Understanding the SAS® Macro Facility

Martin J. Rosenberg, Ph.D., M. J. Rosenberg & Associates

Abstract

This tutorial introduces the SAS macro facility and describes what macros are, why they are necessary, common applications, and how SAS processes them. No prior experience with macros is necessary, but an understanding of the DATA and PROC steps is assumed.

What Are Macros?

Macros are a facility within SAS for writing flexible code. To illustrate what flexible code means let’s look at a program written in a traditional programming language. PROC FREQ is a SAS procedure for constructing and analyzing contingency tables and is itself a program written in PL/I in the mainframe and minicomputer versions of SAS or in C in the PC version. Let’s examine the two examples below.

Example 1: Two-way table

```
PROC FREQ;
  TABLES BRANCH * BALANCE;
```

Example 2: Three-way table with title

```
PROC FREQ;
  TABLES CLINIC * DRUG * RESPONSE;
  TITLE 'CLINICAL TRIAL RESULTS';
```

The first example is a typical application in the banking industry. Let BALANCE be a variable that groups accounts into one of several categories according to the size of the balance in the account. Then Example 1 uses FREQ to count the number of accounts in each size category in each bank branch. Example 2 is a typical use of PROC FREQ in the pharmaceutical industry. If the variable RESPONSE measures the response to a drug as one of five categories (e.g., worse, none, fair, good, excellent) then this example counts the responses to two or more drugs for each of two or more clinics. Additionally a title specific to this application of PROC FREQ is printed. These two examples produce two very different contingency tables in two disparate industries. Nonetheless, we need not change the PL/I or C code in either case. We merely change the parameters input to PROC FREQ. Contrast this with Example 3 below which shows a SAS program before the macro facility was available.

Example 3: SAS program without macros

```
DATA SALES;
  SET IN.MASTER;
  IF MONTH = 'JUNE';
  etc.
PROC PRINT;
  DATA=SALES;
  TITLE 'SALES RESULTS FOR JUNE';
  etc.
```

In this example, a master dataset is read and observations for the month of June are selected. Later in the program, a report of sales results for June is printed. If now we want to use this program to create a report for July, we must actually go into the SAS code and change all instances of June to July. While this would be easy to do in this simple example, if the program were several thousand lines long and if there were more than one kind of change, the process could be both difficult and error prone. Hence, we needed a capability in SAS to write programs in a flexible way where parameters are specified at execution time. The macro facility provides this capability.

Uses of Macros

The following list shows typical uses of macros:

- Get system information
- Conditional execution of steps
- Data dependent code
- Repetitive code
- Building production systems
- Pass information between steps
- Develop user-friendly and menu driven systems

The last use, user-friendly and menu driven systems has been one of the most widespread applications of macros and the SUGI Proceedings from 1982 through 1986 are replete with examples of systems built with macros. Although macros can still be used for this purpose, the introduction of the SAS/AF product in Version 5 has provided an alternative which is easier to use both at the time of initial programming and for subsequent maintenance. I advise people who wish to build user-friendly systems to explore this alternative.

Components of Macros

Macros are constructed from three components: macro variables, macro statements, and model text. Macro variables follow the usual naming conventions except that they begin with an ampersand (&) in order to distinguish them from ordinary variables. Macro variables are a powerful mechanism in and of themselves and are a good way to begin using the macro facility. A macro variable can have as values: a portion of a SAS statement, a whole SAS statement, or even many SAS statements. At execution time, the value is substituted for the macro variable name. One common and simple use of macro variables is for titles which change each time a program is executed.

Macro statements begin with a percent sign (%) immediately followed by a keyword. They end with a semicolon (;). Examples are:
defines a macro named REPORT
ends a macro
conditional execution
conditional execution

The %IF %THEN and %ELSE statements look familiar. However, whereas ordinary IF THEN ELSE statements can only operate on code contained with a single DATA step, the macro equivalents can go across step boundaries and are the only way to conditionally execute entire DATA or PROC steps.

The last component of macros, model text, are just ordinary SAS statements. Example 4 illustrates these concepts. The statements are numbered to aid the description. Lines 3, 4, 5, and 7 contain the macro variable &J. Line 5 is model text, an ordinary SAS statement. This macro is discussed in greater detail in Example 7.

Example 4: A simple macro
1 %MACRO REPEAT;
2 %DO J = 1 %TO 25
3 DATA FILE&J;
4 SET IN.STUDY&J;
5 IF AGE > 65 AND SEX = 'F';
6 PROC PRINT DATA=FILE&J;
7 TITLE 'RESULTS OF STUDY&J';
8 %END;
9 %MEND;

Flow of a SAS Program With Macros

Understanding the flow of a macro can be important when it comes to writing and debugging sophisticated programs with macros. Figure 1 illustrates the flow of code through the SAS system.

Source statements first go to the word scanner, which, while actually a part of the SAS supervisor, is broken out here for clarity. The word scanner breaks the code into tokens in a process called tokenization. Tokens are described in greater detail later, but for now it is enough to know that they are certain individual characters, or words, or groups of words. In particular, the word scanner looks for the two tokens ampersand (&) and percent sign (%). When the word scanner encounters one of these two tokens, it sends the subsequent token to the macro facility for resolution. The macro facility interprets the token, substitutes the appropriate SAS code, and returns the substituted code to the word scanner for more tokenization.

When the word scanner does not detect an & or %, the token continues on to the SAS supervisor. The SAS supervisor collects the tokens into statements and when a statement is complete it compiles it. Finally, when the SAS supervisor has a complete step (i.e. when it encounters the next DATA, PROC, or RUN statement) compilation stops and the step is executed. The important point is that the output of the macro facility is ordinary SAS code which is passed on for compilation.

Tokenization

There are four token types: name, number, special character, and literal string. Example 5 illustrates each type. Names are words including variable names, keywords, procedure names, etc. Each complete number, including a decimal point if present, is a single token. Hence, 3.14 is one number token, not 4 tokens. Example 5 lists all the special character tokens. Note that & and % are tokens, as is a period (.) when it is not being used as a decimal point in a number (e.g. the period in a two level data set name). Literal strings are everything enclosed by a pair of single quotes. If OPTIONS DQUOTE is set, then everything enclosed by double quotes is also a single token. If OPTIONS NODOQUOTE is set, then double quotes do not define a literal, and the double quote marks are special character tokens. To find out the default value of DQUOTE/NODOQUOTE at your site, ask your installation representative or run PROC OPTIONS. When in doubt, include an OPTIONS statement in your program. This will override the default if necessary.

Example 5: Four token types

<table>
<thead>
<tr>
<th>Type</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>DATA 32</td>
</tr>
<tr>
<td>Number</td>
<td>890</td>
</tr>
<tr>
<td>Special Character</td>
<td>&amp; + -</td>
</tr>
<tr>
<td>Literal String</td>
<td>'SALES REPORT'</td>
</tr>
</tbody>
</table>

When OPTIONS DQUOTE is set, double quotes act the same as single quotes with two exceptions. First, a single quote enclosed within double quotes will not start a new literal. Hence, if you wish a single quote (an apostrophe) to be part of label, title, or other string, you can either enclose the string within double quotes and use one single quote, or enclose the string within single quotes and use two single quotes. For example, the following two TITLE statements will print the identical title THIS MONTH'S SALES.

TITLE "THIS MONTH'S SALES";
TITLE 'THIS MONTH'S SALES';

The more important difference between single and double quotes, is the way macro references enclosed within quotes are handled. The SAS system does not scan literals defined by single quotes for macros, but will scan double quoted literals for macros if OPTIONS DQUOTE is set.
(Remember if OPTIONS DQUOTE is not set, then double quotes do not define a literal). Hence, macros and macro variables enclosed within single quotes will not be resolved, but will be resolved if enclosed within double quotes. This is useful for TITLE statements that include a macro variable.

Exercise 1 below illustrates these points. The %LET statement defines a macro variable whose name is MONTH and assigns it the value JUNE. Note that MONTH does not begin with a ampersand (although subsequent references of MONTH should), and that JUNE is not enclosed within quotes. Neither would be a syntax error, but different actions would be taken. If an ampersand preceded MONTH, SAS would first try to resolve MONTH and would assign the value JUNE to the resolved value of MONTH. On the other hand, if JUNE were enclosed in quotes, then the value assigned to MONTH would include quotes. Armed with these facts, for each TITLE and LABEL statement in Exercise 1, try to write down the title or label that would print on SAS output. Answers to the exercises are provided at the end of this paper.

Exercise 1: Using quotes with macro variables

Write down how each TITLE and LABEL statement will print.

```sas
%LET MONTH = JUNE;
TITLE 'SALES REPORT FOR &MONTH';
OPTIONS DQUOTE;
TITLE "SALES REPORT FOR &MONTH";
LABEL BOOKS = "GEORGE'S BOOKS';
OPTIONS NODQUOTE;
TITLE 'SALES REPORT FOR &MONTH';
```

Exercise 2: Tokenization practice

Divide each statement below into tokens. Answers are given at the end of this paper.

```sas
DATA21;
%LET M=JUNE;
IF SEX='MALE';
IF NAME="MARY";
PI=3.141;
PROC FREQ;
OUTPUT;
%MACRO REPORT(JUNE);
```

Examples of Macro Applications

Examples 6, 7, and 8 display macros used to: get system information; conditionally execute steps in a production system; and produce repetitive code.

Example 6: Getting system information

The following list shows macro variables which are automatically created and defined by the SAS system. To use them, just insert them in your programs. A complete list is given in the SAS User's Guide: Basics Version 5, pp. 691-692.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;SYSDATE</td>
<td>today's date</td>
</tr>
<tr>
<td>&amp;SYSDAY</td>
<td>day of the week</td>
</tr>
<tr>
<td>&amp;SYSDSN</td>
<td>last data set created</td>
</tr>
<tr>
<td>&amp;SYSINDEX</td>
<td>number of macros that have started execution</td>
</tr>
<tr>
<td>&amp;SYSRC</td>
<td>last return code from %TSO or %CMS</td>
</tr>
<tr>
<td>&amp;SYSTIME</td>
<td>time at beginning of job</td>
</tr>
<tr>
<td>&amp;SYSVER</td>
<td>version of SAS executing</td>
</tr>
</tbody>
</table>

Example 7: Conditional execution of steps. Production system.

The following is an example of a macro that calls other macros. The code creates the macro %REPORT which in turn calls the macros %DATA, %ANNUAL, %QUARTLY, and %MONTHLY. The automatic macro variable &SYSDATE resolves as the date the job began execution. The SAS date function MONTH uses &SYSDATE as an argument and returns a number from 1 to 12 indicating the month (1=January, 2=February, etc.). In January an annual report for the year just ended is printed. In April, July, and October a report for the calendar quarter just ended is printed. At other times a monthly report is issued. This example illustrates two points. First, it shows a production system that is run each month and automatically selects one of three reports to print. Secondly, it shows structured code where a main "program" REPORT is little more than a shell which calls "subroutines" that perform all the work. This is a standard technique in traditional programming languages and is available in SAS through the macro facility.

```sas
%MACRO REPORT;
  %* Input data;
  %DATA
  %* Determine report type;
  %IF MONTH(&SYSDATE) = 1 %THEN %ANNUAL;
  %ELSE IF MONTH(&SYSDATE) = 4 OR MONTH(&SYSDATE) = 7 OR MONTH(&SYSDATE) = 10 %THEN %QUARTLY;
  %ELSE %MONTHLY;
%MEND REPORT;
```
Example 8: Repetitive code

Suppose you needed to perform identical analyses on 25 data sets named IN.STUDY1 through IN.STUDY25. One way would be to write out all the code, but even using a text editor this could be a long process. Macros can be used to produce a very succinct set of code as shown below. The macro shown here was discussed earlier in Example 4.

Without macros:

```sas
DATA FILE1;
SET IN.STUDY1;
IF AGE > 65 AND SEX = 'F';
PROC PRINT DATA=FILE1;
TITLE 'RESULTS OF STUDY 1';

DATA FILE25;
SET IN.STUDY25;
IF AGE > 65 AND SEX = 'F';
PROC PRINT DATA=FILE25;
TITLE 'RESULTS OF STUDY 25';
```

With macros this can be written:

```sas
%MACRO REPEAT;
%DO J = 1 %TO 25;
  DATA FILE&J;
  SET IN.STUDY&J;
  IF AGE > 65 AND SEX = 'F';
  PROC PRINT DATA=FILE&J;
  TITLE 'RESULTS OF STUDY &J';
%MEND;
%MACROEND;
```

Summary

This paper has attempted to provide the background necessary to learn macros. It has not been my purpose to teach how to program with macros since this is an undertaking far beyond the scope of a 40 minute tutorial. However, once the information in this tutorial has been assimilated, the reader should be able to learn macros from the SAS manual and practice.

We've seen that macros provide flexibility to SAS programs, and allow SAS to be used as a programming language. Experienced macro programmers have learned that while macros are very powerful, they can also become very difficult to maintain, when new features are to be added at a later date. Kretzman (1986) discusses this conflict between the flexibility provided by macros and the difficulty of maintaining them, and makes recommendations for their judicious use. This should not dissuade you from learning to use macros, but does call for some restraint in their use.

I hope this tutorial has been instructive, and welcome you to the new world of possibilities for using the SAS system, that the macro facility provides.

References


Acknowledgements

The work on this paper was completed while I was an employee of Syntex Research, Palo Alto, California. My thanks to Mr. Gerald Hutchison for typing this manuscript, and to Mr. Kerry Moore-Cintas and Ms. Marilyn Kendrick for typesetting the transparencies.

SAS/IML is a trademark of SAS Institute Inc., Cary, NC, USA.

For additional information, please contact:
Dr. Martin J. Rosenberg
M.J. Rosenberg & Associates
99 East Middlefield Road, Suite 31
Mountain View, CA 94043
(415) 961-9260
Flow of a SAS® Program

SAS Source Statements

Word Scanner

MACRO Facility

No tokens with % &

SAS Code

Raw Data

SAS Supervisor

SAS Data

Data Step Compiler

Format Library

PROCedure Libraries

Base
OR®
GRAPH®
FSP®
ETS®
AF®
QC®
IML™
user written

Figure 1
Answers to Exercises

Exercise 1:

<table>
<thead>
<tr>
<th>Answer</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) SALES REPORT FOR MONTH</td>
<td>When not preceded by an &amp;, MONTH is just a word, not a macro variable.</td>
</tr>
<tr>
<td>(b) SALES REPORT FOR &amp;MONTH</td>
<td>Macro variables are not resolved within single quotes.</td>
</tr>
<tr>
<td>(c) SALES REPORT FOR JUNE</td>
<td>With OPTIONS DQUOTE, macro variables resolve between double quotes.</td>
</tr>
<tr>
<td>(d) GEORGE'S BOOKS</td>
<td>With OPTIONS DQUOTE, a single quote enclosed by double quotes will print.</td>
</tr>
<tr>
<td>(e) You should get a warning or an</td>
<td>error message since double quotes cannot substitute for single quotes with OPTIONS NODQUOTE.</td>
</tr>
</tbody>
</table>

Exercise 2:

(a) DATA21 ;
(b) % LET M = JUNE ;
(c) IF SEX = 'MALE' ;
(d) There are two answers
    IF NAME = "MARY" ;
    IF NAME = " MARY " ;  (if OPTIONS DQUOTE)
    (if OPTIONS NODQUOTE)
(e) PI = 3.141 ;
(f) PROC FREQ ;
(g) OUTPUT ;
(h) % MACRO REPORT ( JUNE ) ;

143