SAS* Macros Designed to Create Camera-Ready, "Typed" Tables Directly from SAS Data Sets

Tim J. Gabel, Research Triangle Institute

Abstract

There are often occasions in the reporting of analysis results when the following constraints apply:

1. A large number of tables are to be produced.
2. A specified format must be followed; i.e. tables are to appear in a form not available through PROC PRINT, PROC TABULATE, or some other convenient procedure.
3. Quality control measures dictate that the numbers not be transcribed and typed manually into correctly formatted table shells.

Normally, the use of PUT statements and a FILE statement would be the logical solution. PUT statements offer a great deal of flexibility in the presentation of data. However, in the case of a large number of tables it can become a tedious, time-consuming, and expensive process to set up job after job of PUT statements. SAS Macros can be used to speed things up greatly. This paper will present several techniques and suggestions to aid programmers in designing macros to produce camera-ready tables. It will also discuss a "real-world" example of how this solution was successfully implemented.

Introduction

In the reporting of analysis results, client desires and expectations often differ dramatically from convenient programmer-devised solutions. What may seem like simple tabular specs to a client may cause the programmer hardship due to some of the SAS internal constraints. Perhaps titles are to be underlined, or variable labels will exceed forty characters. Special headings may need to be formed which tie together several columns. Possibly page breaks need some special notation. For whatever reasons, PROC PRINT, PROC TABULATE, or any other SAS output procedure may be deemed inadequate for client wishes.

Tables could be typed by secretarial staff, but what if there were to be several hundred tables, with multiple pages for each. The quality control measures necessary to avoid transcription errors would become a major undertaking. For a large task, this solution is also unacceptable.

How about PUT statements? They offer wide flexibility to format tables according to client wishes. They avoid the quality control problems associated with typing the tables. However, the constraint now becomes programmer time. The tedious task of setting up 100 or more jobs full of PUT statements is very uninviting. In some cases it may be unavoidable. However, in many situations SAS Macros can speed up this process immensely. This paper will present several techniques and suggestions which I have found to be very helpful in designing customized macros for report writing applications. It will show how the user can design his own macros to mass-produce tables. By creating a generalized macro that inputs table parameters from the user, producing a large volume of tables can be a relatively simple task. This paper will also offer a "real-world" example of how the SAS Macro Facility helped to produce camera-ready tables with limited programmer intervention.

SAS Macro Language Tools

The neat and precise presentation of results is often as important to a data analyst as the methods he uses to achieve those results. The way in which he chooses to display the data to the audience can affect the way the data is interpreted. Often times, the role of the SAS programmer is to prepare the analyst's results for inclusion in a final report. There are generally some specific requirements in terms of how the data should appear. The programmer must resolve the differences between what the computer will readily provide, and what the analyst wants it to provide. This is not always a small difference.

There are some basic tools available in the SAS Macro Language which can be directly applied to report writing applications. It is possible to use these tools, by themselves and in combination with one another, to create a complex, "table-generating" macro. Such a macro can make it much easier for the programmer to produce output results. It can be as general or specific as the programmer wants it to be. It merely needs to be supplied with enough information about each table that is to be printed. The first portion of this paper will discuss four of the tools from the SAS Macro Language, and how they can be used in the development of a "table-generating" macro.

1. OPTIONS DQUOTE

As many programmers already know, it is very important to use the DQUOTE Option whenever using the SAS Macro Language. When this option is in effect, the double quote works exactly as the single quote. It can be used anywhere the single quote might appear such as TITLE statements, LABEL statements, etc. The DQUOTE option takes nothing away from the programmer; it allows text to be enclosed in either single or double quotes. However, there is a notable difference between the two types of quotes. A macro reference enclosed in single quotes cannot be resolved. It is treated as literal text. On the other hand, a macro reference enclosed in double quotes will be resolved by the macro
The following example illustrates this point.

\[
\begin{align*}
\%LET \text{COLOR1}=\text{BLUE}; \\
\%LET \text{COLOR2}=\text{GREEN}; \\
\%LET \text{VARFMT1}=\text{text for level 1 of a variable}; \\
\%LET \text{VARFMT2}=\text{text for level 2 of a variable}; \\
\%LET \text{VARFMTn}=\text{text for level n of a variable}; \\
\end{align*}
\]

Each format, which would normally appear in a VALUE statement in PROC FORMAT, is assigned to a separate macro variable. Then a series of IF-THEN statements would decide which format to print.

\[
\begin{align*}
\text{IF VARIABLE}=1 \text{ THEN PUT "VARFMT1"; ELSE IF VARIABLE}=2 \text{ THEN PUT "VARFMT2"; ELSE IF VARIABLE}=n \text{ THEN PUT "VARFMTn";}
\end{align*}
\]

A little messier than PROC FORMAT, but now we're not restricted to forty characters. The macro's references to titles, footnotes, row labels, etc. should also be surrounded by double quotes.

2. \%LENGTH Function

The \%LENGTH function can be a valuable tool for customized report writing, especially in applications requiring the centering or underlining of text. The function returns the length of a string or macro reference. It is important to realize that it resolves into an actual integer, as opposed to resolving into an expression whose value is an integer. There are times when SAS will only allow an integer to appear in an expression or function call. One such case is the \%n format modifier, which is used in conjunction with PUT statements. This allows the user to repeat a particular format or string \n times with one PUT statement. When building a general macro that underlines variable labels or titles, the number of underscore characters that form the line will change from table to table. Therefore, by assigning the title or variable label to a macro variable, the \%LENGTH function will tell the macro how many underscores to print so that the line is exactly under the string being output.

For instance, assume it is desired that titles be underlined. The following code could appear in the macro:

\[
\begin{align*}
\text{PUT \&TITIE \"Distribution of Age by Race\" \&COLOR1 \%LENGTH(\&TITIE) 12}; \\
\text{PUT \&TITIE \%LENGTH(\&TITIE) \&COLOR2 \%LENGTH(\&TITIE) \%LENGTH(\&TITIE) \%LENGTH(\&TITIE) \%LENGTH(\&TITIE) \%LENGTH(\&TITIE) \&COLOR1};
\end{align*}
\]

If the title were assigned as,

\[
\begin{align*}
\%LET \&TITIE=\text{Distribution of Age by Race};
\end{align*}
\]

the resolved macro code would read:

\[
\begin{align*}
\text{PUT \&TITIE \&COLOR1 \&COLOR2 \&COLOR1};
\end{align*}
\]

No matter how long a title is, the macro will calculate its length and print it with a line under it. In this case, the title was twenty-seven characters long, and the number 27 was used as the format multiplier. As titles and their lengths change from table to table, the format multiplier will reflect the changes. Using \%LENGTH to center text works in much the same way. By knowing the width of the page and the length of the text to be centered, a simple formula yields a column number indicating where to start printing the text. Assume the above example is still the case. However, rather than starting in column 1, suppose the title is to be centered on a 132 column page. The code could be modified as follows:

\[
\begin{align*}
\text{T LOC}=\text{INT}(132 - \%LENGTH(\&TITIE)) / 2; \\
\text{PUT \&T LOC "TITTLE" \&COLOR1 \&COLOR2 \&COLOR1 \&COLOR2 \&COLOR1 \&COLOR2};
\end{align*}
\]

T LOC is a SAS variable which is assigned the value of INT((132-27)/2)=52. The INT (integer) function truncates any fractional remainder due to the division by 2. In this case, the text would start printing at column 52 and the title would be centered on the page. By taking advantage of \%LENGTH, the user can allow the lengths of titles and labels to vary, without having to tell the macro their lengths.

3. Iterative DO Statement

For creating customized reports, the DO statement can be one of the most powerful and versatile tools available to the programmer. It is very similar to the iterative DO statement in the DATA step, in that a start and stop value are supplied and the number of iterations is based on their values. Any SAS code that is inside the loop will be repeated for each iteration. For instance, in an earlier example we saw how to circumvent PROC FORMAT and create formats longer than forty characters. We could have replaced the series of IF-THEN statements with a single IF statement inside an iterative
A DO loop, and the macro would generate the necessary number of statements.

Instead of:

```sas
IF VARIABLE=1 THEN PUT "&VARNAMES1;" ELSE IF VARIABLE=2 THEN PUT "&VARNAMES2;" ELSE etc.
```

We would write:

```sas
DO I=1 TO VARIABLE; IF VARIABLE=I THEN PUT "&VARNAMESI;" ELSE %END;
```

In this case, a null ELSE will be output on the last iteration of the DO loop. The lone semicolon on the last line of code would terminate the null ELSE. The macro variable VARIABLE is the stop value in the DO loop, and would have been passed as a parameter to the macro. The double ampersand (&&) that appears in the second line causes a recursive action to occur. It signals for the macro processor to re-scan the reference to VARNAMES. That way, &I will be resolved into its value on the first scan, and then VARNAMES will be resolved into its value based on the value of &I.

Most of the time, the iterative DO is not a tool in and of itself. Rather, it is used in conjunction with other SAS Macro Language tools to strengthen their value to the programmer. The index variable is a macro variable and that is a key to the power of the DO. By using the index as a comparison value, or as an argument in a function call, a macro can be written to handle the variability involved in customizing reports. For instance, in the above example, VARIABLE might have three levels or twenty levels. By using &I to look at the value of VARIABLE, the user only needs to tell the macro how levels of VARIABLE (by defining &VARNAMES). In the next section another application will be shown using the value of &I from the DO-loop.

4. %SCAN Function

The %SCAN function is the macro equivalent to the SAS SCAN function. It scans a string for "words" that are separated by delimiters. This is a function that can have a major impact on the development of a generalized macro. Realize that a great deal of information can be passed to the macro with one parameter. By listing several items in one %LET statement, and then telling the macro how many were listed, the macro can always look to the same macro variable and pick off each individual item listed.

The SAS variables that appear in a table will vary from one table to the next. It is not necessary to force the macro to know ahead of time how many variables may appear, or even know their names. That defeats the purpose of having a general macro. By using %SCAN, the macro only needs to know the parameter that will contain all of the SAS variable names. All of the variable names would be listed in a %LET statement and separated by a delimiter. We would also define a parameter representing the total number of variables. Then, placing the %SCAN function within a DO loop would allow the scan to step through the variable names, treating each one as a separate entity. The index of the DO will appear as the second argument in the %SCAN call. It will tell the %SCAN which item to retrieve. This logic is implemented in the following code.

```sas
%LET VARNAMES=AGE / SEX / RACE / EDUCATE;
%LET FORMATS=AGE. / SEX. / RACE. / EDUCATE.;
%LET NUMVARS=4;
```

We have defined three parameters for the macro to use. VARNAMES holds the names of the variables, FORMATS holds their format names, and NUMVARS tells the macro many variables there are. Using the information we have supplied, the macro could do something like this:

```sas
PUT %DO I=1 TO NUMVARS;
  %SCAN(VARNAMES,&I,/) %SCAN(FORMATS,&I,)
%END;
```

This code would generate a PUT statement to write out all four variables across a page. Each variable would have its own format associated with it. The resolved code would read:

```sas
PUT AGE AGE. SEX SEX. RACE RACE. EDUCATE.;
```

Notice that a slash (/) has been specified as the delimiter in the %SCAN. Just as in the SAS scan, the %SCAN has a set of default delimiters that it will use if none is provided. However, it is always a good idea to indicate a delimiter for the %SCAN to use. That way, the programmer always has control over what characters have special meaning to the macro. He should know the data well enough to choose a delimiter that will never appear as part of the text being scanned. When designing a general macro for these types of applications, try to anticipate what the data will look like and specify a delimiter accordingly.

The 1985 Department of Defense Surveys

The 1985 Department of Defense Survey of Officers, Enlisted Personnel, and Military Spouses was a world-wide sample of military personnel and their spouses. 89,000 military members and 41,000 spouses were surveyed, and data was collected on a wide variety of topics. These topics ranged from demographic information to length of service in the military to feelings about social issues and military life. The data was analyzed and presented in three separate reports to the U.S. Department of Defense.

In addition to these reports, six volumes of supplemental tables were prepared. Over 450 tables were printed, summarizing the data as 2-, 3-, and 4-way cross tabulations. In all, they totaled nearly 5,000 pages. The vast number of
tables that had to be created made it imperative that they not be typed. However, the tables had a common thread. All of the information was to be broken down by branch of the military. Most of the tables had the same five columns: Army, Navy, Marine Corps, Air Force, and Total DoD. Since so many tables had the same column structure, it became clear that one macro could produce all of the tables. The number of columns was always the same, so the print positions across the page also remained constant. A macro was developed that would mass-produce the tables.

The macro required parameters to tell it titles, footnotes, etc. It also needed information about the row variables. This included names, labels, formats, and number of levels for each variable. There were different versions of the macro, depending on whether it was a 2-, 3-, or 4-way cross tabulation. Therefore, it also had to know which of the row variables, if any, would be nested inside of other variables. Once all of this information was supplied, the macro was able to create and execute all of the PUT statements necessary to print each table.

The tables were output to an online file by using a FILE statement in the DATA step. The file was then printed on a Talaris laser printer. This is an important factor to consider since the laser printer produced a far superior typeface than a line printer would have produced. This gave the tables a "typed" look, rather than a "computer-generated" look. That fact pleased the report authors, and allowed the tables to be directly inserted into the text wherever they were needed.

At the end of this paper is a table labeled Table 3.8. It is the first page of one of the tables that was produced for the 1985 Department of Defense Surveys. Also included is the SAS code that defined the parameters for the table. This should give a general impression of the information the macro needed to create the tables. Space does not permit listing the entire macro.

Conclusion

In all, the macros for the 1985 Department of Defense Surveys took 80-100 hours to develop and debug. A great deal of time was spent trying to discover ways around some of the stumbling blocks that existed. The issues and examples that have been raised in this paper show a few of the problems that were encountered. Once the macro was executing properly, each table took approximately 5-7 minutes to set up and run. Since there were so many tables to generate, the initial start-up time was justifiable. The most time consuming part of the whole process turned out to be keying in titles and footnotes for each table. Typing in the parameters didn’t require any SAS knowledge, so programmers were freed up to tackle other programming tasks. It is my hope that some of the discoveries that were made by trial and error on this project will help other programmers to develop a macro system to use for their own applications involving tables and report writing.

References


To contact the author:
Tim J. Gabel
Research Triangle Institute
PO Box 12194
Research Triangle Park, NC 27709

SAS is the registered trademark of SAS Institute Inc., Cary, NC, USA.
### Table 3.8
Personal Characteristics of Civilian Spouses by Sex and Service for Spouses of Officers

<table>
<thead>
<tr>
<th>Sex of Spouse</th>
<th>Service</th>
<th>Army</th>
<th>Navy</th>
<th>Corps</th>
<th>Force</th>
<th>Total</th>
<th>DoD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Male</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Age of Spouse</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Less than 20 years</td>
<td></td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
<td></td>
</tr>
<tr>
<td>20-24 years</td>
<td></td>
<td>5.22</td>
<td>3.19</td>
<td>5.56</td>
<td>3.66</td>
<td>4.23</td>
<td></td>
</tr>
<tr>
<td>25-29 years</td>
<td></td>
<td>21.57</td>
<td>21.28</td>
<td>22.22</td>
<td>25.61</td>
<td>23.03</td>
<td></td>
</tr>
<tr>
<td>30-34 years</td>
<td></td>
<td>22.52</td>
<td>25.53</td>
<td>13.89</td>
<td>27.44</td>
<td>24.84</td>
<td></td>
</tr>
<tr>
<td>35-39 years</td>
<td></td>
<td>20.87</td>
<td>17.02</td>
<td>8.33</td>
<td>15.85</td>
<td>18.00</td>
<td></td>
</tr>
<tr>
<td>Over 39 years</td>
<td></td>
<td>29.82</td>
<td>32.98</td>
<td>50.00</td>
<td>27.44</td>
<td>29.90</td>
<td></td>
</tr>
<tr>
<td>Total Age</td>
<td></td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
<td></td>
</tr>
<tr>
<td>Number of Cases</td>
<td></td>
<td>148</td>
<td>94</td>
<td>36</td>
<td>164</td>
<td>442</td>
<td></td>
</tr>
<tr>
<td>Number of Personnel (in 1000's)</td>
<td></td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Education of Spouse</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Less than 12 years</td>
<td></td>
<td>0.67%</td>
<td>1.06%</td>
<td>0.00%</td>
<td>0.62%</td>
<td>0.72%</td>
<td></td>
</tr>
<tr>
<td>High school graduate</td>
<td></td>
<td>8.68</td>
<td>10.64</td>
<td>11.11</td>
<td>8.64</td>
<td>9.11</td>
<td></td>
</tr>
<tr>
<td>Some college</td>
<td></td>
<td>29.37</td>
<td>25.53</td>
<td>33.33</td>
<td>32.72</td>
<td>29.90</td>
<td></td>
</tr>
<tr>
<td>College graduate</td>
<td></td>
<td>25.41</td>
<td>24.47</td>
<td>27.78</td>
<td>23.46</td>
<td>24.52</td>
<td></td>
</tr>
<tr>
<td>More than 4 years college</td>
<td></td>
<td>35.87</td>
<td>38.30</td>
<td>27.78</td>
<td>34.57</td>
<td>35.75</td>
<td></td>
</tr>
<tr>
<td>Total Education</td>
<td></td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
<td></td>
</tr>
<tr>
<td>Number of Cases</td>
<td></td>
<td>146</td>
<td>94</td>
<td>36</td>
<td>162</td>
<td>438</td>
<td></td>
</tr>
<tr>
<td>Number of Personnel (in 1000's)</td>
<td></td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Race/Ethnicity of Spouse</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Black</td>
<td></td>
<td>15.25</td>
<td>2.13%</td>
<td>5.71%</td>
<td>7.27%</td>
<td>9.44%</td>
<td></td>
</tr>
<tr>
<td>Hispanic</td>
<td></td>
<td>7.47</td>
<td>6.38</td>
<td>0.00</td>
<td>6.06</td>
<td>6.60</td>
<td></td>
</tr>
<tr>
<td>White</td>
<td></td>
<td>70.98</td>
<td>85.11</td>
<td>85.71</td>
<td>80.61</td>
<td>77.69</td>
<td></td>
</tr>
<tr>
<td>Other</td>
<td></td>
<td>6.30</td>
<td>6.38</td>
<td>8.57</td>
<td>6.06</td>
<td>6.26</td>
<td></td>
</tr>
<tr>
<td>Total Race/Ethnicity</td>
<td></td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
<td></td>
</tr>
<tr>
<td>Number of Cases</td>
<td></td>
<td>149</td>
<td>94</td>
<td>35</td>
<td>165</td>
<td>443</td>
<td></td>
</tr>
<tr>
<td>Number of Personnel (in 1000's)</td>
<td></td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

Source: Spouse Questionnaire: Q. 18,37,38,39,40,42,43,45

1218
**TRANSLATION TABLE FOR MACRO VARIABLES**

<table>
<thead>
<tr>
<th><strong>TABNUM</strong></th>
<th><strong>ROWLAB1</strong></th>
<th><strong>ROWLAB2</strong></th>
<th><strong>ROWLAB3</strong></th>
<th><strong>NEST1</strong></th>
<th><strong>NESTLVL1</strong></th>
<th><strong>NEST1FMT</strong></th>
<th><strong>NUMROWS</strong></th>
<th><strong>ROWVARS</strong></th>
<th><strong>ROWLVL5</strong></th>
<th><strong>FORMATS</strong></th>
<th><strong>INDSET</strong></th>
<th><strong>DEST</strong></th>
<th><strong>LABLIST</strong></th>
<th><strong>TITLE1</strong></th>
<th><strong>TITLE2</strong></th>
<th><strong>TITLE3</strong></th>
<th><strong>FOOT</strong></th>
<th><strong>PRTMAC</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Table Number</strong></td>
<td><strong>Labels for Nest Variables</strong></td>
<td><strong>Labels for Nest Variables</strong></td>
<td><strong>Labels for Nest Variables</strong></td>
<td><strong>Name of Outside Nest Variable</strong></td>
<td><strong>Number of Levels for Nest Variable</strong></td>
<td><strong>Format Name for Nest Variable</strong></td>
<td><strong>Number of Inside Row Variables</strong></td>
<td><strong>Names of Inside Row Variables</strong></td>
<td><strong>Number of Levels for Inside Row Variables</strong></td>
<td><strong>Format Names for Inside Row Variables</strong></td>
<td><strong>Input Data Set Name</strong></td>
<td><strong>DD Name for Output File</strong></td>
<td><strong>Labels for Inside Row Variables</strong></td>
<td><strong>Titles for the Table</strong></td>
<td><strong>Titles for the Table</strong></td>
<td><strong>Titles for the Table</strong></td>
<td><strong>Footnote for the Table</strong></td>
<td><strong>The Macro Name</strong></td>
</tr>
</tbody>
</table>