WE-R-DRUGS PHARMACEUTICALS CORP.: SAS* PROGRAMMING WRITTEN BY MACROS

Ira A. Bader, New York City Health & Hospitals Corporation

Background

BP-Down is a new treatment for high blood pressure that is being tested by the We-R-Drugs Pharmaceuticals Corp.

Multi-center clinical trials for the new drug require a fixed schedule of patient visits over the course of each experiment. In one research protocol, each patient must report for blood pressure readings and a new supply of medication on Mondays of alternate weeks, for a total of ten visits. A two-week supply of BP-Down is dispensed at Visits 4 through 9, with placebo given at the other visits.

Although the patients are required to adhere to this visit schedule they sometimes come on other days. Cars break down, vacations interfere, people get sick, etc. Blood pressure readings taken at these unscheduled "interim" visits must also be recorded and entered into the data base.

Description of Data

One observation in dataset BP contains one patient's information of a particular kind across all visits. Each patient has three observations: one for systolic blood pressure, one for diastolic blood pressure, and one for date-of-visit. (Blood pressure is reported as "systolic over diastolic" as in "130/85".) Variables such as V001, V004, and V10A contain data for visits 1, 4, and 10A (an interim visit) respectively.

The Problem

Jack Q. Megabyte is the programmer who must produce the efficacy reports for BP-Down. He has been given strict instructions by his boss to have all the report writing programs completed well in advance so they will be ready to run as soon as the data base department releases the data. The only problem is that Jack can't know which interim visits have occurred until he sees the data. So the programs that run perfectly with test data will need extensive revisions for adding the interim variables V000, V01A, V01B, V04Z, etc.

Furthermore, each of the 14 evaluations of BP-Down has its own research protocol, requiring different patterns of regular visits. So even if the programmer correctly produces the programs for the first protocol, the job will need to be done 13 more times for the remaining protocols.

Wouldn't it be nice if the programmer could omit all mention of visit-specific variable names (V001, V01A, V002, etc.) from the programs. Let base SAS software figure out what the variable names are, what visits they pertain to, and then write SAS code to process the specific pattern of visits. Will our programmer be able to write a program only once that will execute 14 different ways for 14 different patterns of visits?

The Solution

The visit designations 000, 001, 01A, etc. will be read from the variable names supplied by PROC CONTENTS, and inserted into a series of macro variables by calls to the SYMPUT function. This list of visits will then furnish the building blocks to reconstitute the list of visit-specific variables. Here's how it's done.

List of visit-specific variables - Run PROC CONTENTS with option NOPRINT and create an output dataset. The variable called NAME in the output dataset has as its values all the variable names found in the original dataset. Use the SUBSTR function to identify and keep only those variables with names whose initial letter is "V", i.e., the visit-specific variables, such as V000, V001, V01A, etc.

```
PREFIX = SUBSTR(NAME,1,1);
    IF PREFIX = 'V';
        VISIT = SUBSTR(NAME,2,3);
```

Get the visits in the right order - Since the visit designations are all 3 characters long, packed with leading zeros, here is how they sort: 000, 001 002, 003, 004, 005, 006, 007, 008, 009, 010, 01A, 01B, 042, 06A, 09A, 10A. Most of the interim visits come at the end of the list. We would like each interim to directly follow the regular visit to which it pertains, e.g., 01A after 001, 09A after 009, etc.

Now create a sort variable called VSORT containing the values of VISIT. Apply the following rule, using the SUBSTR, INDEXC, and LEFT functions, to get the visits sorted out in the order in which they occur: If VISIT starts with zero, and the third position is a number, (i.e., to exclude the interim designators "A", "B" or "Z") then remove the leading zero from VSORT, and left justify.

```
IF SUBSTR(VISIT,1,1) = '0' AND
INDEXC(SUBSTR(VISIT,3,1),'0123456789')=1
    THEN SUBSTR(VSORT,1,1) = ' ';
        VSORT = LEFT(VSORT);
```

Visit numbers become values of macro variables - Apply PROC TRANSPOSE to your sorted visit numbers, and your output dataset will contain one variable per visit. 000 is the value of COL1, 001 is the value of COL2, 01A is the value of COL3 . . . and 10A becomes the value of COL17. These visit designations are the building blocks SAS needs in order to write your program for you. They will be supplied in each iteration of %DO loops to synthesize your SAS code.

When you run your %DO loop, you will need to close the loop when you reach the last visit. You can count the visits by using a technique taught in the SAS Macro Language Course for counting the number of observations in a dataset. Set the VISITS dataset into a DATA _NULL_ step, and use the POINT=N and NOBS options to count the observations (the number of observations exactly corresponds to the number of visits.)

```
DATA _NULL_;
   IF 0 THEN SET VISITS POINT=N NOBS=COUNT;
   CALL SYMPUT ('NVIS',LEFT(PUT(COUNT,3.)));
      STOP; RUN;
```

A call to SYMPUT puts this number into macro variable NVIS which you can use to close your %DO loops.

```
%MACRO MVAR;
DATA _NULL_; SET TRANSVIS;
  %DO I=1 %TO &NVIS;
  %GLOBAL VISNUM&I;
  CALL SYMPUT ("VISNUM&I",TRIM(LEFT(COL&I)));
    %END;
%MEND MVAR;
```

In this case, the %DO loop in macro MVAR calls SYMPUT 17 times, each time inserting a visit designation into a macro variable. The value of VISNUM1 is 000, VISNUM2 is 001, VISNUM3 is 01A . . . and VISNUM17 is 10A.

You are now ready to write as many macros as you need to generate SAS code with the visit designations 000, 001, 01A, etc. inserted wherever you want them.

You write the macros, base SAS software writes the SAS - Let's say you want PROC PRINT to produce a simple listing of your data, and you want to specify the variables in a VAR statement. Here is what your VAR statement looks like:

```
VAR %MVARLIST ;
```

A %DO loop in this macro generates the variable names V000, V001, V01A etc. as the successive values of V&&VISNUM&I.

```
%MACRO MVARLIST;
    %DO I=1 %TO &NVIS;
        V&&VISNUM&I
    %END;
%MEND MVARLIST;
```

Say you want a fancy listing with 3-line column headings. Here is what your LABEL statement looks like:

```
LABEL %MLABEL ;
```

The macro MLABEL expands V&&VISNUM&I = "VISIT* &&VISNUM&I*-----" to

```
V000 = "VISIT* 000*-----"
V001 = "VISIT* 001*-----"
V01A = "VISIT* 01A*-----"
```

etc., once for each iteration of the %DO loop.

```
%MACRO MLABEL;
%DO I=1 %TO &NVIS;
   V&&VISNUM&I="VISIT* &&VISNUM&I*-----"
%END;
%MEND MLABEL;
```

The same technique can be used to write FORMAT statements, RENAME statements or anything else you like. The programmer-written code in Figure 1 produces the report in Figure 4. The macro MPRINTIT in Figure 2 generates the identical code (Figure 3) producing the same report (Figure 4).

The Benefits

In walks Dr. Apothecary, the head researcher, and says, "Whose idea was it to have a Visit 4Z? We never had Z-visits before." You explain that a statistician wanted to flag interim visits occurring before the full therapeutic effect of BP-Down had a chance to build up.

After Dr. Apothecary fires the statistician he instructs the data base department to move all 4Z data to Visit 5. Do you now have to eliminate all 15 references to "04Z" in your report-writing program? You can't. They're not there. Remember, base SAS software writes your program, not you. Once variable V04Z is eliminated from the data base, all variable lists in your program will be automatically adjusted. If your data center would run the job at 2 o'clock in the morning, all the changes to your program would literally be made while you sleep.

Other things you can't do:

* Make keying errors while entering visit numbers.

* Correctly change 14 references to "04Z" but forget to change the 15th.

* Revise the program from memory and mistakenly enter the visits found in a

similar but not identical protocol.

* Run this program against data from the wrong protocol.

In short, your references to the visits can never be wrong because base SAS software writes the programming from scratch on each execution, exactly in accordance with the variable names currently found in the input dataset.

Acknowledgments

I would like to thank Dr. Henry T. Davis of Rorer Central Research for emphasizing the importance of developing programs that will work from one protocol to the next. I also thank Bernadette Tang Tom and James Enright of the New York City Health and Hospitals Corp. for their helpful suggestions.

*SAS is a registered trademark of SAS Institute, Inc., Cary, NC, USA.

For additional information contact Ira A. Bader, NYC Health & Hospitals Corp., 230 W. 41 St., New York, NY 10036, (212) 391-6281.

```
409        *====                                                    ====;
410        *====    DATA _NULL_ STEP PRODUCES THE MAIN REPORT     ==================;
411        *====                                                    ====;
412        DATA _NULL_;
413            FILE PRINT;
414                 SET ALL;
415        PUT  #3  @38 'RESEARCH CENTER' +1 CENTER @78 'PATIENT I.D.' +1 PATIENT
416             #6  @28 78*'-'
417             #8  @38 'VISIT'                  @48 'DATE OF VISIT'
418                 @66 'SYSTOLIC B.P.'          @84 'DIASTOLIC B.P.'
419             #9  @38 '-----'                  @48 '--------------'
420                 @66 '--------------'         @84 '--------------' / ;
421        IF DATE000 NE . THEN PUT
422            @39 '000'  @51 DATE000  @70 SYST000 3. @88 DIAS000 3. / ;
423        IF DATE001 NE . THEN PUT
424            @39 '001'  @51 DATE001  @70 SYST001 3. @88 DIAS001 3. / ;
425        IF DATE01A NE . THEN PUT
426            @39 '01A'  @51 DATE01A  @70 SYST01A 3. @88 DIAS01A 3. / ;
427        IF DATE01B NE . THEN PUT
428            @39 '01B'  @51 DATE01B  @70 SYST01B 3. @88 DIAS01B 3. / ;
429        IF DATE002 NE . THEN PUT
430            @39 '002'  @51 DATE002  @70 SYST002 3. @88 DIAS002 3. / ;
431        IF DATE003 NE . THEN PUT
432            @39 '003'  @51 DATE003  @70 SYST003 3. @88 DIAS003 3. / ;
433        IF DATE004 NE . THEN PUT
434            @39 '004'  @51 DATE004  @70 SYST004 3. @88 DIAS004 3. / ;
435        IF DATE04Z NE . THEN PUT
436            @39 '04Z'  @51 DATE04Z  @70 SYST04Z 3. @88 DIAS04Z 3. / ;
437        IF DATE005 NE . THEN PUT
438            @39 '005'  @51 DATE005  @70 SYST005 3. @88 DIAS005 3. / ;
439        IF DATE006 NE . THEN PUT
440            @39 '006'  @51 DATE006  @70 SYST006 3. @88 DIAS006 3. / ;
441        IF DATE06A NE . THEN PUT
442            @39 '06A'  @51 DATE06A  @70 SYST06A 3. @88 DIAS06A 3. / ;
443        IF DATE007 NE . THEN PUT
444            @39 '007'  @51 DATE007  @70 SYST007 3. @88 DIAS007 3. / ;
445        IF DATE008 NE . THEN PUT
446            @39 '008'  @51 DATE008  @70 SYST008 3. @88 DIAS008 3. / ;
447        IF DATE009 NE . THEN PUT
448            @39 '009'  @51 DATE009  @70 SYST009 3. @88 DIAS009 3. / ;
449        IF DATE09A NE . THEN PUT
450            @39 '09A'  @51 DATE09A  @70 SYST09A 3. @88 DIAS09A 3. / ;
451        IF DATE010 NE . THEN PUT
452            @39 '010'  @51 DATE010  @70 SYST010 3. @88 DIAS010 3. / ;
453        IF DATE10A NE . THEN PUT
454            @39 '10A'  @51 DATE10A  @70 SYST10A 3. @88 DIAS10A 3. / ;
455        PUT /  @28 78*'-' ;
456            RETURN;
```

Figure 1. Programmer-written code that produces the report in Figure 4.

```
58          *====                                                          ====;
59          *====    MACRO MPRINTIT - VARIABLES & TEXT FOR DATA _NULL_ STEP ====;
60          *====                                                          ====;
61          %MACRO MPRINTIT;
62              %DO I=1 %TO &NVIS;
63                  IF DATE&&VISNUM&I NE . THEN PUT
64                      @39    "&&VISNUM&I"
65                      @51 DATE&&VISNUM&I
66                      @70 SYST&&VISNUM&I 3.
67                      @88 DIAS&&VISNUM&I 3. / ;
68              %END;
69          %MEND MPRINTIT;
```

Figure 2. Macro MPRINTIT produces the code in Figure 3.

```
276         *====                                                    ====;
277         *====    DATA _NULL_ STEP PRODUCES THE MAIN REPORT    ==================;
278         *====                                                    ====;
279         DATA _NULL_;
280             FILE PRINT;
281                 SET ALL;
282         PUT   #3  @38 'RESEARCH CENTER' +1 CENTER @78 'PATIENT I.D.' +1 PATIENT
283               #6  @28 78*'-'
284               #8  @38 'VISIT'                 @48 'DATE OF VISIT'
285                   @66 'SYSTOLIC B.P.'         @84 'DIASTOLIC B.P.'
286               #9  @38 '------'                @48 '--------------'
287                   @66 '--------------'        @84 '--------------' / ;
288         %MPRINTIT
289        + IF DATE000 NE . THEN PUT @39 "000" @51 DATE000 @70 SYST000 3. @88 DIAS000 3. /  1-%mprintit
291        + IF DATE001 NE . THEN PUT @39 "001" @51 DATE001 @70 SYST001 3. @88 DIAS001 3. /  1-%mprintit
293        + IF DATE01A NE . THEN PUT @39 "01A" @51 DATE01A @70 SYST01A 3. @88 DIAS01A 3. /  1-%mprintit
295        + IF DATE01B NE . THEN PUT @39 "01B" @51 DATE01B @70 SYST01B 3. @88 DIAS01B 3. /  1-%mprintit
297        + IF DATE002 NE . THEN PUT @39 "002" @51 DATE002 @70 SYST002 3. @88 DIAS002 3. /  1-%mprintit
299        + IF DATE003 NE . THEN PUT @39 "003" @51 DATE003 @70 SYST003 3. @88 DIAS003 3. /  1-%mprintit
301        + IF DATE004 NE . THEN PUT @39 "004" @51 DATE004 @70 SYST004 3. @88 DIAS004 3. /  1-%mprintit
303        + IF DATE04Z NE . THEN PUT @39 "04Z" @51 DATE04Z @70 SYST04Z 3. @88 DIAS04Z 3. /  1-%mprintit
305        + IF DATE005 NE . THEN PUT @39 "005" @51 DATE005 @70 SYST005 3. @88 DIAS005 3. /  1-%mprintit
307        + IF DATE006 NE . THEN PUT @39 "006" @51 DATE006 @70 SYST006 3. @88 DIAS006 3. /  1-%mprintit
309        + IF DATE06A NE . THEN PUT @39 "06A" @51 DATE06A @70 SYST06A 3. @88 DIAS06A 3. /  1-%mprintit
311        + IF DATE007 NE . THEN PUT @39 "007" @51 DATE007 @70 SYST007 3. @88 DIAS007 3. /  1-%mprintit
313        + IF DATE008 NE . THEN PUT @39 "008" @51 DATE008 @70 SYST008 3. @88 DIAS008 3. /  1-%mprintit
315        + IF DATE009 NE . THEN PUT @39 "009" @51 DATE009 @70 SYST009 3. @88 DIAS009 3. /  1-%mprintit
317        + IF DATE09A NE . THEN PUT @39 "09A" @51 DATE09A @70 SYST09A 3. @88 DIAS09A 3. /  1-%mprintit
319        + IF DATE010 NE . THEN PUT @39 "010" @51 DATE010 @70 SYST010 3. @88 DIAS010 3. /  1-%mprintit
321        + IF DATE10A NE . THEN PUT @39 "10A" @51 DATE10A @70 SYST10A 3. @88 DIAS10A 3. /  1-%mprintit
324         PUT /  @28 78*'-' ;
325             RETURN;
```

Figure 3. Computer-generated code that produces the report in Figure 4.

```
**********************************************
*                                            *
*  ---   WE-R-DRUGS PHARMACEUTICALS CORP.  --- *
*           BP-DOWN PROTOCOL 46,750          *
*                                            *
**********************************************
```

SYSTOLIC & DIASTOLIC BLOOD PRESSURE

11MAR88

RESEARCH CENTER 3019 PATIENT I.D. 101

VISIT	DATE OF VISIT	SYSTOLIC B.P.	DIASTOLIC B.P.
000	12JAN85	167	90
001	14JAN85	168	91
01A	15JAN85	169	92
01B	19JAN85	170	93
002	28JAN85	171	94
003	11FEB85	172	95
004	25FEB85	173	96
04Z	26FEB85	174	97
005	11MAR85	150	89
006	25MAR85	149	88
06A	01APR85	148	87
007	08APR85	147	86
008	22APR85	146	85
009	06MAY85	145	84
09A	11MAY85	143	83
010	20MAY85	175	98
10A	27MAY85	176	99

Figure 4. Report that is produced both by programmer-written code in Figure 1 and computer-generated code in Figure 3.