	68	119	2062	1401	3,823
Telephonic	8	.	12	3	27	33	218	94	395
Total	223	3	50	21	154	280	2681	1588	5,000

Segment 2005 for CC							
Segment 2004 for CC	1-25 pctl	26-50 pctl	51-75 pctl	75+ pctl	Inactive	Non user	Total Population
1-25 pctl	124	1	1	.	30	65	221
26-50 pctl	3	166	27	6	50	67	319
51-75 pctl	2	16	116	16	59	38	247
75+ pctl	1	7	13	184	89	14	308
New user	19	39	19	9	130	244	460
Non user	1	15	5	2	954	2468	3,445
Total	150	244	181	217	1312	2896	5,000

Segment 2005 for CC							Overall (CC - last 12 mths)				Migration			Special
Yrs_in_2005	1-25 pctl	26-50 pctl	51-75 pctl	75+ pctl	Inacti ve	Non user	Total Pop	Total Trans	Avg Trans	Pct Trans	New_ Non (+)ve	Exist (+)ve	Exist (-)ve	Active
U	17	16	5	10	23	152	223	323	4.5	1.6%	3	3	11	1
Yr01	.	.	.	.	.	3	3	.	.	.	0	0	0	0
Yr02	5	8	7	15	4	11	50	1,251	32.1	6.2%	3	0	2	6
Yr03	.	3	1	3	5	9	21	948	79.0	4.7%	1	2	2	1
Yr04	12	15	11	16	35	65	154	1,868	21.0	9.2%	3	4	15	6
Yr05	34	26	23	32	60	105	280	2,434	13.9	12.0%	15	6	24	7
Yr06	63	135	104	107	703	1569	2,681	8,812	7.9	43.4%	50	27	88	37
Yr7+	19	41	30	34	482	982	1,588	4,677	7.7	23.0%	14	9	19	21
Total	150	244	181	217	1312	2896	5,000	20,313	9.7	100.0%	89	51	161	79