DANCES WITH DATA

Ginger Barlow, Boots Pharmaceuticals, Inc.
Angela S. Ringelberg, Boots Pharmaceuticals, Inc.

ABSTRACT: Under the confines of different line sizes and page sizes, how do you produce custom reports that are 'reader-friendly'? What do you do when long character strings wrap to the next line without taking into account the structure of the string, (i.e. spacing, punctuation, etc.)? How do you force pages to break between groups when all the observations in the next group will not fit on the current page? If numeric data is stored as character, how can you line up the decimal places?

In the process of customizing reports, we have developed several tools and techniques to manipulate data using the DATA _NULL_ step.

I. EXPLANATION OF PROBLEM

When comparing DATA _NULL_ and PROC PRINT, there are advantages and disadvantages of using either step. PROC PRINT is quick and easy, but lacks the flexibility of the DATA _NULL_ step in data presentation. DATA _NULL_ may require more programming, but it gives the programmer more layout options.

With DATA _NULL_ the programmer has absolute control over the placement of headings, variables, constants and page breaks. She can use conditions, first/last.variables, and loops to customize the layout of data without altering or creating a new data set.

The three techniques included below are useful in producing customized reports with the DATA _NULL_ step. The first tool will convert character digits to numeric values in one data step, while retaining the original variable name. The second technique includes a macro that counts the number of observations in particular BY groups. We show how to use this information to break pages between chosen BY groups. The last technique presented is a macro that breaks a long character string into two shorter strings, breaking between words and not within a word. These two strings can be placed on separate lines thereby taking less horizontal space than the longer string.

II. TECHNIQUES USED

A. CONVERSION FROM CHARACTER DIGITS TO NUMERIC VALUES

Because of constraints of different data entry systems and procedures, it is sometimes necessary to convert one data type to another. One reason we use this conversion is to produce formatted output in the DATA _NULL_ step. This technique is also an efficient way to convert values for merging or for comparison checks in datasets which contain the same variable but of different types.

In this example we are converting from character digits to numeric values in one data step and maintaining the original variable name.

```plaintext
DATA dataset (DROP=old_string
RENAME=(numeric_value=old_string));
SET dataset ;
numeric_value=INPUT(old_string,8.) ;
RUN;
```

This data step creates a new variable which is a numeric variable. It drops the character variable and then assigns its name to the new numeric variable. Therefore, we maintain the same variable name as in the original data set but the variable type has been changed. This technique can also be used to convert several variables in the same data step. Now, for example, character digits that have been converted to numerical values can easily be formatted in the output statement allowing decimals to line up under each other.

B. PROC MEANS TO COUNT OBSERVATIONS

For our reporting purposes, it is necessary that we break pages between and not within groups. If the groups have varying numbers of observations, we use the technique in Figure 1 to control page breaks in the DATA _NULL_ step. It includes a macro that uses PROC MEANS to count the number of observations in each group.

The macro in Figure 1 reproduces the original data set with a new variable called NUNES which contains the number of observations in the specified group.
C. BREAKING LONG CHARACTER STRINGS

When producing customized reports that include long character strings, you cannot always fit all of the character string on one line. It may be necessary to break a long character string and place it in the specified column on the next line without breaking in the middle of a word.

The macro in Figure 2 produces two variables from the original string (broken between words.) OHALF is the first half of the string and THALF the second.

III. PROGRAM EXAMPLE AND OUTPUT

In Figure 3, we have used all three techniques described above to produce a patient listing of current medications. The file used is a SAS® data set containing a list of patient medications. The seven variables in the data set are (see Figure 5):

PATIENT identifying number assigned to the patient
MEDDATE date of visit
MED medication
MEDSTART date medication started
MEDSTOP date medication stopped
MEDON "Y" if medication is ongoing
INDICAT condition requiring medication

The data is grouped by patient and date of visit (MEDDATE). (Note that the OBSCOUNT macro sorts the data in this order.) We chose to allow breaking pages within a patient number, but not within a visit. The OBSCOUNT macro is used to count the number of observations at each visit. The INDICAT variable is too long to fit in the number of columns available, so the SCANNER macro is invoked. We also converted a character string into a SAS date, while retaining the same variable name. (See Figure 3, Section 1.)

In Figure 3, Section 2, we use the FILE option PRINT to send the output to a print file and define HEADER, LINESLEFT and PAGESIZE variables to use in the DATA _NULL_ step.

The variable NLINES from the OBSCOUNT macro in Figure 3, Section 3, breaks pages between visits. If there are too many observations within the current group to fit on the current page (LINESLEFT < NLINES + buffer) the program will break to a new page. We also carry over any group information that will not be printed by later PUT statements (ie. PATIENT).

In Figure 3, Section 4, we use FIRST.variable to avoid printing out repetitive data. In this example, we only want PATIENT to print at the first observation for that patient or at the top of a new page, and MEDDATE (visit date) to print once for each visit. We use LAST.variable to put a blank line between groupings. (See Figure 3, Section 4.) Figure 4, Section 1, designates where the INDICAT variable has been broken by the SCANNER macro.

IV. CONCLUSION

We have presented three techniques to use with the DATA _NULL_ step to produce custom reports. With formal reports, there may be a need for more flexibility than PROC PRINT allows. However, for certain reports and debugging purposes PROC PRINT is quick and easy, and more than adequate. When deciding how to present data, a programmer should consider the cost/benefit of customizing with the DATA _NULL_ step, or producing a fast listing using PROC PRINT.

CONTACT INFORMATION

Ginger Barlow / Angela S. Ringelberg
Boots Pharmaceuticals, Incorporated
Statistics and Clinical Information
8800 Ellerbe Road
Shreveport, Louisiana 71106
(318) 861-8200

TRADEMARKS

SAS is a registered trademark of SAS Institute, Incorporated.

REFERENCES


FIGURE 1

* PURPOSE: THIS MACRO GIVES THE NUMBER OF OBSERVATIONS IN A SPECIFIED GROUP. THE
* OBSERVATION COUNT OBTAINED WILL BE USED FOR OUTPUT USING A DATA _NULL_
* TO CONTROL PAGE BREAKS.
* 2 VALUES MUST BE GIVEN WHEN INVOKING THIS MACRO:
* ORIGINAL = THE DATA SET FROM WHICH THE OBSERVATION COUNT WILL BE COMPUTED.
* SORTVARS = THE LIST OF VARIABLE(S) THAT THE DATA SET MUST BE SORTED BY. THE LAST
* VARIABLE IN THE SORT LIST IS THE VARIABLE ON WHICH THE NUMBER OF
* OBSERVATIONS WILL BE COMPUTED.
* TO INVOKE THE MACRO: %OBSCOUNT(data set, sort variable list);

%MACRO OBSCOUNT(ORIGINAL, SORTVARS);
   PROC SORT DATA=&ORIGINAL;
      BY &SORTVARS;
   RUN;

%* PICK OFF LAST VARIABLE FROM BY LIST FOR &CNTVAR VARIABLE IN PROC MEANS **;
   DATA &ORIGINAL;
      SET &ORIGINAL;
      WORD=""||&SORTVARS||"";
      BACKWARD=REVERSE(WORD);
      FIRSBACK=SCAN(BACKWARD, 1);
      VARVAR=COMPRESS(REVERSE(FIRSBACK));
      FOUND=INDEX(VARVAR, "");
      LEN=LENGTH(VARVAR);
      IF FOUND=1 THEN VAR=COMPRESS(SUBSTR(VARVAR, 2, LEN-2));
      ELSE VAR=COMPRESS(SUBSTR(VARVAR, 1, LEN-1));
      CALL SYMPUT('OBSVAR', VAR);
   RUN;

%LET CNTVAR=&OBSVAR;
%* FIND THE NUMBER OF OBS IN THE SPECIFIED GROUP **;
   PROC MEANS NOPRINT;
      BY &SORTVARS;
      VAR &CNTVAR;
   OUTPUT OUT=COUNT N=NLINES;
   RUN;

%* MERGE THE NUMBER OF OBS IN WITH THE ORIGINAL DATA SET **;
   DATA &ORIGINAL;
      MERGE &ORIGINAL COUNT;
      BY &SORTVARS;
   RUN;
%MEND OBSCOUNT;
FIGURE 2

• PURPOSE: MACRO TO BREAK UP LONG CHARACTER STRINGS BETWEEN WORDS
• THIS MACRO HAS 2 VALUES THAT MUST BE GIVEN WHEN INVOKING THE MACRO.
• UNELEN = THE NUMBER OF COLUMNS AVAILABLE FOR THIS STRING. (LESS
to 15 COLUMNS - THIS SPACE MARGIN IS REOUIRED BECAUSE
THE MACRO MAY HAVE TO CONCATENATE PART OF THE 2ND HALF
OF THE STRING TO THE FIRST HALF TO ALLOW BREAKS BETWEEN WORDS
AND NOT WITHIN A WORD.)
• VARNAME = THE CHARACTER VARIABLE TO BREAK
• TO INVOKE THE MACRO: %SCANNER(number of spaces available,variable)
• THE MACRO WILL PRODUCE TWO VARIABLES: OHALF-FIRST HALF OF THE STRING
THALF-SECOND HALF OF THE STRING

%MACRO SCANNER(LINELEN,VARNAMEx:

%*** DIVIDE STRING INTO TWO PARTS ***;
ONEHALF=SUBSTR(&VARNAME,1,&LINELEN);
TWOHALF=SUBSTR(&VARNAME,&LINELEN+1);
SPACE1=SUBSTR(ONEHALF,&LINELEN,1);
SPACE2=SUBSTR(TWOHALF,1,1);
SPLIT=SCAN(TWOHALF,1,space);

%*** SET UP 1ST AND 2ND PART OF STRING WITHOUT BREAKING WITHIN A WORD ***;
IF SPACE1=" THEN
DO;
  IF SPACE2=" THEN
    DO;
      OHALF=TRIM(ONEHALF)||SPLIT;
      START=LENGTH(SPLIT);
      THALF=SUBSTR(TWOHALF,START+1);
    END;
  ELSE
    DO;
      OHALF=ONEHALF;
      THALF=TWOHALF;
    END;
  ELSE IF SPACE1=" THEN
    DO;
      OHALF=ONEHALF;
      THALF=TWOHALF;
    END;
%MEND SCANNER;

765
FIGURE 3

* PURPOSE: GENERATE A LISTING OF CURRENT MEDICATIONS, CATEGORIZE BY PATIENT AND DATE. *

OPTIONS CC=CR ERRORABEND NOCENTER NODATE;

LIBNAME BPIXX 'BPIXXST.TEXT';
FILENAME SCANNER 'BPIXXST.TEXT|SCANNER.SAS';
FILENAME OBSCOUNT 'BPIXXST.MACROS|NLINES.SAS';
TITLE1 'BOOTS PHARMACEUTICALS';
TITLE2 ';

PROC FORMAT;
  VALUE $ONGO 'Y'='ONGOING';
RUN;

%INC SCANNER;
%INC OBSCOUNT;

*** GET CURRENT MEDICATIONS ***;

DATA CURMED (DROP=MEDDATE RENAME=(NMEDDTE=MEDDATE)) ;
  SET BPIXXX.CURMED (WHERE=(MED=' ', AND MEDVNONE));
  NMEDDTE=INPUT(MEDDATE,MMDDVYS.);
RUN;

%OBSCOUNT(CURMED,PATIENT MEDDATE) ;

*** OUTPUT CURMED DATA ***;

DATA _NULL_;
  SET CURMED END=EOF ;
  BY PATIENT MEDDATE ;
  FILE PRINT HEADER=HD LL=LL PS=PS60 ;
  FORMAT MEDDATE MMDDVYS. ;
  IF FIRST.MEDDATE AND LL<NLINES+4 THEN DO;
    PUT _PAGE_;
    IF *FIRST.PATIENT THEN PUT @4 PATIENT @ ;
    END;
  IF FIRST.PATIENT THEN PUT @4 PATIENT @ ;
  IF FIRST.MEDDATE THEN PUT@11 MEDDATE @ ;
  PUT @22 MED @55 MEDSTART @ ;
  IF MEDSTOP '=' THEN PUT @66 MEDSTOP @ ;
  ELSE PUT @66 MEDON $ONGO. @ ;
  %SCANNER(20,INDICAT) ;
  IF THALF=' ' THEN PUT @77 OHALF / @77 THALF ;
  ELSE PUT @77 INDICAT ;
  IF LAST.PATIENT THEN PUT ;
  IF LAST.MEDDATE THEN PUT ;
  RETURN ;

HD: PUT @1'TABLE X. CURRENT MEDICATIONS /
  @1 'PATIENT DATE MEDICATION '
  @55 'STARTED ONGOING INDICATION/ 
  @1 '------ ------ --------------------- ' 
  @55 '------ ------ --------------------- ' ;

RUN ;

766
### TABLE X. CURRENT MEDICATIONS

<table>
<thead>
<tr>
<th>PATIENT</th>
<th>DATE</th>
<th>MEDICATION</th>
<th>STARTED</th>
<th>ONGOING</th>
<th>INDICATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>06/25/80</td>
<td>PAIN KILLER</td>
<td>04/00/80</td>
<td>ONGOING</td>
<td>AS INDICATED FOR FRONTAL HEADACHE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PAIN KILLER</td>
<td>00/00/88</td>
<td>ONGOING</td>
<td>AS INDICATED FOR BACK PAINS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VITAMIN</td>
<td>00/00/88</td>
<td>ONGOING</td>
<td>POTASSIUM SUPPLEMENT</td>
</tr>
<tr>
<td>07/03/80</td>
<td>VITAMIN</td>
<td>00/00/88</td>
<td>ONGOING</td>
<td></td>
<td>POTASSIUM SUPPLEMENT</td>
</tr>
<tr>
<td>09/27/80</td>
<td>VITAMIN</td>
<td>00/00/88</td>
<td>ONGOING</td>
<td></td>
<td>POTASSIUM SUPPLEMENT</td>
</tr>
<tr>
<td>10/15/80</td>
<td>VITAMIN</td>
<td>00/00/88</td>
<td>ONGOING</td>
<td></td>
<td>POTASSIUM SUPPLEMENT</td>
</tr>
<tr>
<td>26</td>
<td>07/12/80</td>
<td>BURN CREAM</td>
<td>10/06/85</td>
<td>10/12/85</td>
<td>SEVERE 3RD DEGREE BURN</td>
</tr>
<tr>
<td>07/25/80</td>
<td>THYROID MED</td>
<td>00/00/88</td>
<td>ONGOING</td>
<td></td>
<td>HYPOTHYROIDISM</td>
</tr>
<tr>
<td>08/08/80</td>
<td>THYROID MED</td>
<td>00/00/88</td>
<td>ONGOING</td>
<td></td>
<td>HYPOTHYROIDISM</td>
</tr>
</tbody>
</table>

### FIGURE 5

<table>
<thead>
<tr>
<th>OBS</th>
<th>PATIENT</th>
<th>MEDDATE</th>
<th>MED</th>
<th>MEDSTART</th>
<th>MEDON</th>
<th>MEDSTOP</th>
<th>INDICATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>26</td>
<td>06/25/80</td>
<td>PAIN KILLER</td>
<td>04/00/80</td>
<td>N</td>
<td>05/4/80</td>
<td>AS INDICATED FOR FRONTAL HEADACHE</td>
</tr>
<tr>
<td>2</td>
<td>26</td>
<td>06/25/80</td>
<td>PAIN KILLER</td>
<td>00/00/88</td>
<td>Y</td>
<td></td>
<td>AS INDICATED FOR BACK PAINS</td>
</tr>
<tr>
<td>3</td>
<td>26</td>
<td>05/20/80</td>
<td>VITAMIN</td>
<td>00/00/88</td>
<td>Y</td>
<td></td>
<td>POTASSIUM SUPPLEMENT</td>
</tr>
<tr>
<td>4</td>
<td>26</td>
<td>07/15/80</td>
<td>VITAMIN</td>
<td>00/00/88</td>
<td>Y</td>
<td></td>
<td>POTASSIUM SUPPLEMENT</td>
</tr>
<tr>
<td>5</td>
<td>26</td>
<td>02/07/80</td>
<td>VITAMIN</td>
<td>00/00/88</td>
<td>Y</td>
<td></td>
<td>POTASSIUM SUPPLEMENT</td>
</tr>
<tr>
<td>6</td>
<td>26</td>
<td>10/15/80</td>
<td>VITAMIN</td>
<td>00/00/88</td>
<td>Y</td>
<td></td>
<td>POTASSIUM SUPPLEMENT</td>
</tr>
<tr>
<td>7</td>
<td>26</td>
<td>10/15/80</td>
<td>PAIN KILLER</td>
<td>10/11/90</td>
<td>N</td>
<td>10/12/90</td>
<td>BACK PAIN</td>
</tr>
</tbody>
</table>

767