MACROs - A FEW USES TO GET YOU STARTED
MICHAEI M. BROWN, SYNTEX RESEARCH

ABSTRACT
Due to the fact that MACROs are a large, complex, self-contained language, within an ever growing software system, figuring out where and how to use them can be difficult. Also, since there are always novice programmers appearing each year, reiterating MACRO uses becomes necessary. Therefore, this paper will present four ways to put the MACRO language to work. The first example shows how to create a block of statements for inclusion in other areas of the job. The second example demonstrates a repetitive variable replacement within a block of statements. The third example shows how to use the %LET statement in the PROC step. Last, a use for the SYMGET statement will be demonstrated. The examples presented attempt to do so in a simple straightforward fashion; and when put to use they can become powerful.

INTRODUCTION
MACROs are very powerful and can be used in a very complicated manner, however, there are a few ways to use MACROs that are not so complicated. For instance, the same block of statements can be used over and over again. MACROS are good for simplifying this kind of problem. Instead of writing statements over and over, a programmer can write the statements once and embed them within a MACRO.

EXAMPLE 1. REPEATING BLOCKS OF CODE
This MACRO contains a DATA statement and a PROC PRINT statement which is executed before the data is statistically analyzed. The MACRO is named EX1, and will be stored temporarily in the program, by using the %MACRO EX1 and %MEND statements before and after the statements to be included in this particular MACRO. After creating this MACRO, it can be used later in the program by placing the statement %EX1 where this MACRO is to appear and be executed.

Figure 1 shows the DATA and MACRO EX1. Figure 2 shows program execution of this MACRO, and Figure 3 shows some of the output that was subsequently created.

EXAMPLE 2 USING MACROs WITH VARIOUS REPLACEMENT
MACRO usage is also very helpful when you have to run a PROC on one or two variables at a time, but you have quite a few variables to process. Instead of writing the same block of statements and changing the variable being processed, one block of MACRO statements is used to identify all my variables for replacement.

My second MACRO example does this. This MACRO contains a PROC TTEST and two MACRO variables in the VAR statement which will be used for replacement. The MACRO is named EX2 and the variables VARI and VAR2 are identified in parenthesis. After the %MEND statement, the variables that are to be included in the PROC TTEST are identified with the MACRO replacement statement. This statement can be written as many times as one needs and will be executed that many times, so be careful.

Figure 4 shows the MACRO EX2 program statements, Figure 5 shows the program execution of this MACRO, and Figure 6 shows some of the output that was subsequently created.

EXAMPLE 3 USING THE %LET STATEMENT
Occasionally, a programmer may come upon a situation where it is necessary to pass a variable value to a group of MACRO statements. The %LET statement is good for this function.

In my third example of MACRO uses, the %LET statement is used to pass a value of a MACRO variable called CODE to the inside of a MACRO group. This MACRO group, CODCHART, uses the %IF, %THEN, and %DO statements to conditionally execute a PROC CHART, based on the value of the variable CODE. This example also shows how to use MACROs in a fashion that emulates DATA step programming.

Figure 7 shows the program statements using the %LET statement, Figure 8 shows the conditionally executed MACRO group, Figure 9 shows the output of the conditionally executed MACRO PROC group.

EXAMPLE 4 USING THE SYMGET STATEMENT
My final example shows one more way in which MACROs can be used without too much trouble. An occasion may arise where you want to assign a value of one MACRO variable to another variable. This can be done using the SYMGET statement.

In the final example, the SYMGET statement is used to retrieve the value of the variable CODE, which was used in the last DATA step. The SYMGET statement then passes the value of the MACRO variable CODE to the variable REPORT. This is used to execute a small report writer based on the value of the variable REPORT.

Figure 10 shows the statements used to pass the value of the MACRO variable CODE to the variable REPORT. Figure 11 shows the output produced by this sample used of SYMGET.
This MACRO program took some sales data and produced statistics, graphs and reports using some different MACRO statements. The way the MACROs were used was not too difficult to understand. Perhaps, you can use some of these examples or your own in a setting suitable for you.

I would like to acknowledge Marie Whalen and Kathy Jones for their valuable help in putting this poster/paper together.

wp/3443h

OPTIONS MACROGEN REPORT SYMBOLGEN DOUBLE MONO;
DATA CENPS;
INPUT SUBJECT ID; GROUP $; DAGE $; BMED $; PMED $; BW $; PMED $; BW $;
BUDDY;
A 76 97 90 72 45 34 27
B 65 76 62 45 65 87 91 91
C 32 45 12 73 84 62 65 67
D 21 44 52 11 63 34 91 67
E 74 76 02 55 72 46 55 66
F 36 67 50 85 38 68 68 65
G 29 41 86 56 95 62 74 65
H 34 51 33 40 95 62 74 65
I 40 67 54 30 75 95 71 70
J 9 74 74 64 72 38 30 71 24
K 68 78 48 82 30 43 92 61
L 26 36 47 58 64 69 71 82
**EXAMPLE 1 BUILDING BLOCKS OF MACRO CODE:**

MACRO (x);
DATA EXMPSS;
SET EXMPSS;
PROC FREQ;
TABLES ID GROUP;
/*THE MACRO GROUP WILL BE INCLUDED HERE*/
RUN;
/*THE MACRO GROUP WILL BE INCLUDED HERE*/

MACRO Y;
DATA EXMPSS;
SET EXMPSS;
PROC MEANS;
VAR BMI DAGE; 
/*THE MACRO GROUP WILL BE INCLUDED HERE*/
RUN;
/*THE MACRO GROUP WILL BE INCLUDED HERE*/

MACRO Z;
DATA EXMPSS;
SET EXMPSS;
PROC UNIVARIATE;
VAR DAGE; 
/*THE MACRO GROUP WILL BE INCLUDED HERE*/
RUN;
/*THE MACRO GROUP WILL BE INCLUDED HERE*/

PROCEDURE 1 (MACRO CODE PROGRAM STATEMENTS)
SAS PROC CORR

```sas
DATA example;
  Input X Y;
  Datalag X Y;
  Run;
```

**Figure 1 (Macro: COVAR EXECUTED PROGRAM STATEMENTS)**

**SAS PROC CORR**

**VARIABLE: (MO)**

<table>
<thead>
<tr>
<th>GROUP</th>
<th>N</th>
<th>MEAN</th>
<th>STD DEV</th>
<th>STD ERROR</th>
<th>MINIMUM</th>
<th>MAXIMUM</th>
<th>MIN. VARIANCES</th>
<th>T</th>
<th>DF</th>
<th>P &gt;</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>6</td>
<td>48.14466667</td>
<td>21.3642960</td>
<td>0.3232277</td>
<td>25.0000000</td>
<td>76.0000000</td>
<td>0.72948</td>
<td>9.9</td>
<td>0.4854</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>6</td>
<td>39.56480557</td>
<td>19.20359453</td>
<td>0.38565084</td>
<td>11.0000000</td>
<td>65.0000000</td>
<td>0.72948</td>
<td>10.0</td>
<td>0.4857</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**VARIABLE: (MO2)**

<table>
<thead>
<tr>
<th>GROUP</th>
<th>N</th>
<th>MEAN</th>
<th>STD DEV</th>
<th>STD ERROR</th>
<th>MINIMUM</th>
<th>MAXIMUM</th>
<th>MIN. VARIANCES</th>
<th>T</th>
<th>DF</th>
<th>P &gt;</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>6</td>
<td>48.00000000</td>
<td>21.3642960</td>
<td>0.3232277</td>
<td>25.0000000</td>
<td>76.0000000</td>
<td>0.72948</td>
<td>9.9</td>
<td>0.4854</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>6</td>
<td>51.00000000</td>
<td>19.20359453</td>
<td>0.38565084</td>
<td>11.0000000</td>
<td>65.0000000</td>
<td>0.72948</td>
<td>10.0</td>
<td>0.4857</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 2 (Output from Macro EX2)**
**Example 6 Using Strfmt;**

**Data Example;**

**Format Code;**

**Report Strfmt ("CODE");**

**IF Report."C" Then Quito.**

**Q01;**

**Data New;**

**Set Example;**

**File Print Mistakes Weaken-8;**

**By Sales-Month-Year;**

**If First Obs Then Lmk: 4;**

**Put ///**

**#1 Subject #2 Group #3 Quito #4 (Q2) (Q3) (Q4) (Q5) (Q6) (Q7);**

**Group Sales;**

**Return;**

**Put #7 "Quarterly Sales Report;**

**Return;**

**Figure 10 (Strfmt Program Statements)**

**Quarterly Sales Report**

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>36</td>
<td>87</td>
<td>99</td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td>63</td>
<td>76</td>
<td>82</td>
</tr>
<tr>
<td>3</td>
<td>B</td>
<td>32</td>
<td>45</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>B</td>
<td>21</td>
<td>44</td>
<td>52</td>
</tr>
<tr>
<td>5</td>
<td>A</td>
<td>43</td>
<td>26</td>
<td>82</td>
</tr>
<tr>
<td>6</td>
<td>A</td>
<td>25</td>
<td>36</td>
<td>47</td>
</tr>
<tr>
<td>7</td>
<td>B</td>
<td>17</td>
<td>71</td>
<td>37</td>
</tr>
<tr>
<td>8</td>
<td>B</td>
<td>52</td>
<td>47</td>
<td>26</td>
</tr>
<tr>
<td>9</td>
<td>A</td>
<td>52</td>
<td>54</td>
<td>45</td>
</tr>
<tr>
<td>10</td>
<td>B</td>
<td>51</td>
<td>73</td>
<td>61</td>
</tr>
<tr>
<td>11</td>
<td>A</td>
<td>68</td>
<td>49</td>
<td>68</td>
</tr>
</tbody>
</table>

**Figure 11 (Output from the Use of the Strfmt Statement)**

1023