Using Macros as a Data Table Generation Tool

Daphne E. Ewing, Bio-Pharm Clinical Services, Blue Bell, PA

Abstract

Displaying a variety of data in a consistent manner can be tedious and frustrating. Making adjustments to these displays, such as adding/removing columns, can be time consuming and just as frustrating.

This paper will show the user how to set up macros which work together to make data table generation consistent, easy to follow, and easy to modify. I will define several SAS macros, show how they interact with each other and the data, and give examples of input and output. Given these macros, I will also show how to modify displays easily.

Introduction

When preparing clinical data listings for the FDA, or other review boards, accuracy and consistency are essential. Implementing consistency can be a challenge, especially when there are multiple people working on the same project. This paper will show how to use several macros in a DATA_NULL step that work together to gain more consistent output. These macros also make changing data listings easy and fast.

Review the Requirements

The first step is to have a layout of what data is to be presented for all data listings. These layouts can then be reviewed to identify the portions of the displays that do not change and those that do change. Usually, the heading at the top of each page will not change. One other portion that may be consistent across listings is the patient identifying information and the order in which it is displayed. The portions of the displays that change are usually the different data items on each display (i.e. Demography fields versus Adverse Event fields). The remainder of this paper will show a breakdown of the different portions of the data listing and the macros associated with them.

The Heading (HDG MACRO)

The heading of data listings can be comprised of items such as the name of the company, the protocol, the condition, the page number, the system date and time, the type of report (clinical or statistical), and possibly the appendix number. To generate the heading for all listings, a heading macro can be written to keep that part of the display consistent. Below is an example of a heading macro.

```
%macro hdg(id, page, pub, title="", fnct="", line="");
put &id, 'Pharmaceutical Company';
if "PAGE" ne ""
  with systval(61)-25 'PAGE ' &page;
/ &id, 'Protocol No. ABC-123'
/ &id, 'Clinical/Statistical Report'
/ &id, 'SAS :SOFTWARE: SYSTEM';
if "PAGE" ne ""
  with &line + 3 &str(1);
end;

%end;
```

This HDG macro is comprised of the PUT statements necessary to display the heading information for all tables. In the example above, the name of the company and page number appear on the first line. There is reference here to the LS macro variable. This is a global macro variable which is set prior to the DATA_NULL which defines the number of spaces to a line, or linesize. The protocol appears on the second line, and on the third line is the type of the report and the program name that created this report (APGM), the system date and the system time. The two lines incrementing the FNCT macro variable are used to account for the number of lines necessary to display the default lines from the FOOT macro (which will be described in more detail later). The FNCT variable is used and incremented to handle the number of lines required of the footnote lines on each listing. Since, as shown below, the FOOG macro automatically displays a solid line, a blank space, and one line of text, this HDG macro adds 3 to the FNCT macro variable to handle the FOOT macro. This makes it possible to determine the number of actual data lines which can still be displayed on the table. The last portion of this macro is the displaying of the appendix number which is described in more detail below.

Since appendix numbers change fairly frequently, it is easiest to change them in one place, rather than editing every single program where there is an appendix number change. Every program for a given project creates a different display or group of displays. Each of which will have a different appendix number. Hence the FSOC FORMAT is a solution to this problem. The following $TABLE format is an example:

```
proc format;
  value stable
    'l.dem' = 'Appendix D.1'
    'l.ab' = 'Appendix D.2'
    's.dem.a' = 'Appendix C.1'
    's.dem.b' = 'Appendix C.2'
    's.ab.a' = 'Appendix C.22'
    's.ab.b' = 'Appendix C.23'
  ;
```

By defining a macro variable in each program which designates the program (APGM), a cross reference can be made using the
$TABLE format to determine the appendix number. To help with consistency, using the same code in the PGM variable as the program name provides a means of locating the program from the hard copy if the need arises. The PGM variable would be set prior to the DATA _NULL_ step of your program using the following syntax:

```sas
%LET PGM = "L..DEMO";
```

The last %IF portion of the HDG macro shows how to use the PUT function with the $TABLE format to get a file containing the appropriate appendix number. This can then be displayed using a PUT statement.

### Patient Identifying Information (TOPLIST MACRO)

The patient identifying information is usually displayed consistently across all listings. The sort order of these is likely to be the same as well (although each table may have additional sorting variables).

For example if the top of each listing is to display the investigator name, patient's treatment group, patient number, and patient initials, and evaluability code, a macro can be defined to do this in the same manner for each listing. Below is an example of the TOPLIST macro which assumes the data coming into the DATA _NULL_ is sorted by at least investigator name, treatment group, and patient number:

```sas
%MACRO TOPLIST;
  %IF FIRST.PTNO OR TOP THEN DO;
  %IF FIRST.INVNAME OR TOP THEN PUT 1 'Investigator: ' INVNAME;
  %IF FIRST.TRTMNT OR TOP THEN PUT 2 'Treatment Group: ' TRTMNT;
  END;
  %IF FIRST.PTNO THEN PTCOUNT + 1;
  %PUTFLD (PTNO) %PUTFLD (evalcode) %PUTFLD (ptno) %PUTFLD (invname) %PUTFLD (trtmnt) %PUTFLD (evalcode);
%MEND;
```

This TOPLIST macro is then called by each data listing program to print the patient identifying information across all listings. The use of a nested IF structure is efficient and easy to follow. Due to the sort sequence, there can never exit a FIRST.INVNAME without a FIRST.TRTMNT. The TOP variable is defined in the First DATA _NULL_ to be a variable which is set each time a new page is begun. This way each new page will have the patient identifying information regardless of whether it is the first observation for that patient. Below is a DATA _NULL_ showing how each program would simply call this TOPLIST macro:

```sas
DATA _NULL;  
  SET X END=EOF;  
  PUT "FIRST.INVNAME, FIRST.TRTMNT, FIRST.PTNO, FORWARD END;" X;  
%TOPLIST;  
RUN;
```

This eliminates the chances of different programmers coding this section inconsistently. The PUTFLD macro used in this TOPLIST macro will be described in more detail below.

### Displaying each Data Field (COL, FMT, and PUTFLD MACROS)

Determine, for each display, the column where each data item is to begin. By defining a COL macro to define column numbers for each data item, both the column titles as well as for the actual data items can reference the same macro. Below is an example of a COL macro:

```sas
%MACRO COL (VAR);
  %LET AVAL = %UPCASE (VAR);
  %IF AVAL EQ SEX THEN 1;
  %ELSE IF AVAL EQ PTNO THEN 2;
  %ELSE IF AVAL EQ EVALCODE THEN 3;
  %ELSE IF AVAL EQ SEX THEN 4;
  %ELSE IF AVAL EQ AGE THEN 5;
  %ELSE IF AVAL EQ RACE THEN 6;
  %ELSE IF AVAL EQ HEIGHT THEN 7;
  %ELSE IF AVAL EQ WEIGHT THEN 8;
  %ELSE PUT "ERROR-NO DECODE FOR VAR";
%MEND;
```

This macro is driven by the variable name. Given a particular variable name, this macro will return a number which indicates the column in which the data is to be printed. This data, as mentioned above, can be the actual data, or can also be a trigger for the column headings as well. For example, the following code:

```sas
PUT %COL (PTNO) 'Patient ID';
PUT %COL (EVALCODE) 'Eval Code';
```

This would cause the titles to be printed in the columns specified by the COL macro.

There are many situations when the data coming into the data null is not formatted exactly as desired. For example, SEX may be denoted by an 'F' or 'M'. On the display, this would look nicer if it read 'Female' and 'Male'. Hence using a macro like the FMT macro below, formats can be assigned to each variable as necessary:

```sas
%MACRO FMT (VAR);
  %LET VAR = %UPCASE (VAR);
  %IF VAR EQ SEX THEN 3;
  %ELSE IF VAR EQ PTNO THEN 4;
  %ELSE IF VAR EQ AVAL THEN 5;
  %ELSE IF VAR EQ TRTMNT THEN 6;
  %ELSE IF VAR EQ HT THEN 7;
  %ELSE IF VAR EQ WT THEN 8;
  %ELSE PUT "ERROR-NO DECODE FOR VAR";
%MEND;
```

There need not be an entry in the FMT macro for every field. This would mean that the PUT statement would not have a format associated with it, which is not an error. Now that both the column and the format for each data item have been defined, a final macro is defined to pull these two together. This macro is called PUTFLD, since it is part of a PUT statement and is displaying a given data field:

```sas
%MACRO PUTFLD (NAME, OFFSET);
  %IF "OFFSET" EQ "" THEN %COL (NAME) + " OFFSET"
  %ELSE %COL (NAME) + " OFFSET"
%MEND;
```

This PUTFLD macro calls the COL macro to determine the column to begin 'putting' the data and calls the FMT macro to determine
the format to make the display of the data items come together. The offset parameter is helpful in that it tells the PUTFLD macro to move your data items over in the display for any given reason. If the PUTFLD macro is called with an offset=2, the data item will be displayed two columns to the right of the column indicated by the result of the COL macro for that variable. The most common reason would be to align data nicely under a longer column title. Below is an example of the DATA_NULL step which utilizes the PUTFLD macro:

```plaintext
DATA_NULL;
SET X END=EOF;
   BY INVNAME TRMT TNO (other vars);
   %TOPLIST;
   PUT %PUTFLD(Sex)
   %PUTFLD(RACE)
   %PUTFLD(HEIGHT)
   /;
RUN;
```

In the above DATA_NULL step the TOPLIST macro and the PUTFLD macro have been shown. The PUTFLD macro is called from within a PUT statement. This allows for the display of multiple fields within one PUT statement. This is especially handy when the fields are related.

The Footing (FOOT MACRO)

The footing of each listing may be the same across the board, or there may be just parts of it that are similar. Below is an example of a footing on listings where there is a solid line delimiting the actual data, then a legend type footer indicating what an "E" and an "N" stand for in the body of the listing.

```plaintext
%MACRO FOOT(FID,FNCT=FNCT);
   LINE=42+4*FNCT;
   PUT $1 'E=Evaluable, N=Not Evaluable';
%MEND;
```

The first item to note in the macro is the FNCT macro variable. This is a global variable which was set prior to the DATA_NULL step which defined the number of lines on the page, or pagez. The FNCT macro variable is the variable which contains the number of lines necessary to display all the footnote lines. This footer assumes that EVERY listing will have both the solid line and the legend text for "E" and "N". Additional footer lines can be coded right into the individual programs, or a select statement based on the FID macro variable can be used. For example:

```plaintext
%MACRO FOOT(FID,FNCT=FNCT);
   LINE=42+4*FNCT;
   PUT $1 'Patients not included on this listing have no
   'Adverse Events';
   OTHERWISE
   PUT $1 'E=Evaluable, '
   'N=Not Evaluable';
%MEND;
```

This example shows that for the LAE listing, the 'E' and 'N' legend text should not be displayed. Instead, a description of why all the patients are not included in the display is printed. Keeping the more generic footnotes in this FOOT macro make it much easier to change them if there happens to be a typographical error, or if there is need for a change in a footnote that appears in multiple displays.

The END of the Display (EOF MACRO)

When the last record in the data set has been read, some final processing needs to be done. The program has been keeping a count of the number of patients being included in the listing (via the TOPLIST macro), hence, displaying this total number of patients can be done. Since there is no more data to process, the program can display text indicating this is the last appendix page. Below is a macro which will take care of these two final issues:

```plaintext
%MACRO EOF;
   IF EOF THEN DO;
      LINE FOOT;
      PUT %1 'LAST APPENDIX;';
      PUT PAGE= /910 'Total Patients: '
      //910 'PAGECOUNT; '
      END;
%MEND;
```

First, there is a test to see if the end of the file has been reached. Note that in order to determine this, there must be an END=EOF option in the SET statement of the DATA_NULL step. If the end of the file has been reached, the footer is displayed, and on the first line of the current display page, the words LAST APPENDIX are added to the PAGE number line. Then a new page is begun and the only thing displayed on this page is the patient count. This patient number count be helpful to be sure that all listings contain information for the same number of patients. Any other data counts that are important to the data listing can be maintained in the code and displayed in this manner.

The DATA_NULL_

To pull all of these macros together, one DATA_NULL can be used to show how these macros all interact. Below is the DATA_NULL_:

```plaintext
DATA_NULL;
SET X END=EOF;
   BY INVNAME TRMT TNO (other vars);
   FILE PRINT HEADER=HOLD NOTITLE
   LINES=PS=FACT SET X BND=-BOF;
   %TOPLIST;
   PUT %PUTFLD(Sex)
   %PUTFLD(RACE)
   %PUTFLD(HEIGHT)
   %PUTFLD(WT)
   %PUTFLD(AGB10FFSET=1)
   /;
   %TOPLIST;
   PUT //910 'Patients not included on this listing have no
   'Adverse Events';
   OTHERWISE
   PUT %1 'E=Evaluable, '
   'N=Not Evaluable';
%MEND;
```

This example shows that for the LAE listing, the 'E' and 'N' legend text should not be displayed. Instead, a description of why all the patients are not included in the display is printed. Keeping the more generic footnotes in this FOOT macro make it much easier to change them if there happens to be a typographical error, or if there is need for a change in a footnote that appears in multiple displays.

The END of the Display (EOF MACRO)

When the last record in the data set has been read, some final processing needs to be done. The program has been keeping a count of the number of patients being included in the listing (via the TOPLIST macro), hence, displaying this total number of patients can be done. Since there is no more data to process, the program can display text indicating this is the last appendix page. Below is a macro which will take care of these two final issues:

```plaintext
%MACRO EOF;
   IF EOF THEN DO;
      LINE FOOT;
      PUT %1 'LAST APPENDIX;';
      PUT PAGE= /910 'Total Patients: '
      //910 'PAGECOUNT; '
      END;
%MEND;
```

First, there is a test to see if the end of the file has been reached. Note that in order to determine this, there must be an END=EOF option in the SET statement of the DATA_NULL step. If the end of the file has been reached, the footer is displayed, and on the first line of the current display page, the words LAST APPENDIX are added to the PAGE number line. Then a new page is begun and the only thing displayed on this page is the patient count. This patient number count be helpful to be sure that all listings contain information for the same number of patients. Any other data counts that are important to the data listing can be maintained in the code and displayed in this manner.

The DATA_NULL_

To pull all of these macros together, one DATA_NULL can be used to show how these macros all interact. Below is the DATA_NULL_:

```plaintext
DATA_NULL;
SET X END=EOF;
   BY INVNAME TRMT TNO (other vars);
   FILE PRINT HEADER=HOLD NOTITLE
   LINES=PS=FACT SET X BND=-BOF;
   %TOPLIST;
   PUT %PUTFLD(Sex)
   %PUTFLD(RACE)
   %PUTFLD(HEIGHT)
   %PUTFLD(WT)
   %PUTFLD(AGB10FFSET=1)
   /;
   %TOPLIST;
   PUT //910 'Patients not included on this listing have no
   'Adverse Events';
   OTHERWISE
   PUT %1 'E=Evaluable, '
   'N=Not Evaluable';
%MEND;
```

This example shows that for the LAE listing, the 'E' and 'N' legend text should not be displayed. Instead, a description of why all the patients are not included in the display is printed. Keeping the more generic footnotes in this FOOT macro make it much easier to change them if there happens to be a typographical error, or if there is need for a change in a footnote that appears in multiple displays.
PUT &G1.; &LS.; /  
/ &GCOL(PATNO); 'Patient'  
/ &GCOL(PATNAME); 'Patient'  
/ &GCOL(EVALCODE); 'Evaluability'  
/ &GCOL(AGE); 'Age'  
/ &GCOL(HEIGHT); 'Height'  
/ &GCOL(WEIGHT); 'Weight'  
/ &GCOL(PATNO); 'Number'  
/ &GCOL(AGS); 'Gestation'  
/ &GCOL(EVALCODE); 'Code'  
/ &GCOL(SEX); 'Sex'  
/ &GCOL(AGE); '(yrs)'  
/ &GCOL(RACE); 'Race'  
/ &GCOL(HEIGHT); '(cm)'  
/ &GCOL(WEIGHT); '(kg)'  
&G1. &LS. ' /  
RETURN;  
/  
/  
FOOT:  
%FOOT(&PGM);  
PUT &G1.; 'Severity: '  
/ &G2.; '1= Mild',  
/ &G2.; '2= Moderate',  
/ &G2.; '3= Severe',  
RETURN;  
/  
This DATA _NULL_ incorporates all the macros which have been 
presented within this paper. The FILE statement is partial but 
includes the items necessary for the macros to run. The PS and 
LS macro variables referenced here are global variables indicating 
page size and line size. In the FILE statement, the HEADER=HDG 
refers to the lines of code which are executed at the top of each 
page. These are found at the HDG: label below the RETURN 
statement of the DATA _NULL_. These are not to be confused 
with the HDG macro which is called from this section of code. The 
code in the FOOT section is called at the end of each page (code 
not included here), and at the end of the file as well. At the end 
of this paper, there is an example of the output produced by this 
DATA _NULL_ step.  

Modifying the Display  
For one reason or another, there are always changes to be made 
to the listings. By using these macros, changes to the displays can 
be made quickly and easily. For example, if the RACE field 
(decoded) is taking up too many columns for the display to look 
nice, a simple change to the column numbers in the COL macro 
will adjust for this change. A simple re-run of the program will 
automatically make the changes to the titles as well as to the data 
itself. Another example would be the addition of a data field. This 
would be done using the following simple steps:  
- add an entry to the COL macro for the new field.  
- modify COL numbers to separate out all the fields with 
  this new addition.  
- add an entry to the FMT macro for the new field.  
- in the DATA _NULL_ add a PUTFLD call for the new 
  field.  
- add the necessary title information to the heading of the 
  DATA _NULL_ for the new field.  

Again, a re-run will then display the new field, with all other fields 
shifted properly.  

Conclusion  
The PGM macro variable, the COL macro and the FMT macro are 
specific to each program and must reside within each program. All 
other macros can be placed in an included piece of code, or they 
may reside in a SAS MACRO library. These macros are very 
helpful in making the displays across an entire project (and project 
team) consistent. They make changing a display very easy as well 
as quick. These macros all work together to provide a fine tool in 
data table generation. They allow maximum flexibility while 
adhering to standard programming techniques.  

Author Information  
The SAS source code contained within this paper was developed 
by the following programmers at Bio-Pharm Clinical Services: Mike 
Menouie, Larry Specht, and Daphne Ewing.  

Bio-Pharm Clinical Services  
Four Valley Square  
512 Township Line Road  
Blue Bell, PA 19422  
(215) 283-0713 x425  

SAS is a registered trademark or trademark of SAS Institute, Inc. 
in the USA and other countries. * indicates USA registration.  

Other brand and product names and registered trademarks or 
trademarks of their respective companies.
Re-cap of All the SAS Code

%MACRO HDG(BID,PAGB=PNUM,TITLB=TITLEzFNCT=FNCT);
  PUT @1 'Pharmaceutical company' ~IF ~&PAGE·
  THEN &PAGB 'PAGE' &PAGE / @1 'Protocol No. ABC-123'
  &TITLE 'clinical/Statistical Report'
  ~STR(1;)
  PUT ~CENTER &TITLE @'END~
  %MEND;

PROC FORMAT;
  VALUE STABLE
    'L.DEMO' = 'Appendix 0.1'
    'L.AE' = 'Appendix 0.2'
    'S.DEMO.A' = 'Appendix C.1'
    'S.DEMO.E' = 'Appendix C.2'
    'S.AE.A' = 'Appendix C.22'
    'S.AE.R' = 'Appendix C.23';

%MACRO TOPLIST;
  IF FIRST.PTNO OR TOP THEN DO;
    IF FIRST.TRTMNT OR TOP THEN DO;
      IF FIRST.INVNAME OR TOP THEN
        PUT @1 'Investigator: ' INVNAME;
        END;
    IF FIRST.PTNO THEN PTCOUNT + 1;
    PUT %PUTFLD(PTNO); %PUTFLD(EVALCODE) @;
  END;
%MACRO COL(VAR);
  %LET VAR = %UPCASE(&VAR);
  %IF &VAR EQ PTNO %THEN 1;
  %ELSE %IF &VAR EQ PTNITS %THEN 8;
  %ELSE %IF &VAR EQ EVALCODE %THEN 15;
  %ELSE %IF &VAR EQ SEX %THEN 25;
  %ELSE %IF &VAR EQ AGE %THEN 35;
  %ELSE %IF &VAR EQ RACE %THEN 45;
  %ELSE %IF &VAR EQ HEIGHT %THEN 55;
  %ELSE %IF &VAR EQ WEIGHT %THEN 65;
  %ELSE PUT 'ERROR-NO DECODE FOR &VAR';
%MACRO FMT(VAR);
  %LET VAR = %UPCASE(&VAR);
  %IF &VAR EQ EVALCODE %THEN SEVC;
  %ELSE %IF &VAR EQ SEX %THEN SEX;
  %ELSE %IF &VAR EQ AGE %THEN AGE;
  %ELSE %IF &VAR EQ RACE %THEN RACE;
  %ELSE %IF &VAR EQ HEIGHT %THEN HEIGHT;
  %ELSE %IF &VAR EQ WEIGHT %THEN WEIGHT;
%END;
Beginning Tutorials

Re-cap of All the SAS Code (continued)

The overall DATA _NULL_

DATA _NULL_;  
SET X END=EOF;  
BY INVNAME TRTMT PTNO {other vars};  
FILE PRINT HEADER=HDG NOTITLES  
LL=LS PS=PS N=PS;  
...  
VTUPLIST;  
PUT %PUTFLD(SEX)  
%PUTFLD(AGE,OFFSET=1)  
%PUTFLD(RACE)  
%PUTFLD(HEIGHT)  
%PUTFLD(WEIGHT);  
RETURN;  
HDG:  
TOP=1; /* Increment page number */  
FNT=1; /* number of footnote lines */  
$HDG(&PGM);  
PUT %CENTER 'Listing of Patients Demographics';  
PUT @l &LS*' /  
@%COL(PTNO) 'Patient'  
@%COL(PTINITS) 'Patient'  
@%COL(EVALCODE) 'Evaluability'  
@%COL(AGE) 'Age'  
@%COL(HEIGHT) 'Height'  
@%COL(WEIGHT) 'Weight'  
@%COL(PTNO) 'Number'  
@%COL(PTINITS) 'Initials'  
@%COL(EVALCODE) 'Code'  
@%COL(SEX) 'Sex'  
@%COL(AGE) '(yr)'  
@%COL(RACE) 'Race'  
@%COL(HEIGHT) '(cm)'  
@%COL(WEIGHT) '(kg)'  
@g &LS'' /;  
RETURN;  
FOOT:  
$FOOT(&PGM);  
PUT $1 'Severity: 1=Mild,  
' '2=Moderate,  
' '3=Severe.';  
RETURN;  
RUN;