PharmaSUG 2012 - Paper TF13

Rediscovering the DATA _NULL_ for Creating a Report, And Putting That Text File Into RTF Format In a Single Datastep

David Franklin, TheProgrammersCabin.com, Litchfield, NH

ABSTRACT

Before PROC REPORT existed, most text output files were created by PROC PRINT and PROC TABULATE, or where a report was needed that needed a bit more sophistication, DATA _NULL_ was used.

This paper rediscovers the DATA _NULL_ as it is used for creating a text output file and presents a few macros, tricks and techniques that will make your report shine, including a macro for wrapping text so it can go onto multiple lines, doing “Page x of y” processing, centering or right aligning text, page breaking where you want it, and carrying over group values from one page to another. Finally, the output that is produced from a full example will be transformed into an RTF file by a macro in a single datastep.

INTRODUCTION

The DATA _NULL_ is one of the most flexible methods for generating a text output file of your report. As a programmer you are able to control where a character is put on the page and where page breaks begin. It is also possible to wrap text over multiple lines and place them where you want them to be.

Presented will be a few examples, macros, tricks and techniques that will make generating a report by this manner easier to do so.

It must be noted at this point that the DATA _NULL_ technique presented here produces a text output file that can be read in a text editor, and therefore by its very nature does not allow for proportional fonts like Times Roman.

THE PUT STATEMENT IS KEY

In its basic form, the DATA _NULL_ takes information and puts it in the SAS LOG by using the PUT statement. The syntax of the PUT statement is very involved, but an abbreviated version is given below:

PUT
<n *>"character-string" -- writes out the given quoted character string n times (if n is missing, then output once)
@(n | numeric-variable | (expression)) -- moves the column pointer to the given column
+(n | numeric-variable | (expression)) -- moves the column pointer n columns from the current position
#(n | numeric-variable | (expression)) – move the column pointer to row n
/ -- moves the line pointer to the first column on the next line
PAGE -- forces a page break where pointer goes to the first column of the first line of the next page
variable<: format> -- value from variable being output, with optional user format
@ | @@ -- Causes the pointer to be held for the next PUT statement

Using the syntax above is possible to take an item of information and specify it at a particular point on the page – the easiest way to think of it is to get a piece of graph paper and go to a particular square and write a character on it and continue until you either run out of characters to print or run off the end of the page.

THE ANATOMY OF THE DATA STEP TO PRODUCE THE REPORT

The data step, using a DATA _NULL_, has four main parts:

- Initialization
- Body
- Header
- Footer

Below is a basic datastep call that will produce a report:

```
data _null_;
   FILE "<filename>" N=PAGESIZE PRINT LINESLEFT=ll HEADER=hdr NOTITLES NOFOOTNOTES;
   SET <dataset> end=eof;
   put <@position> <variable> <format>;
   if eof then link ftr;
   return;
   hdr: put <@position> "<text>";
        return;
   ftr: put <#line> <@position> "<text>";
      return;
run;
```

The FILE and SET statements are the initialization step and call the name of the file to create and the data to process in the datastep.

The FILE statement opens the file to write the output to. A filename is expected and the other options used are:

N=PAGESIZE -- specifies the number of output lines that are available to the output pointer when writing output

PRINT –the form feed characters are produced in the output file at intervals determined by the value of the PAGESIZE option or the PAGESIZE system option. The N option must either be 1 or PAGESIZE, and the PRINT option must also be specified if HEADER= option is used.

LINESLEFT=var -- names a variable that contains the number of line left in a page. The PRINT option must be specified for this option to be used and the number of lines in each page is governed by the PAGESIZE option or the PAGESIZE system option.

HEADER=hdr -- names a statement label which will be linked to just before the first line of output on each page is written. This option is only valid if the PRINT option is set.

NOTITLES -- controls whether titles created by the TITLE statement are displayed in PRINT files and is only available with use of the PRINT option. The default for PRINT files is TITLES.

NOFOOTNOTES -- controls whether footnotes created by the FOOTNOTE statement are displayed in PRINT files and is only available with the use of the PRINT option. The default for PRINT files is NOFOOTNOTES but if the footnotes are need then use the FOOTNOTES option.

Although the text used in the TITLE and FOOTNOTE statements could be used, there is more control if these are in the data step as we shall see shortly.

The SET statement opens the dataset specified and the EOF option is used to signal when the datastep has finished and is used to force a call to the footnote area, label FTR, for one last time to print the footnote on the last page.

The Body section of the datastep uses the PUT statement, as discussed earlier, and displays the data to report.

The Header section of the datastep sets the header of each page and is primarily a collection of text put out by PUT statements.

Similarly the Footer section of the datastep sets the footer section of each page and is again primarily a collection of text put out by PUT statements.

OUR FIRST EXAMPLE

The output from this example is a simple one page listing of Passenger Traffic at San Francisco International Airport from 2008 to July 2011 and includes such features as a titles, centered, and footnotes:

```
01 data _airpassengers;
02    infile cards;
03    input year $ number comma10.;
04 cards;
05 2011 23,073,700
06 2010 39,253,999
07 2009 37,338,942
08 2008 37,234,592
09 ;
10 run;
11 options ls=64 ps=20;
12 data _null_;
13    call symput('ls',getoption('ls'));
14    call symput('titl1',"Passenger Traffic at San Francisco International Airport");
```

```
15    call symput('titl2',"2008-2011¹");
16 data _null_;
17    file "c:\temp\AirPassengerNumbers.txt" N=PAGESIZE PRINT
18         LINESLEFT=ll HEADER=hdr NOTITLES NOFOOTNOTES;
19    set _airpassengers end=eof;
20    put @20 year @35 number comma10.;
21    if eof then link ftr;
22    return;
23    hdr: put @((&ls-length("&titl1"))/2) "&titl1";
24         put @((&ls-length("&titl2"))/2) "&titl2";
25         put;
26         put @35 "Passenger";
27         put @20 "Year" @36 "Numbers";
28         put @20 4*"—" @35 10*"—";
29         return;
30    ftr: put #18 @1 &ls*"—";
31         put @1 "¹ January thru July numbers for 2011";
32         put @1 "Reference: Airports Council International, December 8, 2011";
33         return;
34 run;
```

The following is the output from the datastep:

```
   Passenger Traffic at San Francisco International Airport
                        2008-2011¹

                                 Passenger
                  Year            Numbers
                  ____           __________
                  2011           23,073,700
                  2010           39,253,999
                  2009           37,338,942
                  2008           37,234,592









_____________________________________________________________________
¹ January thru July numbers for 2011
Reference: Airports Council International, December 8, 2011
```

Output 1. Output from a DATA_NULL_ Statement

In the example, a lot is going on but if it is broken down the datatep is basically the same as the basic structure shown above.

Lines 1 to 10 just bring in our data.

Line 11 set the LINESIZE and PAGESIZE options for the output.

Lines 12 thru 15 set some macro variables that are going to make things a little easier later on. Line 13 gets the LINESIZE and set that value as a macro variable. Lines 14 and 15 contain the two title headers that will be used.

Lines 17 through 19 do initialization, setup the file where the output is going to be directed to and get the data.

Line 20 is the PUT statement that outputs the Body of the output file.

Line 21 just makes sure one the dataset is ended the footer section is output.

Lines 23 to 29 do the header. Something special here though – I am centering the titles so rather than trying to do some calculations of my own I am setting the starting position of the text by calculation, getting the length of the line from the LINESIZE option in the above data step.

Lines 30 to 33 present the footer. Line 30 puts out a horizontal line the length of the line of the output, using character Ox97, not the dash character.

WRAPPING TEXT ACROSS MULTIPLE LINES, ALIGNING TEXT, AND PAGE X OF Y PROCESSING

Our first example was a simple one page report but now, to quote Emeril Lagasse, “Lets kick it up a notch”. Lets consider wrapping output lines, centering and right aligning text, page breaking and “Page x of y” processing.

When programming reports it becomes clear very quickly that text will exceed the length of allowable space either through the text being longer than the linesize of the page or allowable columns available on a line. For example, you may have an Adverse Event raw text of 75 characters but your allowable column width is 45. To solve this problem in PROC REPORT you have the WRAP option, but this is not available in a DATA _NULL_ so some tool must be available to break up the line of text to output so that the first line of text is printed out on the first line and subsequent lines after it. Also, if there are multiple variables that need to be wrapped, the maximum number of lines need to be output, for example if the text for an Adverse Event is wrapped to two lines and the Action Taken text requires three lines, then the maximum lines that should be printed is three. Before we go and consider these considerations, lets first look at a macro that will “wrap” a line of text:

```
%MACRO WRAP (TXTVAR,SLPTTXT,TXTLEN,TXTINDENT,MAXLINES);

  /*TXTVAR - the variable or constant to be wrapped
    SLPTTXT - character value(s) at which text is split
    TXTLEN - width of column
    TXTINDENT - TXTINDENT n characters for second and laster lines
    MAXLINES - max number of lines to create
    NOTE: All parameters are required*/

   LENGTH _TMPTXT $32767 _&TXTVAR.1-_&TXTVAR.&MAXLINES $&TXTLEN;
   INFORMAT _&TXTVAR.1-_&TXTVAR.&MAXLINES $char&TXTLEN..;
   FORMAT _&TXTVAR.1-_&TXTVAR.&MAXLINES $char&TXTLEN..;
   ARRAY _&TXTVAR.{&MAXLINES} $&TXTLEN;
   _TMPTXT=&TXTVAR;
   ENDPOS=0;
   k=1;
   DO UNTIL (MISSING(STRIP(_TMPTXT)));
      IF K>1 THEN _TXTINDENT=input("&TXTINDENT",best.);
      else _TXTINDENT=0;
      ENDPOS=(input(&TXTLEN.,best.)-_TXTINDENT) -
             INDEXC(REVERSE(SUBSTR(_TMPTXT,1,(input(&TXTLEN.,best.)-_TXTINDENT))),&SLPTTXT);
      if (_TXTINDENT-1)>=0 then
        _&TXTVAR.{k}=repeat(' ',_TXTINDENT-1)||LEFT(SUBSTR(_TMPTXT,1,ENDPOS));
      else
        _&TXTVAR.{k}=LEFT(SUBSTR(_TMPTXT,1,ENDPOS));
      _TMPTXT=strip(SUBSTR(_TMPTXT,ENDPOS+1));
      _&TXTVAR.0=k;
      k+1;
   END;
   drop k _tmptxt endpos _TXTINDENT;
%MEND;
;
```

The WRAP macro will effectively create an array of variables at the length given by the TXTLEN parameter. The number of lines needed is in the variable _&txtvar.0. If the number of lines specified in the MAXLINES parameter is not enough a message in the LOG will say something like a subscript is out of range.

Now more on the _&txtvar.0 number. This is useful in three ways – first it shows the number of lines needed for that variable; second, when used with multiple calls to the macro on multiple variables it can be used to work out the maximum lines needed; and third, knowing the maximum number of lines needed can help to determine if a page break is needed before the data is printed. Lets look a snippet of code:

```
45 %wrap(AETERM," ",30,2,10);
46 %wrap(AEOUT," ",15,0,10);
47 _maxlines=max(1,_aeterm0,_aeout0);
48 if (linesleft-_maxlines)<5 then do;
49    link ftr;
50    put _page_;
51 end;
;
```

In lines 45 and 46 the WRAP macro is called, and the maximum number of lines needed is calculated at line 47 -- if the AETERM variable needs two lines and the AEOUT variable required three lines, then the maximum number of lines needed is three. At line 48, this information is used to work out if the entire record will print out on the page -- if not then the footnote is called and a page break set before the record is printed (LINESLEFT value comes from the LINESLEFT option in the FILE statement). All of this will become clearer in our next example.

To left or right align a text variable the LEFT and RIGHT text functions can be used, but what about centering? Below is a macro that will do centering, left and right text alignment:

```
%macro aligntxt(var,len,align,lenB);
   %if (&lenB eq ) %then %let lenB=&len;;
   informat &var $char&lenB..;
   format &var $char&lenB..;
   if lengthn(&var)>&len then
     put 'WAR' "NING: Length of text in variable &var > &len, " &var=;
   else if ^missing(&var) then do;
      %if %upcase(&align)=L %then %do;
         &var=strip(&var);
      %end;
      %else %if %upcase(&align)=C %then %do;
         if &len=length(&var) then &var=strip(&var);
         else &var=repeat(' ',floor(((&len-length(&var))/2)-1))||strip(&var);
      %end;
      %else %if %upcase(&align)=R %then %do;
         if &len=length(&var) then &var=strip(&var);
         else &var=repeat(' ',&len-length(&var)-1)||strip(&var);
      %end;
   end;
%mend aligntxt;
```

Placing a placeholder “Page x of y” in the document can be done in any part of the document, whether it be in the header or footer or in the body of the page. However, doing the necessary counts and replacing the “Page x of y” text with actual page numbers must be done in post processing, i.e. after the report is written in the datastep. A macro to do this is given below:

```
%macro SetPage(outrpt);
   data _null_;
      retain _pagenum 0;
      infile "&outrpt" length=len end=eof lrecl=200;
      input _txt $varying200. len;
      if index(_txt,'Page x of y') then _pagenum+1;
      if eof then call symput('pgnum',compress(put(_pagenum,8.)));
   data _x;
      attrib _txt length=$200 format=$char200.
             _txt2 length=$200 format=$char200.;
      infile "&outrpt" length=len SHAREBUFFERS;
      file "&outrpt" lrecl=200;
      input _txt $varying200. len;
      if index(_txt,'Page xxxx of yyyy') then do;
         _pagenum+1;
         _txt2='Page '||compress(put(_pagenum,4.))||" of &pgnum";
         _txt2l=length(strip(_txt2));
         _txt2=repeat(' ',17-_txt2l-1)||strip(_txt2);
         _txt=tranwrd(_txt,'Page xxxx of yyyy',_txt2);
      end;
   run;
%mend SetPage;
;
```

Note that the macro actually uses the text “Page xxxx of yyyy” rather than “Page x of y” – the reason for this is that it is easier to replace a longer text with a smaller text. The macro has three steps. First, the number of pages, i.e. instances of “Page xxxx of yyyy” appear, are counted and put in a macro variable called PGNUM. The second step reads in the output file and replaces the text “Page xxxx of yyyy”, updating the original file – all of this is done is one step with the SHAREBUFFERS option in the INPUT statement.

Now for an example, bringing this all together. Here we have some Adverse Event Data from a fictitious study involving a new sauce called Sweet Sauce II (the last study, Sweet Sauce I, was a dismal failure):

```
data _ae0;
   attrib PTNO length=$4
          AETERM length=$100
          AESER length=$4
          AESTDT length=$10
          AEENDT length=$10
          AEACN length=$25
          AEREL length=$25
          AETXGR length=$25
          AEOUT length=$25;
   infile cards dlm="~";
   input PTNO $ AETERM $ AESER $ AESTDT $ AEENDT $ AEACN $ AEREL $ AETXGR $ AEOUT $;
   *PTNO~AETERM~AESER~AESTDT~AEENDT~AEACN~AEREL~AETXGR~AEOUT;
cards;
0001~HEARTBURN~NO~2009-02-21~2009-12-09~NO ACTION~NOT RELATED~MODERATE~RESOLVED
0001~UPSET STOMACH~NO~2010-04-20~2011-04-09~NO ACTION~POSSIBLE~MILD~RESOLVED
0001~NAUSEA~NO~2009-04-26~2009-05-17~NO ACTION~NOT RELATED~MILD~RESOLVED
0001~NAUSEA~NO~2009-10-30~2010-08-28~NO ACTION~NOT RELATED~MILD~RESOLVED
0001~BLACK AND BLOODY STOOLS~YES~2010-04-12~2010-04-15~TEMPORARILY
DISCONTINUED~POSSIBLE~SEVERE~RESOLVED
0002~HEARTBURN~NO~2010-05-25~2010-10-09~NO ACTION~NOT RELATED~MILD~RESOLVED
0002~FLATULENCE~NO~2009-06-10~2010-06-10~NO ACTION~PROBABLE~MILD~ONGOING
0002~BELCHING~NO~2010-02-23~2010-02-24~NO ACTION~NOT RELATED~MILD~RESOLVED
0002~UPSET STOMACH~NO~2008-06-22~2009-04-18~NO ACTION~NOT RELATED~MODERATE~RESOLVED
0003~NAUSEA~NO~2008-12-19~2009-07-18~NO ACTION~NOT RELATED~MILD~RESOLVED
0003~SWELLING OF THE MOUTH~YES~2009-05-23~2010-03-16~TEMPORARILY DISCONTINUED~NOT
RELATED~SEVERE~RESOLVED
0004~ITCHING~NO~2009-11-26~2009-12-10~NO ACTION~NOT RELATED~MILD~ONGOING
0005~HEARTBURN~NO~2009-12-12~2010-05-06~NO ACTION~NOT RELATED~MILD~RESOLVED
0005~DROWSINESS~NO~2009-12-28~2010-10-08~NO ACTION~NOT RELATED~MILD~RESOLVED
0005~HEARTBURN~NO~2009-04-09~2009-04-28~NO ACTION~NOT RELATED~MILD~RESOLVED
0005~FATIGUE~NO~2008-06-26~2009-01-25~NO ACTION~PROBABLE~MODERATE~RESOLVED
0005~DROWSINESS~NO~2010-01-04~2010-12-01~NO ACTION~NOT RELATED~MILD~ONGOING
0005~RASH~NO~2009-04-22~2009-10-06~NO ACTION~NOT RELATED~MILD~RESOLVED
0005~HIVES~NO~2009-08-24~2009-12-09~NO ACTION~NOT RELATED~MILD~ONGOING
0005~NAUSEA~NO~2011-04-07~2011-11-20~NO ACTION~NOT RELATED~MILD~RESOLVED
0006~HEARTBURN~NO~2009-03-06~2009-07-17~NO ACTION~NOT RELATED~MILD~RESOLVED
0006~ITCHING~NO~2010-04-26~2010-07-13~NO ACTION~NOT RELATED~MILD~ONGOING
0006~BIRD BITE~NO~2010-05-29~2010-07-01~NO ACTION~NOT RELATED~MILD~RESOLVED
0007~HEARTBURN~NO~2009-02-24~2010-01-24~NO ACTION~NOT RELATED~MODERATE~ONGOING
0007~NAUSEA~NO~2010-08-21~2011-06-11~NO ACTION~NOT RELATED~MILD~RESOLVED
0007~DIZZINESS~NO~2008-11-15~2009-09-29~NO ACTION~NOT RELATED~MILD~RESOLVED
0008~SWELLING OF THE FACE~NO~2009-03-26~2009-10-30~NO ACTION~NOT RELATED~MILD~RESOLVED
0008~BELCHING~NO~2008-12-11~2009-02-07~NO ACTION~POSSIBLE~MILD~RESOLVED
0008~NAUSEA~NO~2008-10-12~2008-11-16~NO ACTION~NOT RELATED~MILD~ONGOING
0009~FATIGUE~NO~2009-05-22~2009-11-01~NO ACTION~PROBABLE~MODERATE~RESOLVED
0009~RINGING IN THE EARS~NO~2008-11-29~2009-07-20~NO ACTION~NOT RELATED~MILD~ONGOING
0010~DIARRHEA~NO~2009-01-03~2009-04-27~NO ACTION~POSSIBLE~MILD~RESOLVED
0010~NAUSEA~NO~2010-06-01~2011-01-23~NO ACTION~NOT RELATED~SEVERE~RESOLVED WITH SEQUELAE
0010~HEARING LOSS~NO~2009-06-29~2009-08-23~NO ACTION~NOT RELATED~MILD~ONGOING
0011~HEARTBURN~NO~2009-02-01~2009-03-02~NO ACTION~POSSIBLE~MILD~RESOLVED
0011~RASH~NO~2009-12-19~2010-12-03~NO ACTION~POSSIBLE~MILD~ONGOING
0011~HIVES~NO~2011-05-17~2011-07-05~NO ACTION~POSSIBLE~MILD~ONGOING
0011~DIARRHEA~NO~2009-07-14~2009-07-31~NO ACTION~NOT RELATED~MILD~RESOLVED
0011~SEVERE AND PERSISTENT STOMACH PAIN~YES~2009-01-02~2009-06-06~TEMPORARILY DISCONTINUED~NOT
RELATED~SEVERE~RESOLVED
0011~NAUSEA~NO~2009-01-28~2009-07-01~NO ACTION~NOT RELATED~MODERATE~RESOLVED
0011~HEARTBURN~NO~2009-07-22~2010-05-07~NO ACTION~POSSIBLE~MILD~RESOLVED
0011~RASH~NO~2010-07-13~2010-09-24~NO ACTION~NOT RELATED~MILD~ONGOING
0011~ITCHING~NO~2009-08-10~2009-12-13~NO ACTION~NOT RELATED~MILD~RESOLVED
0011~HIVES~NO~2009-09-01~2010-03-16~NO ACTION~NOT RELATED~MILD~ONGOING
0011~PAIN IN TOE~NO~2009-03-17~2009-08-24~NO ACTION~NOT RELATED~MILD~RESOLVED
;
proc sort data=_ae0;
   by ptno aestdt aeendt;
run;
;
```

Before we can go any further, we have to design the layout, where exactly the variables are going to be output and plan the header, using a linesize of 134 characters wide and pagesize of 58.

As part of this step we need to work out whether the amount of space that we have allocated for each variable will actually fix the data – all our data is character so it is possible to use a small piece of code to find out the maximum size of each variable:

```
141  data _null_;
142     array len {9} len1-len9;
143     retain len1-len9;  *9 variables in dataset;
144     set _ae0 end=eof;
145     array ary {*} _character_;
146     do i=1 to dim(ary);
147        len(i)=max(len(i),lengthn(ary{i}));
148     end;
149     if eof then do;
150        do i=1 to dim(ary);
151           vname=vname(ary(i)); *Get variable name in array;
152           put @1 vname= len(i)= ;
153        end;
154     end;
155  run;

vname=PTNO len1=4
vname=AETERM len2=34
vname=AESER len3=3
vname=AESTDT len4=10
vname=AEENDT len5=10
vname=AEACN len6=24
vname=AEREL len7=11
vname=AETXGR len8=8
vname=AEOUT len9=22
```

Using this information we can create a design given below:

```
Yum, Yum and Yum, Inc.                                                                                    Page x of y
Protocol: Sweet Sauce II

                                                  Listing 8
                                              Adverse Events

_____________________________________________________________________________________________________________________
Subject                                Start                                    Related to
Number  Adverse Event                  Date     Stop Date  Serious?  Action Taken  Study Drug?  Toxicity      Outcome
_____________________________________________________________________________________________________________________

 cccc   ccccccccccccccccccccccc  ddmmmyyyy  ddmmmyyyy    ccc     ccccccccccc   ccccccccccc  ccccccccccccc cccccccccccccccccccc

_____________________________________________________________________________________________________________________
Program Name: AELIST.SAS   Date/Time of Creation: ddmmmyyyy hh:mm
```

Variables AESTDT and AEENDT are 10 characters in length but the program will read these dates in as *yyyy-mm-dd* format and output as *ddmmmyyyy* format so the length used will be 9. PTNO, AESER, AEREL, AETXGR and AEOUT variables will fit in the positions allowed but variables AETERM and AEACN will not and these need to be wrapped.

Lets now look at some code and output:

```
01 %let ls=134; %let ps=58;
02 options ls=&ls ps=&ps; *Set our output pagesize;
03 data _null_;
04    retain np 0; *New page flag;
05    file "c:\temp\aelist.txt" N=PAGESIZE print linesleft=ll header=hdr notitles;
06    set _ae0 end=eof;
07    by ptno; *Going to have a blank line between each PTNO value;
08
09    length _aestdtn _aestdtn 8;
10    format _aestdtn _aeendtn date9.;
11    _aestdtn=input(aestdt,yymmdd10.);
12    _aeendtn=input(aeendt,yymmdd10.); *Convert Start and Stop Dates to numeric;
13
14    %WRAP(aeterm," ",26,2,3); *Wrap AETERM to 26 chars, indent 2nd and later to 2 chars;
15    %WRAP(aeacn," ",12,0,3); *Wrap AEACN to 12 character, no indent;
16    maxlines=sum(0,_aeterm0,_aeacn0); *Maximum number of lines needed to print record;
17
18    if (ll-maxlines)<5 then do;
19       link ftr;
20       put _page_;
21    end;
```

```
22
23    if first.ptno or np=1 then do;
24       put; *Put our line between each subject;
25       put @2 ptno $4. @@;
26       np=0;
27    end;
28
29    *if first presence on page for subject then put Subject Number and hold pointer;
30    put @9 _aeterm1 $char26. @37 _aestdtn @48 _aeendtn @61 aeser @69 _aeacn1 @83 aerel
31        @96 aetxgr @113 aeout;
32
33    *If data present in variables created in the WRAP macro, then output these;
34    if ^missing(_aeterm2) or ^missing(_aeacn2) then put @9 _aeterm2 $char26. @69 _aeacn2;
35    if ^missing(_aeterm3) or ^missing(_aeacn3) then put @9 _aeterm3 $char26. @69 _aeacn3;
36
37    if eof then link ftr; *If the end of the dataset, make sure footer gets printed;
38
39    return;
40    hdr:  put @1 'Yum, Yum and Yum, Inc.' @(&ls-length("Page x of y")+1) "Page x of y";
41          put @1 'Protocol: Sweet Sauce II';
42          put @((&ls-length("Listing 8"))/2) "Listing 8";
43          put @((&ls-length("Adverse Events"))/2) "Adverse Events";
44          put;
45          put @1 &ls*'—';
46          put @1 'Subject' @39 'Start' @83 'Related to';
47          put @1 'Number' @9 'Adverse Event' @39 'Date' @48 'Stop Date'
48              @59 'Serious?' @69 'Action Taken' @83 'Study Drug?' @96 'Toxicity'
49              @113 'Outcome';
50          put @1 &ls*'—';
51          return;
52    ftr:  put #(&ps-1) @1 &ls*'—';
53          put "Program Name: AELIST.SAS  Date/Time of Creation: &sysdate9. &systime.";
54          np=1;
55          return;
56 run;
```

The output from the datastep is shown below:

The following is the output from the datastep:

Yum, Yum and Yum, Inc. Page xxxx of yyyy
Protocol: Sweet Sauce II

Listing 8
Adverse Events

Subject Number	Adverse Event	Start Date	Stop Date	Serious?	Action Taken	Related to Study Drug?	Toxicity	Outcome
0001	HEARTBURN	21FEB2009	09DEC2009	NO	NO ACTION	NOT RELATED	MODERATE	RESOLVED
	NAUSEA	26APR2009	17MAY2009	NO	NO ACTION	NOT RELATED	MILD	RESOLVED
	NAUSEA	30OCT2009	28AUG2010	NO	NO ACTION	NOT RELATED	MILD	RESOLVED
	BLACK AND BLOODY STOOLS	12APR2010	15APR2010	YES	TEMPORARILY DISCONTINUED	POSSIBLE	SEVERE	RESOLVED
	UPSET STOMACH	20APR2010	09APR2011	NO	NO ACTION	POSSIBLE	MILD	RESOLVED
0002	UPSET STOMACH	22JUN2008	18APR2009	NO	NO ACTION	NOT RELATED	MODERATE	RESOLVED
	FLATULENCE	10JUN2009	10JUN2010	NO	NO ACTION	PROBABLE	MILD	ONGOING
	BELCHING	23FEB2010	24FEB2010	NO	NO ACTION	NOT RELATED	MILD	RESOLVED
	HEARTBURN	25MAY2010	09OCT2010	NO	NO ACTION	NOT RELATED	MILD	RESOLVED
0003	NAUSEA	19DEC2008	18JUL2009	NO	NO ACTION	NOT RELATED	MILD	RESOLVED
	SWELLING OF THE MOUTH	23MAY2009	16MAR2010	YES	TEMPORARILY DISCONTINUED	NOT RELATED	SEVERE	RESOLVED
0004	ITCHING	26NOV2009	10DEC2009	NO	NO ACTION	NOT RELATED	MILD	ONGOING
0005	FATIGUE	26JUN2008	25JAN2009	NO	NO ACTION	PROBABLE	MODERATE	RESOLVED
	HEARTBURN	09APR2009	28APR2009	NO	NO ACTION	NOT RELATED	MILD	RESOLVED
	RASH	22APR2009	06OCT2009	NO	NO ACTION	NOT RELATED	MILD	RESOLVED
	HIVES	24AUG2009	09DEC2009	NO	NO ACTION	NOT RELATED	MILD	ONGOING
	HEARTBURN	12DEC2009	06MAY2010	NO	NO ACTION	NOT RELATED	MILD	RESOLVED
	DROWSINESS	28DEC2009	08OCT2010	NO	NO ACTION	NOT RELATED	MILD	RESOLVED
	DROWSINESS	04JAN2010	01DEC2010	NO	NO ACTION	NOT RELATED	MILD	ONGOING
	NAUSEA	07APR2011	20NOV2011	NO	NO ACTION	NOT RELATED	MILD	RESOLVED
0006	HEARTBURN	06MAR2009	17JUL2009	NO	NO ACTION	NOT RELATED	MILD	RESOLVED
	ITCHING	26APR2010	13JUL2010	NO	NO ACTION	NOT RELATED	MILD	ONGOING
	BIRD BITE	29MAY2010	01JUL2010	NO	NO ACTION	NOT RELATED	MILD	RESOLVED
0007	DIZZINESS	15NOV2008	29SEP2009	NO	NO ACTION	NOT RELATED	MILD	RESOLVED
	HEARTBURN	24FEB2009	24JAN2010	NO	NO ACTION	NOT RELATED	MODERATE	ONGOING
	NAUSEA	21AUG2010	11JUN2011	NO	NO ACTION	NOT RELATED	MILD	RESOLVED
0008	NAUSEA	12OCT2008	16NOV2008	NO	NO ACTION	NOT RELATED	MILD	ONGOING
	BELCHING	11DEC2008	07FEB2009	NO	NO ACTION	POSSIBLE	MILD	RESOLVED
	SWELLING OF THE FACE	26MAR2009	30OCT2009	NO	NO ACTION	NOT RELATED	MILD	RESOLVED
0009	RINGING IN THE EARS	29NOV2008	20JUL2009	NO	NO ACTION	NOT RELATED	MILD	ONGOING
	FATIGUE	22MAY2009	01NOV2009	NO	NO ACTION	PROBABLE	MODERATE	RESOLVED
0010	DIARRHEA	03JAN2009	27APR2009	NO	NO ACTION	POSSIBLE	MILD	RESOLVED

Program Name: AELIST.SAS Date/Time of Creation: 22DEC2011 07:38

Output 2. Output of First Page from Adverse Events Using a DATA_NULL_ Statement

The following is the output from the datastep:

Yum, Yum and Yum, Inc. Page xxxx of yyyy
Protocol: Sweet Sauce II

Listing 8
Adverse Events

Subject Number	Adverse Event	Start Date	Stop Date	Serious?	Action Taken	Related to Study Drug?	Toxicity	Outcome
0010	HEARING LOSS	29JUN2009	23AUG2009	NO	NO ACTION	NOT RELATED	MILD	ONGOING
	NAUSEA	01JUN2010	23JAN2011	NO	NO ACTION	NOT RELATED	SEVERE	RESOLVED WITH SEQUELAE
0011	SEVERE AND PERSISTENT STOMACH PAIN	02JAN2009	06JUN2009	YES	TEMPORARILY DISCONTINUED	NOT RELATED	SEVERE	RESOLVED
	NAUSEA	28JAN2009	01JUL2009	NO	NO ACTION	NOT RELATED	MODERATE	RESOLVED
	HEARTBURN	01FEB2009	02MAR2009	NO	NO ACTION	POSSIBLE	MILD	RESOLVED
	PAIN IN TOE	17MAR2009	24AUG2009	NO	NO ACTION	NOT RELATED	MILD	RESOLVED
	DIARRHEA	14JUL2009	31JUL2009	NO	NO ACTION	NOT RELATED	MILD	RESOLVED
	HEARTBURN	22JUL2009	07MAY2010	NO	NO ACTION	POSSIBLE	MILD	RESOLVED
	ITCHING	10AUG2009	13DEC2009	NO	NO ACTION	NOT RELATED	MILD	RESOLVED
	HIVES	01SEP2009	16MAR2010	NO	NO ACTION	NOT RELATED	MILD	ONGOING
	RASH	19DEC2009	03DEC2010	NO	NO ACTION	POSSIBLE	MILD	ONGOING
	RASH	13JUL2010	24SEP2010	NO	NO ACTION	NOT RELATED	MILD	ONGOING
	HIVES	17MAY2011	05JUL2011	NO	NO ACTION	POSSIBLE	MILD	ONGOING

Program Name: AELIST.SAS Date/Time of Creation: 22DEC2011 07:38

Output 3. Output of Second Page from Adverse Events Using a DATA_NULL_ Statement

Looking at this code, most of it is what we saw in the previous example, just with a few more tricks that are useful.

At lines 14-16, AETERM and AEACN text lines are wrapped and the maximum number of lines needed to print the record are calculated, stored in variable maxlines (variables _aeterm0 and _aeacn0 are derived in the %WRAP macro, and contain the number of lines needed to print the variable – refer to discussion regarding the %WRAP macro above).

Lines 18-21 check if the record can fix on the page, i.e. all lines of the text can be printed – if not the footer is printed and a page break made.

Lines 23-27 put out the Subject Number for the first record for a subject, or if a new page has been started – this is flagged in the FTR section of the code by the variable NP with a value of 1, and in effect 'carries over' the subject number for page to page.

Lines 34 and 35 print out any text left in the wrapped variables, with line 37 forcing a footer if the last record has been printed.

To get the page numbers, the macro SETPAGE is run with the following code:

```
%SetPage(%str(c:\temp\aelist.txt));
```

and what appears in the updated output file is (below is just the first few lines of the output file):

The following is the output from the datastep:

```
Yum, Yum and Yum, Inc.                                                                                                  Page 1 of 2
Protocol: Sweet Sauce II
                                                        Listing 8
                                                   Adverse Events

_____________________________________________________________________________________________________________________________

Subject                                Start                                                Related to
Number  Adverse Event                   Date      Stop Date  Serious?  Action Taken        Study Drug?  Toxicity         Outcome
_____________________________________________________________________________________________________________________________

 0001   HEARTBURN                    21FEB2009  09DEC2009    NO      NO ACTION           NOT RELATED  MODERATE         RESOLVED
        NAUSEA                       26APR2009  17MAY2009    NO      NO ACTION           NOT RELATED  MILD             RESOLVED
```

Output 4. Output of Adverse Events Using a DATA_NULL_ Statement After SETPAGE Macro

Something PROC REPORT cannot do, without manipulation, is carry over the subject number, but this is done easily in the data step.

To quoting Emeril Lagasse again, “Lets kick it up another notch”, this time looking at a case where PROC REPORT cannot do the output but a DATA _NULL_ can.

We are using the same dataset as above but there are comments data with two adverse events, as given below:

```
0002~UPSET STOMACH~SUBJECT ATE 6KG OF CHOCOLATE AFTER BREAKUP WITH PARTNER
0006~BIRD BITE~BIRD WAS A PET OF THE SUBJECT. BIRD WELL CARED FOR AND IT IS BELIEVED THAT
SHOTS FOR BIRD WERE UP TO DATE SO NO EXAMINATION NEEDED BY A VET TO SEE IF BIRD WAS ILL.
```

It is easy enough to adapt the output in PROC REPORT so a column can be added to present the comment, but what if it was better to have that comment out as a separate line after the adverse event, i.e. a layout of:

```
Yum, Yum and Yum, Inc.                                                                                                  Page x of y
Protocol: Sweet Sauce II
                                                        Listing 8
                                                   Adverse Events

_____________________________________________________________________________________________________________________________

Subject                                Start                                                Related to
Number  Adverse Event                   Date      Stop Date  Serious?  Action Taken        Study Drug?  Toxicity         Outcome
_____________________________________________________________________________________________________________________________

 cccc   ccccccccccccccccccccccccc  ddmmmyyyy  ddmmmyyyy    ccc     ccccccccccc  ccccccccccc  ccccccccccccccc  ccccccccccccccccc
          Comment: ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
_____________________________________________________________________________________________________________________________

Program Name: AELIST.SAS   Date/Time of Creation: ddmmmyyyy hh:mm
```

This type of output could be done my editing/modifying the output file that PROC REPORT could produce, but this approach is beyond this paper. Instead, the data step that creates the output file will be modified, assuming that the comment data has been merged into the AE dataset we are referencing, using the code snippet below (the additional code needed is underlined):

```
%WRAP(aeterm," ",26,2,3); *Wrap AETERM to 34 chars, indent 2nd and later lines to 2 chars;
%WRAP(aeacn," ",12,0,3); *Wrap AEACN to 12 character, no indent;
%WRAP(comment," ",112,0,2); *Wrap COMMENT to 112 chars, no indent;
maxlines=sum(0,_aeterm0,_aeacn0); *Maximum number of lines needed to print record;
if ^missing(comment) then maxlines=sum(maxlines, comment0);
  *If comment then add this to the number of lines needed to print the comment field;
if (ll-maxlines)<5 then do;
   link ftr;
   put _page_;
end;

if first.ptno or np=1 then do;
   put; *Put our line between each subject;
   put @2 ptno $4. @@;
   np=0;
end;

*if first presence on page for subject then put Subject Number and hold pointer;
put @9 _aeterm1 $char26. @37 _aestdtn @48 _aeendtn @61 aeser @69 _aeacn1 @83 aerel
    @96 aetxgr @113 aeout;

*If data present in variables created in the WRAP macro, then output these;
if ^missing(_aeterm2) or ^missing(_aeacn2) then put @9 _aeterm2 $char26. @69 _aeacn2;
if ^missing(_aeterm3) or ^missing(_aeacn3) then put @9 _aeterm3 $char26. @69 _aeacn3;

if ^missing(comment) then do;
   put @14 'Comment:' @23  comment1 $char112.;
```

```
   if ^missing( comment2) then put @23  comment2 $char112.;
end;
```

with the following output in the output file:

Using this macro the RTF file that is created looks like

```
0002   UPSET STOMACH              22JUN2008  18APR2009    NO      NO ACTION     NOT RELATED  MODERATE         RESOLVED
         Comment: SUBJECT ATE 6KG OF CHOCOLATE AFTER BREAKUP WITH PARTNER
       FLATULENCE                 10JUN2009  10JUN2010    NO      NO ACTION     PROBABLE     MILD             ONGOING
       BELCHING                   23FEB2010  24FEB2010    NO      NO ACTION     NOT RELATED  MILD             RESOLVED
       HEARTBURN                  25MAY2010  09OCT2010    NO      NO ACTION     NOT RELATED  MILD             RESOLVED

0003   NAUSEA                     19DEC2008  18JUL2009    NO      NO ACTION     NOT RELATED  MILD             RESOLVED
       SWELLING OF THE MOUTH      23MAY2009  16MAR2010    YES     TEMPORARILY   NOT RELATED  SEVERE           RESOLVED
                                                                  DISCONTINUED

0004   ITCHING                    26NOV2009  10DEC2009    NO      NO ACTION     NOT RELATED  MILD             ONGOING

0005   FATIGUE                    26JUN2008  25JAN2009    NO      NO ACTION     PROBABLE     MODERATE         RESOLVED
       HEARTBURN                  09APR2009  28APR2009    NO      NO ACTION     NOT RELATED  MILD             RESOLVED
       RASH                       22APR2009  06OCT2009    NO      NO ACTION     NOT RELATED  MILD             RESOLVED
       HIVES                      24AUG2009  09DEC2009    NO      NO ACTION     NOT RELATED  MILD             ONGOING
       HEARTBURN                  12DEC2009  06MAY2010    NO      NO ACTION     NOT RELATED  MILD             RESOLVED
       DROWSINESS                 28DEC2009  08OCT2010    NO      NO ACTION     NOT RELATED  MILD             RESOLVED
       DROWSINESS                 04JAN2010  01DEC2010    NO      NO ACTION     NOT RELATED  MILD             ONGOING
       NAUSEA                     07APR2011  20NOV2011    NO      NO ACTION     NOT RELATED  MILD             RESOLVED

0006   HEARTBURN                  06MAR2009  17JUL2009    NO      NO ACTION     NOT RELATED  MILD             RESOLVED
       ITCHING                    26APR2010  13JUL2010    NO      NO ACTION     NOT RELATED  MILD             ONGOING
       BIRD BITE                  29MAY2010  01JUL2010    NO      NO ACTION     NOT RELATED  MILD             RESOLVED
         Comment: BIRD WAS A PET OF THE SUBJECT. BIRD WELL CARED FOR AND IT IS BELIEVED THAT SHOTS FOR BIRD WERE UP TO
                  DATE SO NO EXAMINATION NEEDED BY A VET TO SEE IF BIRD WAS ILL.

0007   DIZZINESS                  15NOV2008  29SEP2009    NO      NO ACTION     NOT RELATED  MILD             RESOLVED
```

Output 5. Output After Code to put Comment Text out in Output File

but the advantage is that it can be opened in word processors like Microsoft Word easily.

REVISITING THE TXT2RTF MACRO

At PharmaSUG 2010, I presented a macro that took a text file and made it into a RTF file. The macro presented would take a text file, as generated above, and create the RTF file for it that could be opened in Microsoft Word or similar programs.

Since its presentation, there has been a modification to the macro that solves a small issue with the '_' (underline) character – in some versions of Word the character would not be seen in Microsoft Word although it was there. This, unfortunately, is a known issue of these programs and is caused by the resolution of screen vs. the height needed to display the character.

In order to resolve the issue, the original macro was modified and is now presented again (modifications are in BOLD and UNDERLINED):

```
%macro txt2rtf(_txtfn= /*Text file to convert to RTF*/
               ,_rtffn= /*RTF file that is to be output to*/
               ,_pgwh=15840 /*Page width in twips*/
               ,_pght=12240 /*Page height in twips*/
               ,_mgns=1440 /*Margin width in twips*/
               ,_lspe=%str(\landscape) /*If landscape, need to add this value*/
               ,_fnsz=8 /*Font point size - integer*/
               , cndns=Y /*Condensed Y|N - text height is same as spacing*/
               );
   data _null_;
      infile "&_txtfn" end=eof;
      length _txt $200;
      input;
      file "&_rtffn";
      if _n_=1 then do;
         put '{\rtf1\ansi\ansicpg1252\deff0\deflang1033' /
            '{\fonttbl{\f0\fmodern\fprq1\fcharset0 Courier New;}}' /
            '{\colortbl \red0\green0\blue0;}' /
             "\paperw&_pgwh.\paperh&_pght.\margl&_mgns.\margr&_mgns." /
             "\margt&_mgns.\margb&_mgns.&_lspe." /
             "\viewkind4\uc1\pard\ql\fi0\li0\ri0\sb0\sa0" @@;
             %if %upcase(&_cndns)=Y %then put "\sl-%eval(&_fnsz*20)";
```

```
23              %else put;;
24              put "\cf0\f0\fs%eval(&_fnsz*2) " _infile_;
25       end;
26       else do;
27          if substr(_infile_,1,1)=byte(12) then do;
28             _txt='\page '||substr(_infile_,2);
29             put _txt;
30          end;
31          else if ^eof then put '\par ' _infile_;
32          else if eof then put '\par ' _infile_ '}';
33       end;
34    run;
35 %mend;
```

By adding the _CNDNS option you can vary the number of lines presented on the page as the following table details:

Paper Size	Font Size	PAGESIZE _cndns=Y	PAGESIZE _cndns=N	LINESIZE
LETTER	8	58	51	134
LETTER	9	52	45	119
LETTER	10	46	41	104
A4	8	56	49	145
A4	9	50	44	129
A4	10	45	39	116

It must be noted that these pagesize and linesize settings are a guide only and does depend on your own word processor.

In our example above, the pagesize and linesize settings were set to LETTER size, with an 8pt font and spacing being condensed, so the macro call would be:

```
%txt2rtf(_txtfn=%str(c:\temp\AELISTING.TXT) /*Text file to convert to RTF*/
        ,_rtffn=%str(c:\temp\AELISTING.RTF) /*RTF file that is to be output to*/
        ,_lspe=%str(\landscape) /*If landscape, need to add this value*/
        ,_fnsz=8 /*Font point size - integer*/
        ,_cndns=Y /*Condensed Y|N - text height is same as spacing*/
        );
```

Using this macro the RTF file that is created looks like

Yum, Yum and Yum, Inc. Page 1 of 2
Protocol: Sweet Sauce II

Listing 8
Adverse Events

Subject Number	Adverse Event	Start Date	Stop Date	Serious?	Action Taken	Related to Study Drug?	Toxicity	Outcome
0001	HEARTBURN	21FEB2009	09DEC2009	NO	NO ACTION	NOT RELATED	MODERATE	RESOLVED
	NAUSEA	26APR2009	17MAY2009	NO	NO ACTION	NOT RELATED	MILD	RESOLVED
	NAUSEA	30OCT2009	28AUG2010	NO	NO ACTION	NOT RELATED	MILD	RESOLVED

Output 6. Output after TXT2RTF Macro

but the advantage is that it can be opened in word processors like Microsoft Word easily.

For further details on the TXT2RTF macro, please refer to the paper entitled Making an RTF file Out of a Text File, With SAS (CC13), presented at PharmaSUG 2010.

CONCLUSION

In this paper the, the DATA _NULL_ has been rediscovered and has shown how to create three outputs – the first a simple one that was only one page with no wrapping of text, a second output that was multipage where we have to wrap text, align text, break records so that they would not be split over pages, and finally an example where comments were added to that example and placed underneath the record. Also re-presented was a modified macro that took a text file and put it into a RTF format ready for opening with Microsoft Word or similar.

DATA _NULL_ is still a strong and valid option for creating complex outputs that the likes of PROC REPORT cannot present.

REFERENCES

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

PharmaSUG 2010. Making an RTF file Out of a Text File, With SAS. Paper CC13. David Franklin

CONTACT INFORMATION

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

Author Name:	David Franklin
Enterprise:	TheProgrammersCabin.com
Address:	16 Roberts Rd
City, State ZIP:	Litchfield, NH 03052
Work Phone:	603-275-6809
Email:	dfranklin@TheProgrammersCabin.com
Web:	TheProgrammersCabin.com