SIMPLE CONSTRUCTION AND MODIFICATION OF COMPUTER GENERATED TABLES

Michael A. Litzsinger, Quintiles, Inc.
Steven A. Wright, Ph.D., Quintiles, Inc.

ABSTRACT

Computer generated tables have been traditionally constructed by using PUT statement programming. Generally this process produces code that is difficult to change and not particularly motivating to write. SAS procedures such as TABULATE and QPRINT are somewhat flexible, but not flexible enough to produce tables with specific layouts.

This paper describes a Template Generated Table system that eliminates the need for PUT statement coding of tables. This simple but profound technique consists of three separate tasks. They include construction of a table shell or template, preprocessing the data to obtain the needed table values (only discussed here in a limited scope), and combining the template and data into the final report table using a "black box" macro. The template looks similar to the final table, except for data placeholders.

This technique has several advantages over PUT statement programming. Titles, footnotes, and other static text information are easy to modify, as are the row and column placement of the data. The original set-up and changes are made through full screen editing of the template file, saving a lot of time, and avoiding complex PUT statement coding. Our staff has quickly learned the template concept of producing computer generated report tables, and have been reluctant to return to PUT statement coding because of the ease and flexibility of the Template Generated Table system.

SUBJECT PERSPECTIVE

To give some perspective, here are some of the available methods to produce customized report tables using SAS.

The most basic is the abstraction method where statistics are removed from SAS procedural output and compiled into a report table by hand in a word processing environment. This method has no additional costs for the computer generation of the table, since it is merely created on a word processor. But the disadvantages include the possibility of transcription errors and relatively constant costs for reruns or similar tables.

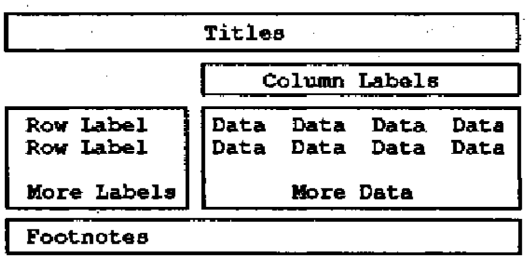
To offset these liabilities, computer generated tables are often used. One way to get custom report tables, though only for limited layouts, is to simply take the output from SAS procedures. Standard manipulations with TITLE, FOOTNOTE, and OPTIONS statements as well as with the data itself can result in a format that can satisfy the demands of many applications. SAS procedures produce reports in a relatively standard format. This often forces decisions on whether the format requested is important enough to warrant the difficulty involved in producing it. There are just some things that are not possible from SAS procedures currently, such as placing vertical lines through the data, and labels greater than 40 characters long. Tricks such as putting column headings in TITLE statements might just not be worth the finagling to achieve the results.

When it is necessary to have specific report tables, the standard method is to use PUT statements within a DATA step to create the customized layout desired. Data may be passed from SAS procedures or may be processed within the DATA step just prior to it being PUT out. However, changes to the report layout can prove to be monumental programming tasks. Modifications must be made and then run before it can be determined whether the programming fix was correct. Additional testing is required to determine whether or not a coding change really works as expected. Complex PUT statement code that is poorly documented can also be almost unreadable when it is encountered by another programmer, or if it has been set aside for awhile. It is hard to make a correction to something that you do not understand. The detailed coding required and the potential difficulty of making changes can make PUT statement programming a cumbersome tool with which to create customized report tables.

DEVELOPMENT OF THE TEMPLATE GENERATED TABLE SYSTEM

Through our use of a technique to produce listings, "A SAS Macro System for Customized Report Writing", presented by John King at SUGI 13, we were introduced to some of the advantages of using a template to define the report layout. A template is an ASCII type file (or flat file) that represents the table specification, line for line and column for column. The table layout is input by simply scanning the template. This is a method that defines the table layout without having to code PUT statements.

Example of Template Layout

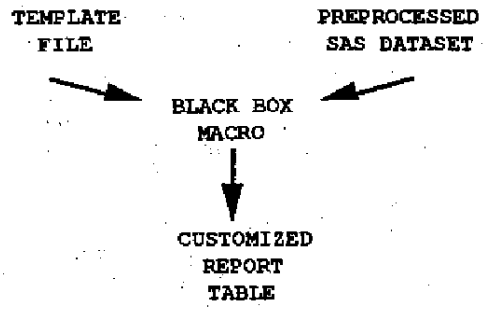


The goal was to develop a methodology for computer generated tables that was easy to use, could do most of our tasks, utilized a full screen construction of the template, and could be executed as a black box. Development of this tool was motivated by the need for specialized report

tables to satisfy a spectrum of clients. Using templates means that changes can be made quickly. A black box approach allows unspecified options to be automatically defined as default, easing access by all levels of SAS programmers, but still allowing versatile performance.

The solution was to use a representation of the desired table to instruct the computer to write the necessary PUT statements automatically! A template that resembled the required table exactly is designed using a full screen text editor. The locations of each data item are marked by a placeholder instead of typing in the actual data values. Separately, the data is preprocessed in its final format to be inserted within the template. Then a black box macro merges the template and data, and PUTs them out as the customized report table!

The Process Summarized



THE TEMPLATE

Since the template is the guide that is used to display the format of the customized report table, its representation is duplicated to create the table. It defines the number of columns and lines that the table will have. The only requirement is that one unique character must be set aside to mark each data location. Using a template to represent the layout has several advantages. It avoids the counterintuitive logic that is usually present with some other methods, that you start with a specification that shows exactly what the report table ought to look like, and then decode this with PUT statement programming to try to get back a table that matches the specification. Setting up the desired table format can also be parcelled off to someone else to create on a word processor, and it does not require any knowledge of the SAS programming language. The template can be copied from the word processed document itself as long as it ends up in an ASCII format. Most importantly, the template is easily edited in a full screen environment. Here, what you see is what you get: to modify the table, just go in and edit it. Different layouts can be tried out here in the editing phase, since you can see the results immediately.

PREPROCESSED DATA

The preprocessed data needs to be structured so it corresponds to the data placeholders. Although the scope of this presentation is not to focus on the handling the SAS data in the preprocessing phase, I would like to note that the SAS data must be in its final format with all data of type character. This is necessary because data within a column may be both numeric and character. Putting the data in its final form in this step allows all manipulation to take place in one logical place. The very term preprocessed implies that the data has been prepared and is now ready for placement into the report table. It encourages a modular approach to the preprocessed results, allowing verification of the data prior to the creation of the table. And unless you are doing data calculations within the DATA step that PUTs out the table, the only restriction that you will notice is the requirement to put your data values in their final format in a prior step.

TEMPLATE GENERATED TABLE MACRO

Conceptually, the programming tasks of this template and data compilation macro are quite fundamental, and the actual macro only needs to be a handful of short DATA and PROC steps. The only required input parameters that the macro needs are the name of the template file and the name of the preprocessed SAS dataset. The tasks performed by this macro are no mystery. The template is scanned, and the data value locations are read and stored as SAS variables. Since the data placeholders are now stored away, they can be removed from the template (set to blanks) as the static text is read in as a SAS variable. Now that the pertinent table information is in a SAS file, it can be MERGED together with the preprocessed SAS data by line, inserting the preprocessed SAS data into each line and column location formerly specified by the placeholders. Then every line is written out individually, simulating the writing of PUT statements. The SUBSTR function can be used to perform the character insertion.

EXAMPLES

Table 1 is a simple example of the Template Generated Table process:

In the template, the symbol * has been reserved to indicate where each data item will be located. This will be referred to as the location character or the data placeholder. Note the designation of the titles, row and column labels as text, and the 2x2 data area.

Example 1 Template

```
Table 1
Subjects That Quit Smoking
=============================
          Treatment Group
          ---------------
          Drug     Placebo
=============================
Males     *        *
Females   *        *
=============================
```

PROC PRINT output of the preprocessed SAS dataset is listed below. The data to be inserted corresponds to the 2x2 layout of the template.

Example 1 Preprocessed SAS Data

THE PREPROCESSED SAS DATASET

OBS	COL1	COL2
1	16/52(31%)	14/63(22%)
2	13/48(27%)	6/37(16%)

To combine the template and data, the "black box" macro is called, with its two required parameters.

Example 1 Macro Call

```
%BLACKBOX ( INTMPLAT = Template File,
            INSASDSN = Preprocessed SAS
                       Dataset )
```

The resulting Template Generated Report Table is below. The data placeholders * are now replaced by data.

Example 1 Report Table

Table 1
Subjects That Quit Smoking

	Treatment Group	
	Drug	Placebo
Males	16/52(31%)	14/63(22%)
Females	13/48(27%)	6/37(16%)

Table 2 is a slightly more complex table. It demonstrates two more features that are often called for in report tables. It shows the titles being centered within the table (indicated by the centering symbol !). It also illustrates a line overprinting the prior line (indicated by the overprinting symbol \) to allow underlining. The centering and overprinting symbols can be placed anywhere on the line. At processing time they will set respective flags, and each will be replaced by a blank. Simple full screen editing is demonstrated by the simple movement of the column labels. "Treatment Group" and "Drug" were shifted right 3 columns, and "Placebo" was shifted right 2 columns as compared to the first example.

Example 2 Template

```
! Table 2
! Subjects That Quit Smoking
                Treatment Group
\             Drug          Placebo
Males         *             *
Females       *             *
```

For example 2 the same data will be used as was in example 1.

Example 2 Preprocessed SAS Data

THE PREPROCESSED SAS DATASET

OBS	COL1	COL2
1	16/52(31%)	14/63(22%)
2	13/48(27%)	6/37(16%)

Here is the "black box" macro call. An additional parameter specifies to write out the report table to a permanent file, instead of just writing it to the print log.

Example 2 Macro Call

```
%BLACKBOX ( INTMPLAT = Template File,
            INSASDSN = Preprocessed SAS
                       Dataset,
            OUTFILE  = Saved Table File )
```

Here is the report table featuring line centering and overprinting, saving the report table to an external file.

Example 2 Report Table

Table 2
Subjects That Quit Smoking

	Treatment Group	
	Drug	Placebo
Males	16/52(31%)	14/63(22%)
Females	13/48(27%)	6/37(16%)

Table 3 shows some additional capabilities of a computer generated table using the Template Generated Table system. The template has 4 lines for data and a maximum of 3 placeholders indicated on a line. The location symbol is changed and the symbol @ is used so that the symbol * can be used as text on the title and footnote lines. An extra title with data has been added and the table number will now be indicated by data. The row labels that had previously displayed "Males" and "Females" will also be filled by data.

Example 3 Template

```
! Table @
! Reported: @
! Subjects That Quit Smoking*
                Treatment Group
\             Drug          Placebo
@             @             @
@             @             @
*Taken from yearly questionnaire
```

Here is the PROC PRINT output of the 4x3 preprocessed SAS dataset to be inserted into the template. The variables COL2 and COL3 are blank for observations 1 and 2. These values are to be inserted into lines that only have 1 placeholder so it follows that they should also be blank.

Example 3 Preprocessed SAS Data

THE PREPROCESSED SAS DATASET

OBS	COL1	COL2	COL3
1	3		
2	02/20/91		
3	U.S.	25/85(29%)	20/85(24%)
4	U.S.S.R.	3/15(20%)	1/15(7%)

Here is the macro call. The location character specified is the symbol @ in agreement with our template.

Example 3 Macro Call

```
%BLACKBOX ( INTMPLAT = Template File,
            INSASDSN = Preprocessed SAS
                       Dataset,
            OUTFILE  = Saved Table File,
            LOCCHAR  = @ )
```

This example shows a generic template that can be used to produce multiple reports by merely altering the input preprocessed SAS data. Note here how the symbol * remains as static text in the title and footnote.

Example 3 Report Table

Table 3
Reported: 02/20/91
Subjects That Quit Smoking*

	Treatment Group	
	Drug	Placebo
U.S.	25/85(29%)	20/85(24%)
U.S.S.R.	3/15(20%)	1/15(7%)

*Taken from yearly questionnaire

Another important feature of the Template Generated Table macro is a self-documenting summary of the tasks performed, written to the SAS log. It is important to identify the input template file and preprocessed SAS dataset, as well as to indicate whether the report table created was written to the print log or to a specific external file. Any messages to aid debugging are also appropriate, and so are user defined errors and warnings. It is critical that the data marked on the template and the data in the preprocessed dataset correspond exactly. One extra template data line without a matching data observation would cause a mismatch, and vice-versa.

Example of Documentation Written to the SAS Log

```
NOTE: The input template C:\TEMPLATE\TABL3.T
      has 4 rows and 3 columns designated
      for data insertion.
NOTE: The input dataset MYLIB.TABL3
      has 4 observations and 3 variables.
NOTE: The output table is 12 rows by 32 cols
      and the page is 60 rows by 132 cols.
      It's written to C:\TABLELIB\TABL3.TBL.
```

Another relevant task is to set explicit linesize and pagesize values and have the table center itself on the page automatically. With the limitations of typeset and space, it is somewhat difficult to show here how beneficial automatic page adjustments are.

Tables 4.1 and 4.2 (on the next two pages) are examples of real world applications. These examples better show the true benefits of the Template Generated Table process.

CONCLUSION

Spending some up front time in development of the Template Generated Table macro really pays off. What you end up with is a powerful but easy to use application that programmers at your site can use repeatedly. The process of Template Generated Tables allows full screen editing of the table specification, offers modifications to be made in a quick straightforward fashion, and does away with PUT statement coding. The actual construction of the report table is done automatically, based on a simple picture representation of the table. This way, you can spend your time verifying the data, not the program that writes it out.

REFERENCE

King, John Henry (1988), "A SAS Macro System for Customized Report Writing", Proceedings of the Thirteenth Annual SAS Users Group International Conference, Cary, NC: SAS Institute Inc., 853-858.

ACKNOWLEDGEMENTS

Special thanks to Kitty Moses of Quintiles, Inc. for her developmental input and her testing of the TGT system.

CONTACT ADDRESS

Michael A. Litzsinger
Steven A. Wright
Quintiles, Inc.
P. O. Box 13979
Research Triangle Park, NC 27709-3979

Example 4.1 Template

Protocol XYZ Site=#

Table 4.1 --- Evaluation of Treadmill Regimen
Stage One Times (in seconds)

	Treatment Group			P-Values Comparing Treatment Groups@		
	Drug X	Drug Y	Drug Z	X vs Y	X vs Z	Y vs Z
Total Patients	#	#	#			
Baseline Visit						
n	#	#	#	#	#	#
Mean	#	#	#			
Std.Dev.	#	#	#			
Min.,Max.	#	#	#			
Final Study Visit						
n	#	#	#	#	#	#
Mean	#	#	#			
Std.Dev.	#	#	#			
Min.,Max.	#	#	#			

@ P-Values are adjusted by study site

Example 4.1 Preprocessed SAS Data

OBS	COL1	COL2	COL3	COL4	COL5	COL6
1	NC400					
2	80	75	76			
3	76	71	66	.651	.033*	.026
4	555	650	700			
5	187	200	245			
6	182,932	108,1040	246,1458			
7	59	60	56	.300	.007**	.213
8	698	751	770			
9	199	219	244			
10	275,1007	268,987	344,1472			

Example 4.1 Report Table

Protocol XYZ Site=NC400

Table 4.1 --- Evaluation of Treadmill Regimen
Stage One Times (in seconds)

	Treatment Group			P-Values Comparing Treatment Groups@		
	Drug X	Drug Y	Drug Z	X vs Y	X vs Z	Y vs Z
Total Patients	80	75	76			
Baseline Visit						
n	76	71	66	.651	.033*	.026
Mean	555	650	700			
Std.Dev.	187	200	245			
Min.,Max.	182,932	108,1040	246,1458			
Final Study Visit						
n	59	60	56	.300	.007**	.213
Mean	698	751	770			
Std.Dev.	199	219	244			
Min.,Max.	275,1007	268,987	344,1472			

@ P-Values are adjusted by study site

Example 4.2 Template:

```
TABLE 4.2                                                    #    -#  -#
VS/MR FORM LISTED BY PATIENT                                          PAGE 1

Country: #              Investigator: #        Gender: #              Age: #
   Site: #                 Patient ID: #          Race: #          Birthday: #

VITAL SIGNS          Baseline    Visit 1      Visit 2      Visit 3      Visit 4 Final
Visit Date:             #           #            #            #            #
Height (inches):        #           #            #            #            #
Weight (pounds):        #           #            #            #            #
Temperature (F):        #           #            #            #            #
Systolic BP:            #           #            #            #            #
Diastolic BP:           #           #            #            #            #
Comments: #

MEDICATION RECORD                   Visit 1      Visit 2      Visit 3      Visit 4 Final
Prescribed Dose:                    AM:   #      AM:   #      AM:   #
                                    NOON: #      NOON: #      NOON: #
                                    PM:   #      PM:   #      PM:   #
Number of Capsules Returned:                        #            #            #
Comments: #
```

Example 4.2 Preprocessed SAS Data:

OBS	COL1	COL2	COL3	COL4	COL5
1	NC400	001	044		
2	U.S.	001	FEMALE	60	
3	NC400	044	WHITE	07/04/30	
4	08/20/90	09/17/90	10/22/90	11/12/90	12/10/90
5	62.5				
6	120				117
7	98.2				98.0
8	120	124	142	136	128
9	72	80	88	76	72
10	PT RESCHEDULED V2 1 WK LATER				
11	2	2	2		
12	2	2	0		
13	2	2	1		
14	12	0	0		
15	PT DID NOT RETURN CAPLETS AT V4				

Example 4.2 Report Table:

```
TABLE 4.2                                                    NC400-001-044
VS/MR FORM LISTED BY PATIENT                                          PAGE 1

Country: U.S.           Investigator: 001      Gender: FEMALE         Age: 60
   Site: NC400             Patient ID: 044        Race: WHITE      Birthday: 07/04/30

VITAL SIGNS          Baseline    Visit 1      Visit 2      Visit 3      Visit 4 Final
Visit Date:          08/20/90    09/17/90     10/22/90     11/12/90     12/10/90
Height (inches):      62.5
Weight (pounds):     120                                               117
Temperature (F):      98.2                                              98.0
Systolic BP:         120          124          142          136          128
Diastolic BP:         72           80           88           76           72
Comments: PT RESCHEDULED V2 1 WK LATER

MEDICATION RECORD                   Visit 1      Visit 2      Visit 3      Visit 4 Final
Prescribed Dose:                    AM:   2      AM:   2      AM:   2
                                    NOON: 2      NOON: 2      NOON: 0
                                    PM:   2      PM:   2      PM:   1
Number of Capsules Returned:                        12            0            0
Comments: PT DID NOT RETURN CAPLETS AT V4
```