A SAS® PROGRAM TO CREATE OUTPUT FILES USED FOR A RANDOMIZATION CODE DISTRIBUTION PROCEDURE

Joseph M. O'Connor, Bristol-Myers Squibb Company
Ute E. Schwiderski, Bristol-Myers Squibb Company
Joseph J. Andary, Boots Pharmaceuticals

ABSTRACT

In order to do randomizations for drug studies at Bristol-Myers Squibb, a CMS REXX program was developed which builds and executes a series of SAS® programs to produce the randomization output. Since this output needs to be packaged and distributed quickly and efficiently, a supplemental SAS® program was developed to perform this task. This paper addresses the features of the SAS® system that were used in the development of the packaging and distribution program. It runs in the SAS® 5.18 display manager and uses the SAS/FSP® procedure FSEDIT as a data entry interface. Applications of the SAS® macro language were found to be particularly useful in creating a relatively simple and flexible program.

INTRODUCTION

At Bristol-Myers Squibb, the randomization process can be divided into the following steps:

1. The randomization request form is filled out by the clinical monitor.
2. The information from the request form is typed into the computer.
3. The randomization hardcopies are generated.
4. The randomization is packaged and labeled by the statistician.
5. The randomization is delivered to the monitor who verifies its receipt.

For the generation of the randomization documents, two programs need to be run. The first is the main randomization program, which is written primarily in REXX. It builds and executes a series of SAS® programs which produce the following output:

1. Reports summarizing data input.
2. Reports showing patient numbers without treatments.
3. Reports showing patient numbers with treatments.
4. Packets of sealed individual patient envelopes.
5. Updates to password-protected library files.

Once these listings are produced, they must be correctly packaged, labeled, and sealed. Because of the sensitivity of the information in them and the need for randomizations to be delivered rapidly, a fast, accurate procedure for this step of the process is essential.

The second program, which is the packaging and distribution program, helps fill this need by generating the following output files:

1. Labels for the packages to be sent to the various departments.
2. A memo to the clinical monitor outlining the distribution procedure that is to be followed.
3. Checklists to guide the statistician in the correct packaging of the randomization.
4. Signature page verifying receipt of the randomization by the clinical monitor.

Before the packaging and distribution program was developed, these documents were produced manually, which was a time-consuming and error-prone process. The input, output, and development of this program will be the focus of this paper.

A SAMPLE RUN

The user enters the display manager, types %INC RANPACK, and hits the submit key. The program begins with a brief signon message and a prompt asking if the user wishes to continue. If the answer is yes, the first FSEDIT screen comes up as below:

SETUP INFORMATION SCREEN

1. **NUMBER OF STUDY SITES? (MAX. IS 5 AT A TIME)** ===> ?
2. **ANY SITES THAT ARE UNUSUAL?** ===> **N {default}**
3. **STD. CC LIST FOR MEMO?** ===> **Y {default}**
4. **WILL PK GET A COPY OF THE CODES?** ===> **N {default}**
5. **PULL IN OUTPUT FILE FROM PREVIOUS RUN?** ===> **N {default}**

The information entered in the first screen controls the appearance of the other displays (some of them will simply be bypassed if they are not needed). A detailed explanation for each of the variables that appear here is in Appendix 1.

The main information screen, which appears next, is shown below:

MAIN INFORMATION SCREEN

DRUG: _____

PROTOCOL NUMBER: _____

FIRST SITE NUMBER: 001 (default value)

PATIENT NUMBERS: __ **TO** __

STUDY TITLE: _____
_____
_____

REQUESTING MONITOR: _____

STATISTICIANS: _____

INVESTIGATORS: **1.** _____
2. _____

The number of fields available for the investigator names is a function of the number of study sites used for a protocol, which is specified in the setup screen. So if there were 5 sites instead of two, 5 investigator name fields would appear.

Two other screens in this program appear only if needed (inclusion controlled by the setup screen). One of them is for the entry of information specific to a particular site. The other one allows changes to the standard memo distribution (CC) list.

If additional screens are not needed (as is most often the case) the user is prompted by one final question before leaving the program:

What documents need to be generated?

(A) Labels **(D) All of the above.**
(B) Memos **(E) Exit the program.**
(C) Checklists

SAMPLE OUTPUT

In Appendix 2 are some examples of the documents produced. They include a sample label, memo, and checklist.

OVERVIEW OF HOW THE SYSTEM OPERATES

1. The packaging program consists of one main program along with several subordinate programs. The main program provides the data entry interface and calls the subordinate programs when needed. The subordinate programs generate the output files requested by the main program.

2. The data entry displays with PROC FSEDIT were individually customized using the screen modification and field attribute options. This customization was a convenient method of setting up error checks as well as allowing for more readable displays.

3. After the data are entered through the screens, the information is passed from the data step to macro variables using the SYMPUT function. When the subordinate programs are invoked (the memo, label, and checklist generators) these macro variables are further modified and used as needed.

IMPORTANT DETAILS

The use of CALL SYMPUT, macro string-handling functions, and macro control structures provide the key to the program's relative simplicity and flexibility. The list below gives the most important examples of how these features were used and shows what problems were solved with them.

1. Use of %SUBSTR function with %EVAL -
   The data sets used need different variable lists under different circumstances. For example, in the main information screen the number of investigator variables can vary from 1 to 5. Instead of creating 5 different data sets (one for each possible screen) the following lines of code were used:

```
%LET INVNC=
INV1 $25. INV2 $25. INV3 $25. INV4 $25. INV5 $25.
;
%LET VALX=%EVAL(10*&N_SITES);
%LET INVST=%SUBSTR(&INVNC,1,&VALX);

DATA RANDOM; INPUT {OTHER VAR.} &INVST;
```

2. CALL SYMPUT with PROC TRANSPOSE -
   The SYMPUT function can only work with two variables at a time, while an FSEDIT screen shows many variables at a time (of one observation). This problem can be solved simply by using PROC TRANSPOSE before invoking CALL SYMPUT. The many variables of one observation are then transformed into two variables with many observations. One variable, called _NAME_, holds all the names of the former variables, the other, COL1, has all the data entered by the user.

3. Iterative execution with the %DO statement -
   The memos and labels targeted for each study site need the name of the site investigator. The label and memo macros are always executed with the %DO statement as shown below:

```
%DO I=1 %TO &N_SITES;
   %LABEL {OR %MEMO};
%END;
```

Inside these macros is the token &&INV&I where the investigator name should appear. The macro processor makes two passes at this token to resolve it. At I = 1, &&INV&I first resolves to &INV1, which then resolves to the name of the investigator at the first study site.

4. Use of the %INDEX function with %SUBSTR -
Although the study sites within a protocol generally only differ by site number (they are usually consecutively numbered) and investigator names, there are times when this is not the case. Sometimes each site has a separate monitor; other times the number of patients might be different. If the user has indicated on the setup screen that site-specific information is needed, a supplemental screen will appear after the main information display is exited. An example is shown below:

SUPPLEMENTAL INFORMATION SCREEN

TAG CODE LEGEND
 # - NONCONSECUTIVE SITE NUMBERS
@ - FIRST PATIENT NUMBER
$ - LAST PATIENT NUMBER

MONITOR NAMES FOR EACH SITE:
(AND TAG CODES IF APPLICABLE)

1. Michael Manning = _____
2. Mary Bates = #003 $90_____

In this example, the = signs activate the lines of code with the %INDEX function, (they must be present in the first monitor field whether or not anything is being changed there) the $ sign overrides the upper bound of the patient numbers specified in the main information screen, and the # character overrides the consecutive numbering of study sites. So in the case of the site where Mary Bates is the monitor, the #003 changes the site number from its default of 002 to 003, while the $90 changes the patient number upper bound from 60 to 90.

The %INDEX function is able to scan for these character codes in the macro variables, and if present, return their location on the string. With the %SUBSTR function, the text immediately next to these codes is able to be parsed and placed in another macro variable. A complete example of how this is done is in Appendix 3.

5. %SCAN function -
This function is used in the program to separate last names from full names. In the checklist documents only the last names of the site monitor(s) appear next to the investigator names, while the memos and labels show the full names. This is done using the syntax:

```
%LET LSTNAME=%SCAN(&FLLNAME,2);
```

FINAL COMMENTS

1. The ease of use and flexibility of PROC FSEDIT compares well with other menuing systems such as ISPF. Even though it is designed primarily for interactive SAS® data set modification, it works well as an application front-end when combined with PROC TRANSPOSE.

2. Because the main program is able to modify the lists of variables present in its data steps, a different FSEDIT screen is needed for each possible list. For example, when the data step for the main information screen has one investigator variable (INV1), a different screen is called than when there are two investigators (INV1 and INV2). This feature may seem to add unneccessary complexity to the program. However, the creation of additional screens was an extremely simple task once the basic template was set up.

3. The use of built-in macro functions kept the size of the program relatively small (about 1000 lines of code) and enhanced its flexibility. They also reduced the number of user-defined macros considerably.

4. Most of the macro variables were globally defined so that they could easily be used in any place of the program.

REFERENCES

Aster, R. and Seidman, R. (1991). **Professional SAS® Programming Secrets.** Windcrest Books, Blue Ridge Summit, Pennsylvania.

Phillips, Jeff, **"Designing Macro-Based Systems"**, SUGI, 1985.

SAS Institute Inc., 1985. **SAS® User's Guide: Basics, Version 5 Edition.** SAS Institute Inc. Cary, North Carolina.

SAS Institute Inc., 1985. **SAS/FSP® User's Guide, Version 5 Edition.** SAS Institute Inc. Cary, North Carolina.

SAS Institute Inc., 1985. **SAS® Guide to Macro Processing, Version 5 Edition.** SAS Institute Inc. Cary, North Carolina.

SAS Institute Inc., 1985. **SAS Views®: SAS® Macro Language, Version 5 Edition.** SAS Institute Inc. Cary, North Carolina.

Author Contact:
Joseph M. O'Connor
Dept. 703/91
5 Research Parkway
P.O. Box 5100
Wallingford, CT 06492-7660
(203) 949-3572

APPENDIX 1
EXPLANATION OF VARIABLES IN SETUP DISPLAY

1. Number of sites? - study sites at BMS have a protocol number such as PL105-075 and a 3 digit site number. Therefore, site 1 of the PL105-075 protocol would be labeled PL105-075-001. The program can generate documents for up to 5 sites at a time.

2. Any sites that are unusual? - study sites for the same protocol are generally uniform with respect to numbers of patients. They are also usually numbered consecutively and have the same clinical monitor overseeing them. If any of the above conditions is not true, the user should answer the question with a Y. This will signal the program to call up the supplemental information screen.

3. Std. CC list for the memo? - if people need to be added to the standard distribution list or the list itself has to be changed, a N answer to this question will activate the distribution list display.

4. Will PK get a copy of the codes? - if the study protocol requires that plasma samples be taken, a copy of the randomization must be sent to the Pharmacokinetics group. This group needs to be on the memo distribution list, and it needs an appropriately labeled package.

5. Pull in output file from the previous run? - each time the program runs, it stores the information entered by the user in an external file which can be retrieved by the program. This feature is useful when there are more than 5 study sites indicated in a randomization request.

APPENDIX 2
SAMPLE DOCUMENTS PRODUCED BY THE PROGRAM

1. Sample label

A DOUBLE-BLIND TRIAL OF PLATYKIN VS. PLACEBO IN THE TREATMENT OF PATIENTS WITH BIPOLAR DISORDER

STUDY IDENTIFICATION

STUDY: PL105-075-001 (KENNISON)
MONITOR: MICHAEL MANNING
RANDOMIZATION FOR 60 PATIENTS: 1-60
DATE: FEBRUARY 23, 1992

SEALED RANDOMIZATION CODE

LABEL IS COPY 1 OF 5: CENTRAL FILES
FULL RANDOMIZATION CODE WITHIN ENVELOPE

APPENDIX 2 (CONT.)

2. Sample memo

From: Sam Schmidt and Joe O'Connor

Date: FEBRUARY 23, 1992

To: Michael Manning

RANDOMIZATION FOR PLATYKIN PL105-075-001 (KENNISON)

Enclosed you will find copies of the randomizations that you requested for the Platykin study PL105-075-001. For this site, 60 patient numbers from 1 to 60 have been provided.

.
.
.
.

jo
Enclosures
CC: Steven Baker Joan Johnston PK Group

3. Sample checklist

Randomization Checkoff Sheet # 3 - monitor's copy

Protocol: PL105-075

A Double-Blind Trial Of Platykin Vs. Placebo In The Treatment of Patients With Bipolar Disorder

Requesting monitor: Michael Manning
Approving statisticians: Sam Schmidt and Joe O'Connor

Investigators	Monitors	Sites	Patient #s
Kennison	Manning	001	1 - 60
O'Donnell	Bates	003	1 - 90

	001	003
(1) Central Files Envelope	___	___
(2) Monitor - Individual patient envelopes (sealed in a large confidential envelope).	___	___
(3) Investigator - Individual patient envelopes (sealed in a large confidential envelope).	___	___
(4) Drug Packaging Group Envelope	___	___
(5) PK Group Envelope	___	___

APPENDIX 3
PROGRAM SOURCE CODE USING THE %INDEX FUNCTION

```
%DO I=1 TO &N_SITES; /* COUNT UP NUMBER OF SITES */

*************************************************
*  %INDEX FUNCTION SCANS FOR A $ SIGN. IF IT    *
*  IS FOUND, IT RETURNS THE POSITION OF IT      *
*  IN THE STRING AND CALLS %CCODES.             *
*************************************************;
   %IF %INDEX(&&MON&I,$) NE 0 %THEN %DO;
     %CCODES(TRIGGER=$, ROOT=PL);
   %END;

*************************************************
*  IF THE %INDEX FUNCTION RETURNS A 0, IT       *
*  ACTIVATES THE %STNDRD MACRO.                 *
*                                               *
*************************************************;
   %ELSE %DO;
     %STNDRD(MACVAR=&PTLAST, ROOT=PL);
   %END;
%END; /* CLOSE OF %DO LOOP */

*************************************************
*  THE CCODES MACRO DOES THE FOLLOWING:         *
*    1. FINDS THE POSITION OF THE               *
*       TAG CODE AND ADDS 1 TO IT.              *
*    2. TAKES THE SUBSTR OF THE MACRO           *
*       VARIABLE FROM ONE SPACE TO THE          *
*       RIGHT OF THE TAG TO 3 SPACES TO         *
*       THE RIGHT OF IT.                        *
*    3. THE RESULT IS PLUGGED INTO A MACRO      *
*       VARIABLE THAT CORRESPONDS TO THE        *
*       NUMBER OF EXECUTIONS MADE BY THE        *
*       %DO LOOP. SO IF I = 3, &&MON&I          *
*       BECOMES &MON3. IF THE TAG CODE IS       *
*       THE $ SIGN AND THE ROOT IS PL, THE      *
*       MACRO VARIABLE THAT RESULTS IS          *
*       &PL3. &PL3 WILL THEN HAVE THE LAST      *
*       PATIENT NUMBER FOR THE THIRD SITE.      *
*                                               *
*************************************************;

%MACRO CCODES(TRIGGER=,ROOT=);
/* MON IS SHORT FOR MONITOR VARIABLE, ROOT IS A */
/* MACRO VARIABLE NAME LESS ITS NUMBER */
   %GLOBAL &ROOT&I;
   %LET POS1=%INDEX(&&MON&I,&TRIGGER);
   %LET POS2=%EVAL(&POS1+1);
   %LET &ROOT&I=%SUBSTR(&&MON&I,&POS2,3);
%MEND CCODES;

*************************************************
*   THE STNDRD MACRO TRANSFERS THE VALUE OF     *
*  ONE MACRO VARIABLE TO ANOTHER. SO IF THE     *
*  SPECIFIED TAG CODE IS THE $ SIGN, I=3, AND   *
*  A $ SIGN IS NOT FOUND IN &MON3, THE VALUE    *
*  OF &PTLAST. IS TRANSFERRED TO &PL3. &PTLAST  *
*  HOLDS THE DEFAULT VALUE FOR LAST PATIENT     *
*  NUMBER FOR A SITE.                           *
*************************************************;
%MACRO STNDRD(ROOT=,MACVAR);
   %GLOBAL &ROOT&I;
   %LET &ROOT&I=&MACVAR);
%MEND STNDRD;
```