**Lesa’s SAS Macro Programming Cheat Sheet**

A ***macro*** is a collection of SAS code (data steps, procs) that is given a name, and can be executed iteratively by referring to the macro’s name. **Macros literally write SAS code for you.** The benefit using macros is that by replacing constants in the program (i.e., *hard code*) with programming variables (i.e., *macro variables*), you can write SAS code that runs through many repetitions of similar procedures automatically. Your code is shorter, more efficient, less error-prone, and best of all, reusable! It takes longer to initially set-up and debug a macro program than hard-coded syntax, but it will save you substantial time in the long run!!

First, turn on the necessary system options to help you debug your macro programs:

MERROR: show error in log if can’t find macro program called

SERROR: show error in log if can’t find macro variable called

MLOGIC: show values in log of resolved macro variables

MPRINT: print SAS code generated by macros in log

SYMBOLGEN: print values of macro variables in log

Statements that begin with a **%** are part of the macro language. These include **%IF, %THEN, %ELSE, %DO, %DO %WHILE, %DO %UNTIL, %EVAL**, and can be used to direct the macro program to execute or not execute SAS statements conditionally.

Macros are defined as follows:

**%MACRO MyProgram;**

**<SAS code>**

**%MEND MyProgram;**

**%*MyProgram*;**

The sole **%MyProgram** statement at the end instructs SAS to run all commands between **%MACRO MyProgram** and **%MEND MyProgram**. This is referred to as *calling* the macro. We can call the macro multiple times, each time telling it to run a different variant of the SAS code (e.g., same proc, different variable) by changing the value of *macro variables* used in the macro program*.*

**Macro variables:**

* Are NOT in any data sets—they are programming variables and can be used in nearly any syntax!
* Macro names and variables can be up to 32 characters (must start with letter or underscore)
* Values of macro variables are read as text and are case-sensitive
  + **%EVAL** can be used for arithmetic expressions with integers
* Values of macro variables can be defined in 2 ways:
  + With **%LET** statements 🡪 **%LET dv=outcomevar; %LET model=1;**
  + With keyword or positional parameters when the macro is defined and called

***%MyProgram*(dv=outcomevar, model=1);** OR ***%MyProgram*(outcomevar, 1);**

* Macro variables are either **GLOBAL** (default) or **LOCAL** 
  + Global: defined outside of macro; can be used anywhere in the SAS syntax file
  + Local: can only be used in the specific macro program in which is was defined
* Macro variables cannot be *resolved* in single quotes (use double quotes in titles and such)
* Macro variables are referred within a macro program by **&** before the name (**&macrovar.**)
  + References to two macro variables must be separated by a period (turns light green)

**&dv.\_&model.** 🡪 **outcomevar\_1**