# MET MA 603: SAS Programming and Applications



*Macros I*

# Macros

A **Macro** (short for macroinstruction) is a single instruction that upon execution expands automatically into a set of instructions in order to perform a particular task.



Macros are used to dynamically create and modify the code in a program. Most macros are created (i.e., defined) by the user, with the purpose of making one's code more efficient.





Macros can be used to accomplish the following:

- Automate repetitive tasks
- Run the same data step or procedure for several different data sets
- Run a data step or procedure only if a certain condition is met
- Create logic based on the current time or date

Macro topics that will be covered include Macro Variables, Macro Programs, and Macro Logic.

# Macro Processing

Macros can be thought of as shorthand for representing pieces of code. In other words, each macro corresponds to text, which can be a single value, or many lines of code.

**Resolving macros** is the process of replacing each macro with the value it has been given.



When a program includes macros, SAS processes the code in two stages. In the first stage, all macro values are resolved. In the second stage, the resulting code is executed as usual.

Macros can be used to create intricate and complicated code. However, the idea behind macros is itself simple as long as two key facts are remembered…

# Macro Processing (cont.)

The two key facts to remember about macros:

1. Macros are text replacements.

2. All macros are resolved to their values before the code is executed.

# Macro Variables

A **Macro Variable** is a macro component that stores a character string. When a macro variable is resolved, SAS replaces the macro variable with the character string.



The simplest way to create a macro variable uses the %LET statement:



```
%let   myregion = West  ;
```



Names for macro variables must follow the usual rules for SAS names.

Once defined, macro variables are referenced using the ampersand.

```
if region = "&myregion" ;
```

# Resolving Macro Variables

When SAS encounters a macro variable, the compiler replaces the macro variable with its value. The system option SYMBOLGEN instructs SAS to display the values each macro resolves to in the log.



```
SYMBOLGEN:  Macro variable MYREGION resolves to WEST
```



For the example on the previous slide, in resolving the macro **myregion**, SAS replaced **&myregion** with **West**.



In recognizing macro names, SAS begins when it encounters the ampersand, and ends when it encounters anything that cannot be part of a SAS name (letter, digit, underscore). See next slide for examples.

# Resolving Macro Variables (cont.)

In recognizing macro names, SAS begins when it encounters the ampersand, and ends when it encounters anything that cannot be part of a SAS name (letter, digit, underscore). For example:



- Space, Semicolon, Quotation mark
- **Ampersand** – indicates the start of another macro variable





- **Period** – think of the period as the delimiter for macro variables, in the sense that if SAS encounters a period when reading a macro variable, the period will not be used when compiling the rest of the code

SAS will not resolve macro variables inside single quotation marks. Double quotation marks must be used.

# Practice

The following macros have been defined:

```
%let macr1 = hello ;
%let macr2 = BU ;
%let macr1macr2 = Boston ;
%let ma = 10 ;
%let cr1 = computer ;
%let cr  = soup ;
%let macr   = 24 ;
```



How will SAS resolve the macro references below?

```
%put macr1 ;
%put &macr1 ;
%put &macr1&macr2 ;
%put &macr1.macr2 ;
%put &macr1macr2 ;
%put &ma.cr1 ;
%put ma.&cr.1 ;
%put &macr.1 ;
```

# Macro Programs

A **Macro Program** is a macro component that contains a series of statements. Unlike a macro variable, a macro program can (and almost always does) contain semicolons.



A macro program can be run repeatedly, saving time and ensuring accuracy (once it has been tested).



Macro programs begin with the **%Macro** statement and end with the **%MEND** statement. Everything between these two statements is part of the macro program. The **%Macro** statement is also used to indicate the name of the macro program.



SAS resolves the code inside the macro only when the macro is called, using the ***%macroname*** statement;

# Passing a Value to a Macro Program

Often we want our macro programs to be dynamic, such that the same macro can produce different results according to our purpose.



This can be achieved by combining macro programs and macro variables. If the macro program uses macro variables, then it will produce different results depending on the value of the macro variable.





Passing values to a macro has two parts. In the **%Macro** statement, after the name of the macro, enter the names of the macro variables inside parentheses (separated by commas). When calling the macro, after the name of the macro, enter the values of the macro variables (separated by commas) inside parentheses. The macro program will be executed using the value that was passed to it.

# Developing a Macro Program

Using macros adds an additional layer of complication to your code. If you try to do everything at once, debugging will be very frustrating.

A good strategy is to first write the code for one case without macros.



Once the code works for one case, then use macros to make it dynamic. This approach will make it easier to identify whether errors are due to the macros or to the regular coding.

# Global and Local Macro Variables

A **Global** macro variable is a macro variable that was created in “open code” (i.e., outside of a macro program). Global macro variables can be referenced anywhere.



The **Global Statement** assigns a macro variable as global, regardless of where it is defined:



```
%global mynote ;
```



A **Local** macro variable is a macro variable created inside of a macro program (and not specifically defined as being global). Local macro variables can only be referenced inside of the macro in which they were defined.

# Automatic Macro Variables

The following macro variables are created automatically (either when SAS is opened or when code is run):

- SYSDATE — the current date
- SYSDAY — the day of the week
- SYSVER — the version of SAS being run
- SYSUSERID — the user ID of login



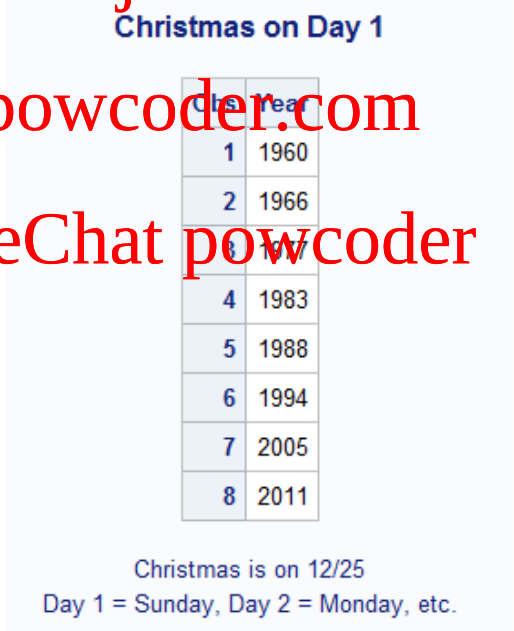
Automatic macro variables work the same as user-defined macro variables, except that they are already defined for you. Automatic macro variables are global.

When creating your own macro variables, don't use names that start with "SYS", so as not to interfere with an automatic macro variable.

# Practice

1. Create a data set that contains the dates from January 1, 1960 through today.
2. Create a macro that prints into the Results Viewer the years for which a particular holiday (combination of month and day) is on a particular day of the week. For example, the years where Christmas (12/25) was on a Sunday:





Christmas on Day 1

| Obs | Year |
|---|---|
| 1 | 1960 |
| 2 | 1966 |
| 3 | 1977 |
| 4 | 1983 |
| 5 | 1988 |
| 6 | 1994 |
| 7 | 2005 |
| 8 | 2011 |

Christmas is on 12/25
Day 1 = Sunday, Day 2 = Monday, etc.

3. Create these data sets:
   - Years when New Years Eve (12/31) was on a Saturday
   - Years when Halloween (10/31) as was on a Friday
   - Years when Independence Day (7/4) was on a Tuesday

# Practice

1. Using the losses.sas7bdat dataset, create a temporary data set called Losses that includes the Year of the loss.
2. Using Proc Means, create a report that summarizes the Non-Weather losses by Year. The result should look like the table on the right (only part of the report is shown).
3. Using the code in #2 above, create a macro that can be used to summarize losses by any metric, using any of the Year, Month, or Qtr functions, and for either Weather or Non-Weather losses. Make your code as flexible as possible.
4. Use your macro to create reports following these requirements:
   - Max of Weather Losses by Month
   - Median of Non-Weather Losses by Quarter
   - Sum of Non-Weather Losses by Weekday
   - Sum of Total Losses by Year (hint: create a new indicator in the Data step with value of 2

**Sum of Losses by Year**

The MEANS Procedure

| Analysis Variable : Amount | | | |
|---|---|---|---|
| Year | LossType | N Obs | Sum |
| 1998 | 0 | 1 | 508 |
| 1999 | 0 | 1 | 2219 |
| 2000 | 0 | 13 | 79475 |
| 2001 | 0 | 12 | 21393 |
| 2002 | 0 | 35 | 180752 |

# Readings

- Textbook section 7.1, 7.2, 7.3, 7.4, 7.5