Paper 244-30

So Now You're Using PROC REPORT–Is It Pretty and Automated?

Daphne Ewing, Octagon Research Solutions, Inc., Wayne, PA
Ray Pass, Ray Pass Consulting, Greenwich, CT

ABSTRACT

PROC REPORT will display data. That's a given. That it is more powerful than the standard SAS® reporting tool PROC PRINT is also becoming common knowledge. But what can PROC REPORT really do? It is certainly more than a glorified PROC PRINT, but it does take some additional work beyond mastering some of the basics of the procedure before it can be used to produce output that can rival that of a serious DATA _NULL_ step. This paper will discuss the use of several customized macros that can work together with some PROC REPORT options, and standard SAS title and footnote statements, to make your PROC REPORT output really shine.

INTRODUCTION

The purpose of this paper is to provide some methods, through a progressive series of examples, to enhance the output created when using PROC REPORT. It is assumed that you know how to write a basic PROC REPORT program including COLUMN, DEFINE, COMPUTE, and BREAK portions of the syntax. It is also assumed you have some understanding of what a basic SAS MACRO can do and that parameters can be passed to the macros. Many of the styles discussed in this paper are preferences, but could be adapted to your individual likes and needs. Each programmer may define an output format that he/she deems appropriate for his/her applications.

THE DATA – GETTING STARTED

The data set used throughout this paper is a fictitious survey of the spending habits of 100 college students during an academic year. The source code in **Example-1** describes the data contained in an external file (C:\PROCREP\PROCREP.DAT). Although the examples in this paper are PC-based, the REPORT procedure is identical on all platforms. The example also contains code to create two SAS formats ($SEXFMT. And $YEARFMT.) which are used throughout the paper, along with a fairly basic PROC REPORT statement. This PROC REPORT program uses some standard options such as CENTER, HEADLINE and HEADSKIP. The output from this procedure may not necessarily span the entire LINESIZE (defined as a global option) but it is centered on the page nonetheless. This will become more important in further examples. **Output-1** (see appendix) displays a portion of the PROC REPORT output from this simple run (due to publishing space limitations, the full population is not included in any of the non-grouped outputs).

Another helpful feature used in this example is the column-spanning header. This feature is created by including a quoted string in parentheses preceding the variables it is to span. Using a '-' (dash) as the first and last character of the quoted string results in an expansion of the string left and right to the entire width of all the columns being spanned. There are other characters which will automatically expand left and right such as "=" and "*".

```
DATA SURVEY;
  INFILE 'C:\PROCREP\PROCREP.DAT';

  INPUT @01 FNAME    $CHAR10.
        @12 LINIT    $CHAR1.
        @15 SEX      $CHAR1.
        @18 ACADYEAR $CHAR1.
        @21 GPA      3.1
        @27 AGE      2.
        @31 BOOKS    4.
        @40 FOOD     4.
        @49 ENT      4. ;
RUN;

PROC FORMAT;
  VALUE $SEXFMT  'M' = 'MALE'
                 'F' = 'FEMALE' ;
  VALUE $YEARFMT '1' = 'FRESHMAN'
                 '2' = 'SOPHOMORE'
                 '3' = 'JUNIOR'
                 '4' = 'SENIOR' ;
RUN;
```

```
PROC REPORT NOWD DATA=SURVEY HEADLINE HEADSKIP CENTER MISSING;
    COLUMNS FNAME LINIT NAME SEX ACADYEAR GPA AGE
            ('-ANNUAL AMOUNT SPENT ON-' BOOKS FOOD ENT TOTAL);

    DEFINE  FNAME    /     DISPLAY NOPRINT;
    DEFINE  LINIT    /     DISPLAY NOPRINT;
    DEFINE  SEX      /     DISPLAY  FORMAT=$SEXFMT.  WIDTH=6;
    DEFINE  ACADYEAR /     DISPLAY  FORMAT=$YEARFMT. WIDTH=9 'ACADEMIC/YEAR';
    DEFINE  GPA      /     ANALYSIS FORMAT=3.1       WIDTH=3;
    DEFINE  AGE      /     ANALYSIS FORMAT=3.        WIDTH=3;
    DEFINE  BOOKS    /     ANALYSIS FORMAT=DOLLAR6.  WIDTH=7  'BOOKS';
    DEFINE  FOOD     /     ANALYSIS FORMAT=DOLLAR6.  WIDTH=6  'FOOD';
    DEFINE  ENT      /     ANALYSIS FORMAT=DOLLAR6.  WIDTH=11 'ENTERTAIN';
    DEFINE  NAME     /     COMPUTED                  WIDTH=10;
    DEFINE  TOTAL    /     COMPUTED FORMAT=DOLLAR6.  WIDTH=7  'TOTAL';

    COMPUTE NAME / CHAR LENGTH=10;
       NAME=TRIM(FNAME) || ' ' || LINIT || '.';
    ENDCOMP;

    COMPUTE TOTAL;
       TOTAL = SUM(BOOKS.SUM, FOOD.SUM, ENT.SUM);
    ENDCOMP;
RUN;
```

Example-1

Each report created can be seen as segmented, meaning there are several “sections” which are all pulled together to form the whole report. Usually each report will have overall titling information, column titles (describing the data), the actual data, and possibly footnotes. Assuming this is the case, making a physical distinction between these display areas makes the output clearer and easier to read. The HEADLINE option causes a natural break between column headers and the actual data. To separate overall report titles from the column headers, another dashed line can be added prior to the column headers, as shown in **Example-2**. The two title statements, TITLE3 and TITLE4 provide this break. The last division would then be between the actual data and the footnotes (or just an indication of “bottom of page”). This is also shown in the example with the first footnote line being one of dashes, and the second, a simple note about the data. The results of these title and footnote statement additions can be seen in **Output-2** as a report with more distinct sections.

```
TITLE1 'PROC REPORT - SAMPLE REPORT';
TITLE2 '------ACADEMIC SURVEY------';
TITLE3 ;
TITLE4    '--------------------------------------------------------------------------------';
FOOTNOTE1 '--------------------------------------------------------------------------------';
FOOTNOTE2 '*INDIVIDUAL DOLLAR AMOUNTS ROUNDED TO NEAREST DOLLAR.';
RUN;

PROC REPORT NOWD DATA=SURVEY HEADLINE HEADSKIP CENTER MISSING;
... (same as Example-1)
RUN;
```

Example-2

The dashed line used by the HEADLINE option, as well as the dashed line which is used when expanding across a column or columns may not be appealing to all users. In light of this, there are ways to make the output from PROC REPORT a bit more “finished”. The SAS system form characters are used to draw lines in the PROC REPORT output. When the SAS system is invoked, there are default values for these characters, but you can change them using the FORMCHAR option. By default, the FORMCHAR option is set to: |----|+|---+=|-/*. The second element within the FORMCHAR string is the one used to draw the line requested by the HEADLINE option of the PROC REPORT statement. Therefore, if you want a solid line you can do this by using the underline character ‘_’, or you can research this with your printer documentation to use a long dash ‘—’.

```
OPTIONS FORMCHAR='|_---|+|---+=|-/\<>*';

TITLE1 'PROC REPORT - SAMPLE REPORT';
TITLE2 '------ACADEMIC SURVEY------';
TITLE3 ;
TITLE4    '________________________________________________________________________________';
FOOTNOTE1 '________________________________________________________________________________';
```

```
FOOTNOTE2 '*INDIVIDUAL DOLLAR AMOUNTS ROUNDED TO NEAREST DOLLAR.                    ';
RUN;

PROC REPORT NOWD DATA=SURVEY HEADLINE HEADSKIP CENTER MISSING;
... (same as Example-1)
RUN;
```

Example-3

The code in **Example-3** shows how to change the FORMCHAR string, specifically the second character is changed from a dash to an underline character. The dividing title and footnote statements are also changed to underlines and not dashed lines to make the report more consistent. The results of these changes can be found in **Output-3**. The FOOTNOTE2 line was also enhanced by adding blanks to the end of the quote causing the footnote to be left justified.

The data in this report may be more clear if we break it up by a particular classification. To illustrate, let us look at the data for academic years separately. This can be done as shown in **Example-4** (with the TITLE and FOOTNOTE statements remaining the same as in **Example-3**) by making the ACADYEAR variable an ORDER variable and displaying it as the first item in the COLUMN list. Note also the ORDER=INTERNAL item on the DEFINE statement for ACADYEAR. This is added in order to sort by the coded year (1, 2, 3 and 4) and not the formatted value of the ACADYEAR variable. The BREAK statement, with the PAGE option, added at the end of the REPORT procedure causes each academic year to begin its listing on a new page. **Output-4** also shows how REPORT only displays the formatted ACADYEAR value for the first occurrence of each new group (or at the top of a new page), thus providing a natural break to the report.

```
PROC REPORT NOWD DATA=SURVEY HEADLINE HEADSKIP CENTER MISSING;
    COLUMNS ACADYEAR FNAME LINIT NAME SEX GPA AGE
            ('-ANNUAL AMOUNT SPENT ON-' BOOKS FOOD ENT TOTAL);

    DEFINE  ACADYEAR / ORDER  FORMAT=$YEARFMT. ORDER=INTERNAL WIDTH=9 'ACADEMIC/YEAR';
    DEFINE  FNAME    / DISPLAY NOPRINT;
    DEFINE  LINIT    / DISPLAY NOPRINT;
    DEFINE  SEX      / DISPLAY  FORMAT=$SEXFMT.  WIDTH=6;
    DEFINE  GPA      / ANALYSIS FORMAT=3.1       WIDTH=3;
    DEFINE  AGE      / ANALYSIS FORMAT=3.        WIDTH=3;
    DEFINE  BOOKS    / ANALYSIS FORMAT=DOLLAR6.  WIDTH=7  'BOOKS';
    DEFINE  FOOD     / ANALYSIS FORMAT=DOLLAR6.  WIDTH=6  'FOOD';
    DEFINE  ENT      / ANALYSIS FORMAT=DOLLAR6.  WIDTH=11 'ENTERTAIN';
    DEFINE  NAME     / COMPUTED                  WIDTH=10;
    DEFINE  TOTAL    / COMPUTED FORMAT=DOLLAR6.  WIDTH=7  'TOTAL';

    COMPUTE NAME / CHAR LENGTH=10;
       NAME=TRIM(FNAME) || ' ' || LINIT || '.';
    ENDCOMP;

    COMPUTE TOTAL;
       TOTAL = SUM(BOOKS.SUM, FOOD.SUM, ENT.SUM);
    ENDCOMP;

    BREAK AFTER ACADYEAR / PAGE;
RUN;
```

Example-4

We could have added LINE statements to create page headings, but these statements can only provide their output between the column headers and the data. There is another way, namely by creating the report with a BY variable (requiring a PROC SORT prior to the PROC REPORT). The code found in **Example-5** (with the TITLE and FOOTNOTE statements remaining the same as in **Example-3**) illustrates this technique. As you can see by the results found in **Output-5**, the BY variable is displayed under the last title line as a dashed line with the BY variable and value identified in the middle.

```
PROC SORT DATA = SURVEY;
  BY ACADYEAR;
RUN;

PROC REPORT NOWD DATA=SURVEY HEADLINE HEADSKIP CENTER MISSING;
  BY ACADYEAR;
    COLUMNS FNAME LINIT NAME SEX GPA AGE
```

```
            ('-ANNUAL AMOUNT SPENT ON-' BOOKS FOOD ENT TOTAL);

    DEFINE  FNAME    /     DISPLAY NOPRINT;
    DEFINE  LINIT    /     DISPLAY NOPRINT;
    DEFINE  SEX      /     DISPLAY  FORMAT=$SEXFMT.  WIDTH=6;
    DEFINE  GPA      /     ANALYSIS FORMAT=3.1       WIDTH=3;
    DEFINE  AGE      /     ANALYSIS FORMAT=3.        WIDTH=3;
    DEFINE  BOOKS    /     ANALYSIS FORMAT=DOLLAR6.  WIDTH=7  'BOOKS';
    DEFINE  FOOD     /     ANALYSIS FORMAT=DOLLAR6.  WIDTH=6  'FOOD';
    DEFINE  ENT      /     ANALYSIS FORMAT=DOLLAR6.  WIDTH=11 'ENTERTAIN';
    DEFINE  NAME     /     COMPUTED                  WIDTH=10;
    DEFINE  TOTAL    /     COMPUTED FORMAT=DOLLAR6.  WIDTH=7  'TOTAL';

    COMPUTE NAME / CHAR LENGTH=10;
       NAME=TRIM(FNAME) || ' ' || LINIT || '.';
    ENDCOMP;

    COMPUTE TOTAL;
       TOTAL = SUM(BOOKS.SUM, FOOD.SUM, ENT.SUM);
    ENDCOMP;
RUN;
```

Example-5

This display now appears very choppy and inconsistent, although it does present the data as we wanted it, broken out and paged by academic year. We can clean it up by using a combination of the NOBYLINE system option along with the #BYVAL feature within the TITLE statement. The NOBYLINE option causes the dashed line (including the BY variable name and its current value) to be removed from the output. We then include the current BY value (academic year) being processed in a TITLE line by using #BYVAL. In **Example-6** the ACADYEAR variable is used in a FORMAT statement in the SORT procedure (this could have been done in a prior data step as well) to associate a permanent format with the variable. This makes all output of this variable display with the formatted value of the variable. Without this statement in the code, the output would display the internal variable value (1, 2, 3 or 4) instead of the text associated with it (Freshman, Sophomore, etc.). The TITLE4 statement is moved to TITLE5, and TITLE4 is now the title line which identifies which academic year is being displayed on each page of the report.

```
TITLE1 'PROC REPORT - SAMPLE REPORT';
TITLE2 '------ACADEMIC SURVEY------';
TITLE3 ;
TITLE4 '#BYVAL(ACADYEAR) YEAR STUDENTS ONLY';
TITLE5    '_____________________________________________________________________________';
FOOTNOTE1 '_____________________________________________________________________________';
FOOTNOTE2 '*INDIVIDUAL DOLLAR AMOUNTS ROUNDED TO NEAREST DOLLAR.
';
RUN;

PROC SORT DATA = SURVEY;
  BY ACADYEAR;
  FORMAT ACADYEAR $YEARFMT.;
RUN;

PROC REPORT NOWD DATA=SURVEY HEADLINE HEADSKIP CENTER MISSING;
... (same as Example-5)
RUN;
```

Example-6

As mentioned above, the paging effect could also be achieved by using the BREAK AFTER statement. This would also give the added ability of obtaining overall summary statistics (all academic years combined) which would not be possible by using the BY method. For illustration purposes however, the first option mentioned will be described in more detail.

The display found in **Output-6** is still a little clumsy looking in that the lines dividing the report title and the column headers, as well as the line separating the actual data from the footnotes, is wider than the overall data display. This situation can be resolved by changing the LINESIZE value (LS=) to match the number of columns being used by the report, as shown in **Example-7**. When the LINESIZE (or number of columns available for print) is decreased, the printing of the SAS date/time (via the default DATE system option) becomes overbearing to the report. This example also shows how you can set the system option to NODATE but still place the run date/time on your output. The functions DATE() and TIME() are used to create the current date/time during the program's run. These are then converted to macro variables using CALL SYMPUT statements. Once these macro variables are created, they can

be used anywhere in the report. The report in **Output-7** shows the program run date and time displaying in one of the footnotes. Note that when you use macro variables in a TITLE or FOOTNOTE statement, you must quote the string using double quotes (not single quotes) in order for the macro variables to resolve.

```
OPTIONS PAGENO=1 NOBYLINE NODATE;

DATA _NULL_;
  EXEDATE=DATE();
  EXETIME=TIME();
  CALL SYMPUT('EXEDATE', PUT(EXEDATE,DATE9.));
  CALL SYMPUT('EXETIME', PUT(EXETIME,TIME5.));
RUN;

TITLE1 'PROC REPORT - SAMPLE REPORT';
TITLE2 '------ACADEMIC SURVEY------';
TITLE3 ;
TITLE4 '#BYVAL(ACADYEAR) YEAR STUDENTS ONLY';
TITLE5    '_____________________________________________________________________________';
FOOTNOTE1 '_____________________________________________________________________________';
FOOTNOTE2 '*INDIVIDUAL DOLLAR AMOUNTS ROUNDED TO NEAREST DOLLAR.                          ';
FOOTNOTE3 "RUN DATE/TIME: &EXEDATE &EXETIME                                              ";
RUN;

PROC SORT DATA = SURVEY;
  BY ACADYEAR;
  FORMAT ACADYEAR $YEARFMT.;
RUN;

PROC REPORT NOWD DATA=SURVEY LS=67 HEADLINE HEADSKIP CENTER MISSING;
... (same as Example-5)
RUN;
```

Example-7

There is another method which can be used to make the display appear more balanced. This is to modify the SPACING between columns as well as modifying the WIDTH each column is to use. **Example-8** shows how this is achieved using the SPACING=<value> in the PROC REPORT statement, as well as adding individual SPACING=<value> syntax in the DEFINE statements, and increasing the WIDTH= values. The results found in **Output-8** show how the report now spans the full LINESIZE (let's assume this was previously set to a production standard of 96).

```
PROC REPORT NOWD DATA=SURVEY HEADLINE HEADSKIP CENTER MISSING SPACING=5;
  BY ACADYEAR;
    COLUMNS FNAME LINIT NAME SEX GPA AGE
            ('-ANNUAL AMOUNT SPENT ON-' BOOKS FOOD ENT TOTAL);

    DEFINE  FNAME    /     DISPLAY NOPRINT;
    DEFINE  LINIT    /     DISPLAY NOPRINT;
    DEFINE  SEX      /     DISPLAY  FORMAT=$SEXFMT.  WIDTH=6 SPACING=8;
    DEFINE  GPA      /     ANALYSIS FORMAT=3.1       WIDTH=3 SPACING=7;
    DEFINE  AGE      /     ANALYSIS FORMAT=3.        WIDTH=3 SPACING=7;
    DEFINE  BOOKS    /     ANALYSIS FORMAT=DOLLAR6.  WIDTH=8  'BOOKS';
    DEFINE  FOOD     /     ANALYSIS FORMAT=DOLLAR6.  WIDTH=6  'FOOD';
    DEFINE  ENT      /     ANALYSIS FORMAT=DOLLAR6.  WIDTH=11 'ENTERTAIN';
    DEFINE  NAME     /     COMPUTED                  WIDTH=10;
    DEFINE  TOTAL    /     COMPUTED FORMAT=DOLLAR6.  WIDTH=7  'TOTAL';

    COMPUTE NAME / CHAR LENGTH=10;
       NAME=TRIM(FNAME) || ' ' || LINIT || '.';
    ENDCOMP;

    COMPUTE TOTAL;
       TOTAL = SUM(BOOKS.SUM, FOOD.SUM, ENT.SUM);
    ENDCOMP;
RUN;
```

Example-8

Consistency of output and ease of providing that consistency become more important when the volume of output is large, and/or the number of programmers producing the output is more than one. The first suggestion for achieving this would be to place all generic information in a central location. The following items would be good candidates:

- Generic titles
- Generic footnotes
- Macro variables for underlines, dashed lines, blank lines
- Macro variables for LINESIZE and PAGESIZE
- Generic BY line text
- Definition of the FORMCHAR option
- Generic options statements (NOBYLINE, NUMBER, etc.)

One option is to create a report initialization file where this information can then be "included" by each of the report programs. Below is a sample initialization file (**RPTINIT.SAS**) with some additional "tricks" included that can be very handy.

```
OPTIONS FORMCHAR='|_---|+|---+=|-/\<>*';
OPTIONS PAGENO=1 NOBYLINE NODATE;
RUN;

/* Set Line size and Page size as macro variables */
%LET LNSIZE = 96;
%LET PGSIZE = 54;
RUN;

/* Prepare some general macro variables */
DATA _NULL_;
  CALL SYMPUT("UNDERLN",REPEAT('_',95));
  CALL SYMPUT("DASHLN",REPEAT('-',95));
  CALL SYMPUT("BLANKLN",REPEAT(' ',(&LNSIZE-1)));

  EXEDATE=DATE();
  EXETIME=TIME();
  CALL SYMPUT('EXEDATE', PUT(EXEDATE,DATE9.));
  CALL SYMPUT('EXETIME', PUT(EXETIME,TIME5.));
RUN;


/* General Titles */
%LET GTTL1  = PROC REPORT - SAMPLE REPORT;
%LET GTTL2  = ACADEMIC SURVEY;
RUN;

/* General BY Title */
%LET GBYTTL = #BYVAL(ACADYEAR) YEAR STUDENTS ONLY;
RUN;

/* General Footnotes */
%LET GFTN1 = &UNDERLN;
%LET GFTN2 = *INDIVIDUAL DOLLAR AMOUNTS ROUNDED TO NEAREST DOLLAR.;
RUN;
```

Sample RPTINIT.SAS file: General

These statements can also be included in an AUTOEXEC file that is run at SAS invocation time. The use of macro variables for the LINESIZE and PAGESIZE can be very powerful in that you can change the values in the one file, and as seen in the CALL SYMPUT statement with the BLANKLN value, this change can be global.

The next step in creating consistent output is to use macros to help divide the report into the titling, actual data (with column headers), and the footnote sections. These macros could be put into an AUTOCALL macro library (with the exception of the **RPTDEF macro** below). The basic titling for the report can be moved into a macro such as the following:

```
%MACRO RPTTTL(PAGEN=1);
  DATA _NULL_;
    _LEFT="&GTTL2";
    IF (_LEFT EQ ' ') THEN LEN_LT = 0;
    ELSE LEN_LT = LENGTH(_LEFT);
    _RIGHT = "&EXEDATE &EXETIME";
    LEN_RT = LENGTH(_RIGHT);
    BLNKCT = &LNSIZE - LEN_LT - LEN_RT;
    CALL SYMPUT("TTL2", _LEFT
                        ||REPEAT(' ',BLNKCT-1)
                        ||_RIGHT);
  RUN;

  OPTIONS PAGENO=&PAGEN;

  TITLE1 "&GTTL1 &BLANKLN";
  TITLE2 "&TTL2";
  RUN;
%MEND;
```

Sample RPTINIT.SAS file: RPTTTL macro

This version of the **RPTTTL macro** results in a few changes to the "look" of the report. To illustrate the ability to provide a title line which is fully justified, the second title line has been modified to put the global title 2 (&GTTL2) on the left hand side of the line with the SAS date/time stamp added right justified on the same title line. This is the purpose of the DATA _NULL_ in the **RPTTTL macro** defined above. This macro assumes that all the reports are going to have these same "title" lines. If your output is not as consistent, there are ways to pass in "changing" titles to a macro that will be described in more detail below. This macro also provides the flexibility of modifying the beginning page number for the report by using the macro variable PAGEN, which is defaulted to 1 since most reports begin on page 1.

The footnoting can be prepared in a similar fashion as shown below in the **RPTFTN macro**:

```
%MACRO RPTFTN;
  FOOTNOTE1 "&GFTN1";
  FOOTNOTE2 "&GFTN2 &BLANKLN";
  RUN;
%MEND;
```

Sample RPTINIT.SAS file: RPTFTN macro

Notice the second footnote has been left justified by adding the extra blanks to the line (&BLANKLN) which was defined in the initialization file.

The actual report definition (PROC REPORT code) will be specific to each program, not generic across many programs. This portion of the program can be grouped by placing the report code within a macro that is always named the same thing (e.g. %RPTDEF). The code below shows the report definition macro (**RPTDEF macro**) which can then be called by a generic program described later in this paper. By placing program specific report code in a consistently named macro, the "report" can be run by a generic macro which calls this "specific" code in a generic manner. This is described in more detail in the **RPTRUN macro**. By including the macro parameter DSET in RPTDEF, the program can be run against a data set specified at runtime via RPTRUN.

```
%MACRO RPTDEF(DSET,PAGEN=1);
PROC SORT DATA = &DSET;
  BY ACADYEAR;
  FORMAT ACADYEAR $YEARFMT.;
RUN;

PROC REPORT NOWD DATA=&DSET HEADLINE HEADSKIP CENTER MISSING SPACING=5;
  BY ACADYEAR;
    COLUMNS FNAME LINIT NAME SEX GPA AGE
            ('-ANNUAL AMOUNT SPENT ON-' BOOKS FOOD ENT TOTAL);

    DEFINE  FNAME    /     DISPLAY NOPRINT;
    DEFINE  LINIT    /     DISPLAY NOPRINT;
    DEFINE  SEX      /     DISPLAY  FORMAT=$SEXFMT.  WIDTH=6 SPACING=8;
    DEFINE  GPA      /     ANALYSIS FORMAT=3.1       WIDTH=3 SPACING=7;
    DEFINE  AGE      /     ANALYSIS FORMAT=3.        WIDTH=3 SPACING=7;
    DEFINE  BOOKS    /     ANALYSIS FORMAT=DOLLAR6.  WIDTH=8  'BOOKS';
```

```
    DEFINE  FOOD     /     ANALYSIS FORMAT=DOLLAR6.  WIDTH=6  'FOOD';
    DEFINE  ENT      /     ANALYSIS FORMAT=DOLLAR6.  WIDTH=11 'ENTERTAIN';
    DEFINE  NAME     /     COMPUTED                   WIDTH=10;
    DEFINE  TOTAL    /     COMPUTED FORMAT=DOLLAR6.  WIDTH=7  'TOTAL';

    COMPUTE NAME / CHAR LENGTH=10;
       NAME=TRIM(FNAME) || ' ' || LINIT || '.';
    ENDCOMP;

    COMPUTE TOTAL;
       TOTAL = SUM(BOOKS.SUM, FOOD.SUM, ENT.SUM);
    ENDCOMP;
RUN;
%MEND RPTDEF;
```

Sample RPTINIT.SAS file: RPTDEF macro

Now that the different sections of the report have been prepared via macros, a typical program will look like **Example-9** below. The results of this program, found in Output-9, show a display almost identical to **Output-8** except for the minor change in the second title line (full justification example). This program is more robust in that only the portions specific to this report have been included.

```
%MACRO RPTDEF(DSET,PAGEN=1);
... (Same report definition as in RPTDEF Example)
%MEND RPTDEF;

%RPTTTL;

/* REPORT SPECIFIC TITLES */
TITLE3 "&GBYTTL";
TITLE5 "&UNDERLN";
RUN;

%RPTFTN;
%RPTDEF(SURVEY);
```

Example-9

In the above code, there really are only two sections which are unique to a given run. These are the titling specific to the data involved, and the actual PROC REPORT code. In light of this, one more macro can be defined to pull all the mentioned macros together providing for ease of programming. The following macro (**RPTRUN macro**) allows for the necessary flexibility.

```
%MACRO RPTRUN(RTTL1, RTTL2, RTTL3, DSET=,
              PAGEN=1);
OPTIONS CENTER NUMBER NOBYLINE NODATE;
RUN;

%RPTTTL;

TITLE3 "&RTTL1";
%IF "&RTTL3" NE "" %THEN %DO;
  TITLE4 "&RTTL2";
  TITLE5 "&RTTL3";
  TITLE6 "&UNDERLN";
%END;
%ELSE %IF "&RTTL2" NE "" %THEN %DO;
  TITLE4 "&RTTL2";
  TITLE5 "&UNDERLN";
%END;
%ELSE %IF "&RTTL1" NE "" %THEN %DO;
  TITLE4 "&UNDERLN";
%END;
%ELSE %DO;
  %PUT ***RPTRUN ERROR: NO REPORT TITLES PASSED
       TO REPORT;
%END;
RUN;

%RPTFTN;
```

```
%RPTDEF(&DSET,PAGEN=&PAGEN);

%MEND;
```

RPTRUN macro

The report code is thus reduced to the RPTDEF macro definition, and a call to the RPTRUN macro with the appropriate sub titles and data set being passed to it as show in the **Example-10** code.

```
%MACRO RPTDEF(DSET,PAGEN=1);
... (Same report definition as in Example-11)
%MEND RPTDEF;

%RPTRUN(&GBYTTL, DSET=SURVEY);
```

Example-10

The above example shows how the actual program code can be reduced to one macro definition (RPTDEF macro) along with a call to one macro (RPTRUN macro) which will then prepare the titling and footnoting that needs to occur. The output found in **Output-10** is identical to that of **Output-9** except that the source code used to produce it has been simplified.

CONCLUSION

The REPORT procedure can be as effective as the DATA _NULL_ step for displaying data. With the use of some global macros and macro variables, the output can be consistent among numerous programmers, and numerous report runs.

So, now that you are using PROC REPORT, you CAN make it more appealing and efficient.

REFERENCES

SAS® Guide to the Report Procedure, Reference, Release 6.11

SAS® Technical Report P-258, Using the REPORT Procedure in a Nonwindowing Environment, Release 6.07

ACKNOWLEDGMENTS

Many of the ideas for this paper were generated while Daphne was working at IBAH, Inc.

CONTACT INFORMATION

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

Daphne Ewing, Octagon Research Solutions, Inc.
Voice: (610) 265-8300 x598
FAX: (610) 265-8188
EMS: dewing@octagonresearch.com

Ray Pass, Ray Pass Consulting
Voice: (203) 625-0777
FAX: on request
EMS: raypass@att.net