Introduction to Macros

Jenny Lee

September 21, 2016

Why use Macros

1. If you are writing similar code over and over again, macros may make your life easier
2. If you make a small change in your program, the macro will do it over and over again in your program(s)
3. Macros can make your program “data driven”, meaning SAS decides what to do based on the data (e.g. call symput)

Macro variable: &

* Name of macro variables uses ampersand symbol &
* Contains a single character value

Macro: %

* Name of macro uses percent symbol %
* Includes a data and proc step and macro statement (e.g. %If - %then, %else and %do - %end).
* Sometimes contains macro variable
* Starts with %*[name]*
* End with %mend;

%LET statement

* **Assigns a value to macro variable**
* %Let bikeclass = Mountain;

Proc print data=models noobs

Where class= “&bikeclass”;;

Format price dollar6.;

Title “Current modeds of &bikeclass bicycles”;

Run;

%PUT statement

This just shows you an example of the %LET statement (to make sure it is working).

Adding parameters to macros

%macro *[macro name]* (*parameter1*=, *parameter2*=, …);

*Macro\_text;*

%mend;

*%[macro name]* ([*parameter1]*=[specified parameter], [*parameter2]*=[specified parameter],…);

Options mprint

Options mprint; → This command prints how the %macro is processed in the log window

Conditional logic

%IF - %THEN - %ELSE

%DO - %END

Data-driven program (call symput)

CALL SYMPUT takes a value from the data step and assigns it to a macro variable.

1. CALL SYMPUT (“[*macro variable*]” , *[value]*); → This sets the macro variable
2. &[*macro variable*] → This calls the macro variable