INTRODUCTION TO THE SAS® MACRO LANGUAGE
Doris Steppe, University of Pittsburgh Medical Center, Pittsburgh, PA

ABSTRACT

The SAS® Macro Language allows you to extend and customize the SAS System and to reduce the amount of text you enter.

The purpose of this poster is to present an overview of SAS Macros and how they can be used to streamline SAS programs.

INTRODUCTION

The SAS macro language consists of macro variables, macro facility interfaces, macro programs, and macro storage techniques.

MACRO VARIABLES

SAS macro variables are the basic units that are used by macro facility interfaces and macro programs; they can be created and resolved anywhere in a SAS program. The name and value of a macro variable is stored in memory in a Symbol Table, either local or global in scope; the value is always simply a string of characters. The Symbol Table is created when the first macro variable is defined. Some automatic variables are always defined, e.g., SYSDATE (see Appendix A for list of automatic variables). There are several OPTIONS available to help with macro variables. Here is a sample SAS job with a macro variable, NUM:

```
options sgen;
%let NUM = 28;
proc print data=data.all;
  where var2 = &NUM;
  var name var3;
  title "Var2 = &NUM";
run;
%put NUM = &NUM;
%put _automatic_;
%put _user_;
```

The SGEN option causes the display in the SAS log of the results of resolving the macro variables. The ‘%let …’ statement defines a macro variable, and the ‘%put …’ statement writes to the SAS log. The ‘&NUM’ is replaced (resolved) by the value of macro variable NUM, so the title command becomes ‘title “Var2 = 28”’.

The % and & characters are called macro triggers; the character following the macro trigger must not be a space. Before the code goes to the SAS compiler, the macro processor stores the macro variables in the Symbol Table and resolves the macro variable references in the code. Strings in double quotes are resolved; strings in single quotes are not resolved.

MACRO FACILITY INTERFACES

Macro Facility Interfaces are ways to use macro variables. They are:

1) CALL SYMPUT (macro-variable, text);

   This creates a macro variable named ‘macro-variable’ with a value of ‘text’.

2) Indirect References using multiple ampersands. A double ampersand (&&) in SAS code is resolved to a single ampersand. If we have a macro variable ‘var1’ with a value of ‘R001’, and another macro variable ‘R001’ with a value of ‘Ohio River’, then a string in the SAS code of ‘&&&var1’ is resolved to ‘&R001’ on the first pass, and that is resolved to ‘Ohio River’ on the second pass.

3) PROC SQL SELECT … INTO :macro-variable… E.g.,

```
proc sql;
select sum(amount) into :totamt from data.all;
run;
```

   This defines a macro variable ‘totamt’ with a value that is the string representing the sum of variable ‘amount’ in SAS data set data.all; i.e., something like ‘270.16’.

MACRO PROGRAMS

A macro program or ‘macro’ contains data steps and/or proc steps involving macro variable creation and use. It is similar to a subroutine or function in a procedural programming language. It has a name which is used to call it, and it can accept
parameters. E.g., here is a simple macro program definition using one of the automatic macro variables, syslast, whose value is the name of the most recently created data set:

```sas
%macro mname;
    proc print data=data.all;
        where var1=&newvalue;
        title "Print &syslast data";
    run;
%mend [mname];
```

and this is a SAS program using that macro:

```sas
options mprint mlogic sgen;
%let newvalue=17;
%mname
```

Notice that there is NO SEMICOLON after `%mname`. This line calls the macro named `mname`; there is no semicolon there because that line will be replaced by the code from the macro program itself, which has the required semicolons. Specify the MPRINT, MLOGIC, and SGEN options to generate as much information as possible from the macro processor.

The following macro definition has one positional parameter ‘opts’ (with no = sign) and one keyword parameter ‘filenm’ (with an = sign).

```sas
%macro mname (opts, filenm=file5);
    proc print data=&filenm opts;
        title "Print &filenm data";
    run;
%mend [mname];
```

This macro can be called with any of these formats, specifying either or both or neither of the parameters:

```sas
%mname ()
%mname (double)
%mname (filenm=file6)
%mname (double noobs,filenm=file7)
```

Option ‘double’ causes output to be double spaced, and option ‘noobs’ suppresses printing of the observation number. When the macro program has a parameter that is unspecified in the calling statement, it will use the default value if there is one, otherwise it will use a null string(nothing).

When the calling statement is in a macro program, it can also be conditional, e.g.:

```sas
%macro newprog;
    %if &var1 = 5 %then %do;
        %mname()
    %end;
%mend;
```

### STORAGE TECHNIQUES

There are three ways to store macro programs for future use:

1) Store the source code in an external file and use this statement to pull it into a SAS program:

```sas
%include(file-pathname);
```

2) Use the Autocall Facility to search predefined source libraries for macro definitions. These libraries can consist of external files or SAS catalogs.

3) Store the compiled macros. This does NOT save the SOURCE code! Always maintain the source code separately. To store the compiled macro:

```sas
libname dlib 'library-path';
options mstored sasmstore=dlib;
%m macro mname(parameters) / store des='description';
...
%mend;
```

To access this stored macro:

```sas
libname dlib 'library-path';
options mstored sasmstore=dlib;
%mname(parameters)
```

### CONCLUSION

The macro language is a powerful addition to SAS, allowing increased efficiency and customization of SAS jobs.

### APPENDIX A

A list of some of the automatic macro variables.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSDATE</td>
<td>date of SAS invocation</td>
</tr>
<tr>
<td>SYSDAY</td>
<td>day of the week of SAS</td>
</tr>
<tr>
<td>SYSTIME</td>
<td>time of SAS invocation</td>
</tr>
<tr>
<td>SYSENV</td>
<td>FORE(interactive execution) or BACK(noninteractive or batch execution)</td>
</tr>
</tbody>
</table>
SYSSCP
abbreviation for the operating system being used such as OS, CMS, OpenVMS, WIN, HP 300, or OS/2.

SYSVER
release of SAS software being used

SYSLAST
name of most recently created SAS data set

ACKNOWLEDGMENTS

SAS is a registered trademark or trademark of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

CONTACT INFORMATION

Doris A. Steppe
University of Pittsburgh Medical Center
WPIC Room E-1110
3811 O’Hara St.
Pittsburgh, PA  15213-2593
Phone: (412) 624-0243
E-Mail: steppeda@msx.upmc.edu