Paper 61-27

Report Creation Using Data _NULL_

Caroline Bahler, Meridian Software, Inc., Raleigh NC
Eric Brinsfield, Meridian Software, Inc., Raleigh NC

ABSTRACT

Reports and tables can be created using PROC REPORT, PROC PRINT and other procedures. However, there are times when the standard procedure output cannot meet the report requirements. In those cases, reports creation using the DATA _NULL_ step is the solution. The objective of this paper is to discuss the techniques used to create a data _NULL_ report.

INTRODUCTION: THE _NULL_ DATA STEP

Report creation within base SAS® can be accomplished through the use of multiple procedures such as PROC REPORT and PROC PRINT. The choice of procedure is dependent upon the needs of the report. In general, the strength of reporting procedures is to present data in a tabular form and allows summarization of the data and formatting of the output. Although capable of handling about 90% of typical report requests, these procedures do have limits. There are cases when the report requirements are not satisfied by any of the standard reporting procedures. The solution in those cases is to use the _NULL_ DATA step.

The _NULL_ DATA step is a specialized case of the DATA step. When you use the _NULL_ key word, the DATA statement processes all statements within the DATA step without data set creation[1].

Some of the uses of DATA _NULL_ are:

- To create customized reports with and without ODS. The reports can be routed to the Output window, directly to a printer, to an external file, or any destination that can be defined by the FILE and FILENAME statement.
- To create macro variables based on values obtained from an input SAS data set or external file.
- To create ASCII data (flat) files for importing into MS Excel®, MS Access® or other programs and databases.
- To execute special functions or call routines available within the DATA step, such as CALL EXECUTE.

The objective of this paper is to discuss the major aspects of creating a custom report using the DATA _NULL_ step.

CREATING CUSTOM REPORTS

PREPARATION

A custom report needs a set of requirements that define precisely how the report should be laid out. The requirements should identify:

- Type of output needed: printed, viewed in Web browser, or routed to a file
- Titles, footnotes
- Summary lines
- Columns within the report and how the columns should be summarized
- Placement of information on the report “page” and timing of page breaks
- Depending on the type of report being created preparation may include creating a mock-up of the report that contains column placement for each element in the report.

COMPONENTS

Output printer/external file location definition:

- FILE – this statement defines the destination for PUT statement output. FILE can point to PRINT, LOG, or a physical location to write the report to along with any properties of that file or a printer[1]. FILE is used only within the DATA step and can utilize the FILEREF defined within a FILENAME[1] or the ODS [2] statements.
- The FILE statement deserves special study by beginning SAS programmers. Examine all of the options available in the FILE statement, because it can provide valuable information, such as the number of lines left on the page (LINESLEFT=).
- FILENAME (optional) – this statement is used to define a physical location of an external file. A FILEREF is defined within the statement and used later within a FILE statement. This statement is specified outside of a DATA step.
- ODS (optional) – the ODS (Output Delivery System) statement is used to define the type of file that will be written to the defined physical location. This statement is specified outside of a DATA step.

Report Output Definition

- PUT – this statement defines the information written to the file defined within the FILE statement[1]. A PUT statement can contain both variables and text contained within quotes. The PUT statement is similar to the INPUT statement and the same options apply[1]. Multiple PUT statements can be used within the DATA step. The following example illustrates the use of the PUT statement to create a report row containing a label and variable value:

```
DATA step statements ……
PUT @01 'Total Sales'
    @15  sales dollar10.2;
```

  In the example above, the text will be written starting in the first column of the file being created and the actual sales amount will be written starting in the 15th column. Specification of the actual column to write the report information is not necessary but useful in certain situations.
- BY – is used to define data display, paging, and summarization order within a report. This statement is used with previously sorted data sets and contains a list of variables in the specific sort order[1]. Use of the BY statement causes the creation of FIRST. and LAST. variables or each variable listed within the BY statement[3]. These variables can be used to determine summarization points and when to output data to the report file or printer.

  FIRST.variable – defines the first occurrence of a BY variable value within the data set[3].

  LAST.variable - defines the last occurrence of a BY variable value within the data set[3].

  For example the following statement,

```
BY STATE;
```

  defines the variables LAST.STATE and FIRST.STATE.

OPTIONS

- RETAIN – this statement changes how the DATA step handles a variable's value. Normally, the value of every variable is reset for each new observation. The RETAIN statement tells the DATA step not to reset the specified variables value. The main use of the RETAIN statement is to create totals or to compare variable values on the current row to values from an earlier row.
- SET – there are several options of interest that can be used to create reports.
  - END = *var* - this option create a variable that contains an identifier that specifies the end of the file.
- _N_ - this is an automatic counter variable created by the DATA step. It can be used to output information at specific observations.

  For example, when generating HTML, several statements need to occur at the top of the file only. The following statements will write the information only for the first observation.

```
data _NULL_;
    set salesdata;
      file RPT;
      if _N_ = 1 then
            put @01 "<HTML>" ……..;
    run;
```

- Formats – PROC FORMAT can be used to create custom formats that can be used to label variables within a report.

REPORT OUTPUT ROUTING

The DATA _NULL_ step can be defined to route its output to a printer or external file. The following is a list of destinations:

- Sending output to the OUTPUT window – using the FILE PRINT statement when in Display Manager will send the report output to the OUTPUT window. This behavior is useful for previewing the report. For example:

```
data _NULL_;
     set salesdata;
     FILE PRINT;
     …
PUT ……..
run;
```

- Sending output directly to a printer – this requires definition of the printer within a FILENAME statement. For example:

```
FILENAME PRINTER1  PRINTER options ;
data _NULL_;
     set salesdata;
     FILE PRINTER1;
     …….
     PUT ……
run;
```

- Sending output to an external file – the report output is sent to an external file by either specifying an external file location within the FILE or FILENAME statements. For example:
  - Using a FILE statement –

    ```
    FILE 'D:\OUTPUT\REPORT1';
    ```
  - Using a FILENAME statement –

    ```
    FILENAME RPT 'D:\OUTPUT\RPT1'
    ```
- Creating special output files – the ODS statement can be used to create external files that are in specific output formats. For example –
  - ODS HTML – creates HTML output
  - ODS PRINTER – creates postscript files.
  - ODS RTF –creates rich text format files (used by word processors).

  To create an HTML file, for example –

```
ODS HTML BODY="D:\HTML\RPT.HTM" ;
data _NULL_;
   set salesdata;
   FILE PRINT ODS=options;
put _ods_;
run;
```

  Note: The FILE statement specifies the PRINT fileref and ODS keyword. This indicates that an ODS destination has been defined.

EXAMPLE REPORTS

EXAMPLE1: SIMPLE CASE WITH NO INPUT DATA

As a simple example that shows the use of many of the components described above, the following program creates a daily calendar for any time period specified in the do-loop. More sophisticated uses would utilize the FILE statement's HEADER= for controlling the page headers and column labels and LINESLEFT= to control the paging and construct the footnotes.

Program comments have been removed from all sample code to save space in this document.

```
%let startdate=01apr2002;
%let stopdate =30apr2002;
options linesize=76 pagesize=74;
%let col1=16;
%let col2=26;
DATA _null_;
   file print notitles;
   do date="&startdate"d to "&stopdate"d;
      page+1;
      pagec=left(put(page,3.));
      if page > 1 then put _page_;
      put  ///;
      put @&col1 date weekdate37.
          @&col2 +40 'Page: ' pagec
        / @&col2 50*'-' @&col2 +50 '*';
      do time='6:30't to '19:45't by '0:15't;
         if time=hms(hour(time),0,0) then
            put @&col2 +2 48*'-'
                @&col2 +50 '|';
         put    @&col2 +2 time time5.
                @&col2 +50 '|';
      end;
      put @&col2 +2 48*'-' @&col2 +50 '*';
   end;
run;
```

In this case, the programmer assumes that all lines for one day will fit on one page; so intelligent paging was not necessary. The output is routed to the default print file, which is the OUTPUT window in Display Manager. When writing custom report programs, you have to coordinate the number of lines you want to print on each page and the number of columns with the SYSTEM OPTIONS PAGESIZE and LINESIZE.

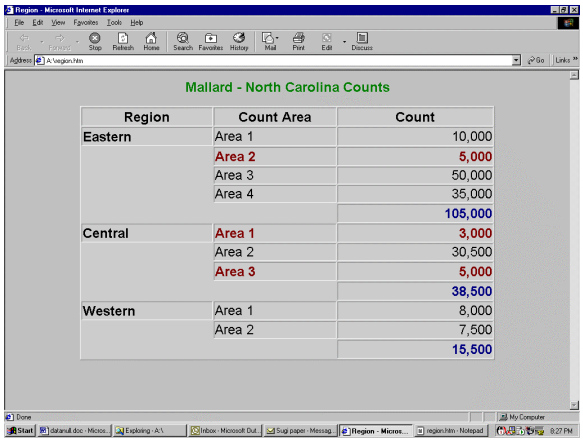
EXAMPLE 2: GENERATING HTML

Hypertext Markup Language (HTML) can be generated within a _NULL_ DATA step by either specifying the HTML tags within the PUT statement or using ODS. The following examples will describe the steps needed to use both approaches.

The report generated contains regions, count areas, and counts for Mallard ducks in North Carolina.

HTML Code Generation

A HTML file is essentially a file that contains tags that tell a browser how to format the text within the file.



Mallard - North Carolina Counts

Region	Count Area	Count
Eastern	Area 1	10,000
	Area 2	5,000
	Area 3	50,000
	Area 4	35,000
		105,000
Central	Area 1	3,000
	Area 2	30,500
	Area 3	5,000
		38,500
Western	Area 1	8,000
	Area 2	7,500
		15,500

Figure 1 HTML Report Mockup

The report in Figure 1 has total lines in blue text and lines with counts below 6,000 in red. To generate the HTML for this report within the DATA _NULL_ step:

Summarize the data and perform any other manipulations needed before going into the last DATA step.

Totals can be created within the DATA _NULL_ step so make sure that the data is sorted appropriately. In this case, by region and area. Note that a BY statement is needed after the SET statement.

```
data _null_;
set mallard end=eof;
by region area;
```

Create the DATA _NULL_ step.

- Define the file where the HTML will be written:

```
File 'H:\HTML\MALCOUNT.HTM';
```

- Create the totals using a RETAIN statement[1] and FIRST. and LAST. logic[1].

```
retain regtot;
   if first.region then regtot=count;
   else regtot=regtot + count;
```

Note: the RETAIN statement will tell the DATA step to keep the value of REGTOT instead of resetting it with the next observation. The FIRST. logic tell the DATA step to set the value of REGTOT to COUNT when a new value of REGION occurs.

Or, you can use an alternative method using a SUM statement:

```
if first.region then regtot=count;
else regtot+count;
```

In this usage, the values of REGTOT are automatically retained, because of the missing equal sign in the statement:

```
else regtot+count;
```

- Use the PUT statement to write HTML to the specified file. The HTML created in the mock up can be cut and paste into a set of put statements to create the report. Appendix A has the complete code for this example. Note that _N_ is used to define the top of the HTML file.

```
if _n_ = 1 then
    put @01 "<html>"
          /@01 '<body bgcolor="#C0C0C0">' …
```

FIRST. logic is used to define the first row of each region.

```
if first.region then do;
     put @01 '<tr>'
          /@01 '<td width="32%"  ……
```

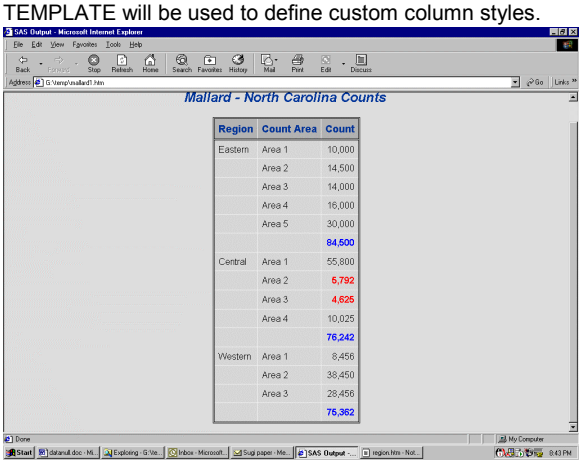
To end the HTML the END = option was used to define the end of file.

```
 if eof then
    put @01 '</table>'
         /@01 '</body>'
         /@01 '</html>'
 ;
```

EXAMPLE 3: HTML CODE GENERATION WITH ODS

To generate the same report using ODS, the PROC TEMPLATE statement will be used to specify the special formatting for total lines (in blue) and counts below 6,000 in red.

PROC TEMPLATE is a procedure that can be used to specify how different portions of a HTML file should be rendered in HTML. A set of default templates has been defined for all procedures and the DATA step. In this example, PROC TEMPLATE will be used to define custom column styles.



Mallard - North Carolina Counts

Region	Count Area	Count
Eastern	Area 1	10,000
	Area 2	14,500
	Area 3	14,000
	Area 4	16,000
	Area 5	30,000
		84,500
Central	Area 1	55,800
	Area 2	5,792
	Area 3	4,625
	Area 4	10,025
		76,242
Western	Area 1	8,456
	Area 2	38,450
	Area 3	28,456
		75,362

Figure 2: Report Generated Using ODS to Generate HTML

Summarize the data and perform any other manipulations needed before going into the last DATA step.

Totals can be created within a DATA step so make sure that the data is sorted appropriately. Other summarization methods can

also be used.

Use PROC TEMPLATE to create specialized HTML formatting. To define a custom style for a table the DEFINE TABLE is used.

```
proc template;
   define table shared.cellstyle;
```

Character variables are formatted by the following statement:

```
define column char_var;
   generic=on;
   blank_dups=on;
end;
```

The following statement is used to define two levels for the numeric column (count) and a description.

```
nmvar low  'Use default style.'
      high 'Use blue text and a bold font.';
classlevels=on;
```

Numeric variable format is defined by the following statement:

```
define column num_var;
   generic=on;
   justify=on;
```

CELLSTYLE defines the format for numeric values below above the cutoff values.

```
cellstyle _val_ < low as data
   {foreground=red
   font_weight=bold},
   _val_ <= high as data,
  /* format for regional totals */
   1 as data
   {foreground=blue font_weight=bold}
   ;
   end;
  end;
run;
```

Specify the physical location of the file using the ODS HTML statement and BODY option.

Create the HTML using a DATA _NULL_ step. Macro variables that have the same names as the values of NMVAR within PROC TEMPLATE are used to specify the cutoff values for each group.

```
/* set the cutoff for the low and high values */
   %let low=6000;
   %let high=60000;
```

In addition, within the FILE PRINT ODS statement the Template and Columns options are specified. The Columns option is used to specify which variables to used within the report and to identify the variable type (character or numeric).

```
data _null_;
   set mallard1;
   file print
   ods=(template='shared.cellstyle'
       columns=(
       char_var= reglabel(generic=on)
       char_var=arealab(generic=on)
       num_var=count(generic=on) ));
put _ods_;
run;
```

REPORT CREATION TIPS

- A complete set of requirements for the report is extremely useful when creating a custom report. The requirements should include the type of output created, summarization levels and page breaks (if any).
- Create a mock-up of the report. The mock up needs to identify all the elements of the report. A mock-up should include title and footnote placement. In some cases the creating a ruler with column numbers at the top of the mock-up is both useful and necessary.
- ODS use – ODS should be used when the report needs to be read by either a word processor or a printer.
- Generating HTML – whether to use ODS to generate HTML depends upon the report. ODS does not handle situations where the report requires the use of specialized tags. In those cases, using the PUT statement to generate the HTML is necessary.

SUMMARY

Report creation can be accomplished through the use of many procedures. However, there are cases when the procedures cannot produce the report request. In those cases, the _NULL_ DATA step is the solution.

REFERENCES

1. SAS Institute Inc. 1999. Statements. Dictionary of Language Elements, SAS Language Reference, Base SAS Software. SAS OnlineDoc®, Version 8, Cary, NC: SAS Institute Inc.
2. SAS Institute Inc. 1999. Using the Output Delivery System in the DATA step, Guide to the Output Delivery System, Reference, Base SAS Software. SAS OnlineDoc®, Version 8, Cary, NC: SAS Institute Inc.
3. SAS Institute Inc. 1991. SAS Language and Procedures: Usage 2, Version 6, First Edition, Cary, NC: SAS Institute Inc. pp 294-296.
4. SAS Institute Inc. 1991. SAS Language and Procedures: Usage 2, Version 6, First Edition, Cary, NC: SAS Institute Inc. pp 246.
5. SAS Institute Inc. 1991. SAS Language and Procedures: Usage 2, Version 6, First Edition, Cary, NC: SAS Institute Inc. pp 311.

CONTACT INFORMATION

Caroline Bahler

Meridian Software, Inc.

12204 Old Creedmoor Road

Raleigh, NC 27613

(919) 387-6735

merccb@meridiansoftware.com

www.meridiansoftware.com

Eric Brinsfield

Meridian Software, Inc.

12204 Old Creedmoor Road

Raleigh, NC 27613

(919) 847-6750

merecb@meridiansoftware.com

www.meridiansoftware.com

TRADEMARKS

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. Meridian Software, Inc.® is a registered trademark of Meridian Software, Inc. Other brand and product names are registered trademarks or trademarks of their respective companies.

APPENDIX

The data used in the following examples are Mallard duck counts.

A. GENERATING HTML

First summarize the data.

```
/* Create format for region */
proc format;
   value $ reg A='Eastern'
               B='Central'
               C='Western'
  ;
run;

/* Sort data set Mallard by region and area */
proc sort data=mallard;
   by region area;
run;

data _null_;
   set mallard end=eof;
   by region area;
   format region $reg.;
   retain regtot;
   file "g:\temp\mallard.htm";

  /* create regional totals */
  if first.region then regtot=count;
     else regtot=regtot+count;

  arealab = 'Area '||put(area,best.);

/* Start HTML and create title and column headers,
HTML and BODY tags are required */
  if _n_ = 1 then
    put @01 "<html>"
       /@01 '<body bgcolor="#C0C0C0">'
       /@01 '<h3 align="center">
            <font face="Arial"
                 color="#008000">'
         'Mallard - North Carolina Counts</font>
              </h3>'
    /* start table definition */
       /@01 '<table border="1" width="75%"
             bgcolor="#CCCCCC" align=center>'
    /* define column headers */
       /@01 '<tr>'
       /@01 '<td width="32%" height="10"
       align="center"><font face="Arial"
                      size="2"> '
            '<b>Region</b></font></td>'
       /@01 ' <td width="30%" height="10"
             align="center"><b><font
             face="Arial"
             size="2">'
            'Count Area</font></b></td>'
       /@01 '<td width="38%" height="10"
             align="center"><b><font
             face="Arial"
             size="2"> '
            ' Count</font></b></td>'
       /@01 ' </tr>'
    ;

/* create first row within a region - include
region label */
  if first.region then do;
     put @01 '<tr>'
        /@01 '<td width="32%" height="19"><font
              size="2" face="Arial"><b>' region
             '</b></font></td>';
/* if number of mallards is less than 6000 the row
is in red */
     if count < 6000 then
       put  @01 '<td width="30%"
                height="19"><font size="2"
                face="Arial"color="#800000"><b>'
                arealab '</b></font></td>'
            /@01 '<td width="38%" height="19"
                 align="right"><font size="2"
                 face="Arial" 'color="#800000">
                 <b>' count comma12.
                 '</b></font></td></tr>'
       ;
     else
       put  @01 '<td width="30%"
                 height="19"><font
                size="2" face="Arial">' arealab
                '</font></td>'
            /@01 '<td width="38%" height="19"
                 align="right"><font size="2"
                 face="Arial">'
                 count comma12.
                 '</font></td></tr>'
       ;
  end;
  else do;
     put @01 '<tr>'
          /@01 '<td width="32%"
               height="19"></font></td>';
     if count < 6000 then
       put  @01 '<td width="30%"
                height="19"><font size="2"
                face="Arial"color="#800000"><b>'
                arealab '</b></font></td>'
           /@01 '<td width="38%" height="19"
                align="right"><font size="2"
                face="Arial"
                color="#800000"><b>'
```

```
                count comma12.
                '</b></font></td></tr>'
      ;
    else
      put  @01 '<td width="30%"
               height="19"><font
               size="2" face="Arial">'
               arealab '</font></td>'
          /@01 '<td width="38%" height="19"
                 align="right"><font size="2"
                 face="Arial">'
                 count comma12.
                 '</font></td></tr>'
      ;
  end;

  /* region totals - number is in blue */
  if last.region then do;
     put @01 '<tr>'
          /@01 '<td width="32%"
               height="19"></font></td>'
          /@01 '<td width="30%"
               height="19"></font></td>'
          /@01 '<td width="38%" height="19"
               align="right">
               <font size="2" face="Arial"
               color="#000080"><b>'
               regtot comma12.
               '</b></font></td></tr>'
     ;
  end;
  if eof then
    put @01 '</table>'
         /@01 '</body>'
         /@01 '</html>'
    ;
run;
```

B. GENERATING HTML USING ODS

First summarize the data.

```
/* Create format for region */
proc format;
   value $ reg A='Eastern'
             B='Central'
             C='Western'
  ;
run;

/* Sort data set Mallard by region and area */
proc sort data=mallard;
   by region area;
run;

/* create  regional totals and labels */
data mallard1;
   set mallard end=eof;
   by region area;
   retain regtot;
   format count regtot comma12.
          reglabel arealab $12.;
   label reglabel="Region"
         arealab ="Count Area"
         count   ="Count";

   arealab = 'Area '||left(put(area,best.));
   reglabel=' ';

   if first.region then do;
      regtot=count;
      reglabel=put(region,$reg.);
      output;
   end;
      else do;
        regtot=regtot+count;
        reglabel=' ';
        output;
      end;

   if last.region then do;
      reglabel=' ';
      arealab=' ';
      count=regtot;
      output;
   end;
run;

/* Create a template that will be used by ODS to
format count column */
proc template;
   define table shared.cellstyle;
   define column char_var;
      generic=on;
      blank_dups=on;
   end;

   nmvar low  'Use default style.'
              high 'Use red foreground and a bold
font.';

   classlevels=on;

   define column num_var;
      generic=on;
      justify=on;
      cellstyle _val_ < low as data
           {foreground=red font_weight=bold},
                _val_ <= high as data,
       /* format for regional totals */
                1 as data
           {foreground=blue font_weight=bold}
               ;
   end;
  end;
run;
```

```
ods html body="g:\temp\mallard1.htm";
title 'Mallard - North Carolina Counts';

/* set the cutoff for the low and high values */
%let low=6000;
%let high=60000;
data _null_;
  set mallard1;
  file print ods=(
   objectlabel='Mallard - North Carolina Counts'

   template='shared.cellstyle'
   columns=(
    char_var=reglabel(generic=on)
    char_var=arealab(generic=on)
    num_var=count(generic=on)
            )
  );
  put _ods_;
run;

ods html close;
```