COMBINING SAS/FSP AND THE NEW MACRO FACILITY TO CREATE A FULL SCREEN INTERFACE TO A SAS PROGRAM

Mark J. Holdbrook, Syntex Research

ABSTRACT

Presented here is a technique for creating generalized report writing software which uses SAS/FSP as a user interface. The user, with little or no knowledge of SAS, can create a report quality listing or table from a SAS dataset by following the instructions provided by a SAS/FSP screen dataset.

SAS/FSP is used to create a "control" dataset (or datasets). The information from this dataset can be passed to macro variables with the CALL SYMPUT function. Additional control information can be obtained from the input dataset which changes too difficult and passed to macro variables as well. SAS code can then utilize these macro variables to adapt to variations in the input dataset and/or final report format.

Such a software system may have several applications. It could be utilized by novice or inexperienced SAS users to meet specific data processing needs, such as generating a report or listing. A generalized software system could alleviate the need to repeatedly edit SAS code to adapt to slightly different or new situations. This would reduce debugging time and speed turn around.

INTRODUCTION

The biostatistics department of a pharmaceutical company regularly produces statistical reports containing tables, analyses and listings of clinical data. Although the studies may vary as to length, number of visits, compounds being tested, statistical design and the like many of the tables and listings are very similar. The SAS products have proved useful both for producing report quality tables and listings and performing statistical analyses.

The problem was each time a new study analysis was begun we would have to edit previous code used for analysis to accommodate the design differences in the new study. This process was error prone, often involving several attempts to make all the necessary corrections. At times it was simpler to write a whole new program from scratch. This was because the algorithm used in the earlier program made the necessary changes too difficult, even though the desired output was very similar.

SOLUTION

The proposed solution was to write a SAS program which using SAS macros would be flexible enough to accommodate most situations by editing a few macro variables at the beginning of the program. The solution was extended even further to allow the values of the necessary macro variables to be input from a SAS dataset which could be edited by SAS/FSP. It was hoped that by using SAS/FSP to edit the dataset that sufficient information could be contained on the edit screen to allow non-programmers as well as experienced SAS users to run the program. This would permit the development of software which the clinicians could use to monitor ongoing studies.

The program would begin by using proc FSEDIT to edit a control dataset which would control the program processing. The screen(s) for FSEDIT could be designed in advance to provide the necessary instructions to edit the data and checks on the values entered. The variables in the control dataset would then be put into macro variables using the CALL SYMPUT routine. Next the input dataset which is to be processed could be polled for certain key parameters necessary for the program. These values could be entered into macro variables as well. The remainder of the program would then utilize these macro variables to generate the appropriate SAS code.

EXAMPLE

This example is diagrammed in figure 1 which shows the user input and programming steps involved in creating a laboratory listing for a clinical study. The program outlined here is a simplified version of one we actually use to produce these listings.

The objective of this program will be to produce a report quality listing of every patient's laboratory data for a given test on a single page. The program will require as input a SAS dataset, LABDATA, which contains the raw data to be reported, see listing 1, and two control datasets, PARAM and LABVAR. Each record in the LABDATA dataset will contain certain identifying information, patient number, visit and treatment, and the lab data for the given patient at that visit. We will assume the SAS dataset containing the raw data already exists, in our environment this is the dataset automatically created when the data is extracted from our database. The user will be required to input the SAS dataset names of the labtests to be reported.

The user would begin by logging on to TSO and executing a simple clist. The clist contains a short SAS program which requests the DSN of the SAS library containing the input dataset and allocates the file. The program then checks for the existence of the control
datasets in the library using a variation of the technique published by Donald J. Henderson and David L. Kuhn. If the datasets do not already exist they are created. The PARAM dataset is then edited using proc FSEDIT and a predefined screen dataset SPARAM (figure 2). The dataset will contain one record with the study number and the name of the input dataset containing the lab data. The same sequence of steps is then repeated for the LABVAR dataset which contains the SAS variable name for one labtest per record (figure 3).

Next the code to create the table is executed. The program begins by reading the input file name from the PARAM dataset and placing the name in a macro variable. The LABVAR file is then read for the SAS variable names of the laboratory tests to be reported and these names are placed in macro variables. Listing 2 shows a sample of SAS code which could be used to create the macro variables with containing the test names. The input dataset, LABDATA, is then read for controlling information, such as the maximum number of visits present, and that information is stored in macro variables as well (listing 3). The remainder of the program utilizes the control information contained in the macro variables to print the listing. Listing 4 shows a sample page from the output.

CONCLUSION

The new macro facility has allowed the creation of software which can handle a wider variety of situations than before without having to revise the program code. Since altering the code is a more time consuming and error-prone process than modifying the value of a few macro variables the use of macros has allowed an even further increase in efficiency.

Using SAS/FSP and the new macro facility together allows the creation of an interface to a SAS program which is simple enough to be used with little or no knowledge of SAS. The result of this method is similar to methods suggested by Roger D. Muller (2) and by C. David Henderson and Roger D. Muller (3) using SAS/FSP. These systems, which are sometimes referred to as "application generators", write SAS code to an external file and then use %INCLUDE to include the file in the SAS input stream. Although it is beyond the scope of this paper to provide an in depth comparison of these two techniques, the similarities do invite certain comparisons. With macros there is no need for an external file, the code may be more readable and it is easier to deal with control datasets which have multiple observations, such as the LABVAR dataset in the example. The disadvantage to using macros is they can be very difficult to debug and sometimes seem to behave in an almost incomprehensible fashion.

I would like to acknowledge Dr. Martin Rosenberg's assistance which helped to formulate this concept and see it through to completion.

REFERENCES


SAS and SAS/FSP are registered trademarks of SAS Institute Inc., Cary, NC, USA.
Input SAS Dataset as Extracted from the Database

<table>
<thead>
<tr>
<th>OBS</th>
<th>PT</th>
<th>VISIT</th>
<th>CHOLESTEROL</th>
<th>CHOLU</th>
<th>TRIGLYCERIDES</th>
<th>TRIGU</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>229</td>
<td>MG/DL</td>
<td>105</td>
<td>MG/DL</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>232</td>
<td>MG/DL</td>
<td>122</td>
<td>MG/DL</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>3</td>
<td>277</td>
<td>MG/DL</td>
<td>176</td>
<td>MG/DL</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>4</td>
<td>223</td>
<td>MG/DL</td>
<td>131</td>
<td>MG/DL</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>1</td>
<td>296</td>
<td>MG/DL</td>
<td>157</td>
<td>MG/DL</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>2</td>
<td>243</td>
<td>MG/DL</td>
<td>114</td>
<td>MG/DL</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>3</td>
<td>256</td>
<td>MG/DL</td>
<td>155</td>
<td>MG/DL</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>4</td>
<td>232</td>
<td>MG/DL</td>
<td>132</td>
<td>MG/DL</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>1</td>
<td>276</td>
<td>MG/DL</td>
<td>166</td>
<td>MG/DL</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>3</td>
<td>224</td>
<td>MG/DL</td>
<td>127</td>
<td>MG/DL</td>
</tr>
<tr>
<td>11</td>
<td>3</td>
<td>4</td>
<td>259</td>
<td>MG/DL</td>
<td>188</td>
<td>MG/DL</td>
</tr>
</tbody>
</table>

Listing 1

SAS Code to Pass Information from the Control Dataset to Macro Variables

* The laboratory variable names which are contained in the VAR variable in the LABVAR dataset are passed to the macro variables VAR1, VAR2, ..., VARn. The total number of tests to be reported is contained in the macro variable NTOTAL.

```
DATA _NULL_;  
    SET SASDATA.LABVAR END=EOF;  
    IF VAR NE '';  
    RETAIN I 0 ;  
    I = I + 1;  
    CALL SYMPUT('NVAR',I,VAR);  
    IF EOF  
    THEN CALL SYMPUT('NTOTAL',I);  
RUN ;
```

Listing 2

SAS Code to Read Control Information from Input Dataset

* The input dataset, &INDATA, is sorted by VISIT. Each VISIT number is put in a macro variable VIS1, VIS2, ..., VISn. The total number of different visits is put in the macro variable MAXVIS.

```
PROC SORT DATA=&INDATA OUT=TEMP ; BY VISIT ;  
DATA _NULL_;  
    SET TEMP END=EOF ; BY VISIT ;  
    IF LAST.VISIT * Do once for each visit  
    RETAIN I 0 ;  
    * I is used as a counter  
    I = I + 1;  
    CALL SYMPUT('VIS',I,LEFT(VISIT));  
    IF EOF  
    THEN CALL SYMPUT('MAXVIS',I);  
RUN ;
```

Listing 3

Sample Page from the Final Output

<table>
<thead>
<tr>
<th>VISIT</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>PATIENT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>229.00</td>
<td>232.00</td>
<td>277.00</td>
<td>223.00</td>
</tr>
<tr>
<td>2</td>
<td>296.00</td>
<td>243.00</td>
<td>256.00</td>
<td>232.00</td>
</tr>
<tr>
<td>3</td>
<td>276.00</td>
<td>224.00</td>
<td>259.00</td>
<td>188.00</td>
</tr>
</tbody>
</table>

SOURCE: BIOANALYSIS MXH$1365 (O80CT84)

Listing 4
PROGRAMMING STEPS TO CREATE A LAB TABLE

_INPUT DATASET
ILABDATA1

~SSDATATO
I MAC'ROVARIABLES I

~J

CODE FOR PROCESSING
INPUT DATASET
(INVAR1, INVAR2,...)

FINAL TABLE

Figure 1
EDIT SAS DATA SET: SASDATA.PARAM
COMMAND==>
|SCREEN 1
|OBS 1

Enter the study number: 

Enter the name of the SAS dataset which contains the lab data:

Press PF3(PF15) to return to the FSP menu then press PF3(PF15) again to continue

Figure 2
EDIT SAS DATA SET: SASDATA.LABVAR
COMMAND==>
|SCREEN 1
|OBS 1

Enter the SAS name for the lab test: 

Enter the SAS name for the lab test units: 

Use PF6(PF18) to DUPLICATE a record.
Use PF9(PF21) to create a NEW record.
Press PF3(PF15) to return to the FSP menu then press PF3(PF15) again to continue

Figure 3