Introduction to SAS Macro Language

Bios 524: Biostatistical Computing
Getting Help!

- Use the SAS OnLine Documentation for help on this subject. Follow this path:
  - Base SAS Software
  - SAS Macro Language Reference
    - Introduction – follow these pages to learn about the macro facility
    - Macro Language Dictionary – find help on all macro statements, functions, etc.
What is the SAS Macro Facility?

From the OnLine Doc:

- “The **macro facility** is a tool for extending and customizing the SAS System and for reducing the amount of text you must enter to do common tasks.”

- “The macro facility allows you to assign a name to character strings or groups of SAS programming statements. From that point on, you can work with the names rather than with the text itself.”
What is the SAS Macro Facility?

- From the OnLine Doc:
  - “When you use a macro facility name in a SAS program or from a command prompt, the macro facility generates SAS statements and commands as needed.”
  - “The rest of the SAS System receives those statements and uses them in the same way it uses the ones you enter in the standard manner.”
What is the SAS Macro Facility?

- Two components
  - Macro Processor
    - This compiles your macro and integrates it with your SAS job.
  - Macro Language
    - This is how you communicate with the macro processor.
Triggering the Macro Processor

- Two delimiters will trigger the macro processor in a SAS program
  - &name
    - This refers to a macro variable. The current value of the variable will replace &name.
  - %name
    - This refers to a macro, which may generate a section of a statement, one or more complete SAS statements, or even whole data or proc steps.
Defining and Using Macro Variables

- `%let`

  - Example

    ```sas
    %let keyvar = DOEntry;
    libname library "c:\bios524\classlib";
    proc print data=classlib.clinics;
      id clinicid;
      var &keyvar;
    proc freq data=classlib.clinics;
      tables &keyvar;
    run;
    ```

  _DOEntry_ is assigned to macro variable _keyvar_. Leading and trailing blanks are ignored.

  As the proc step is compiled, _&keyvar_ is replaced with _DOEntry_.

Macro Variable Values

- Values are character strings
- No distinction is made between numeric and character type.
  - However, see the macro function `%eval`.
- Embedded special symbols require the use of a macro quote function when assigning or using macro variables.
  - See macro functions `%str`, `%nstr`, `%quote`, `%nquote`, to mention a few.
Recognizing a Macro Variable

- The key is the leading “&”.
- SAS views &leadvar and &leadvar1 as two different macro variables.
  - %let leadvar = x;
    - &leadvar resolves to x.
    - &leadvar1 is not resolved to x1. An error message may appear.
- When the end of the macro variable is not clear, delimit it with a “.”
  - &leadvar.1 resolves to x1.
  - Note: &leadvar..1 resolves to x.1.
Resolving Macro Variable within Quotes

Example

%Let project = Assignment 4;
Title ‘Results for &project’;
  Resolves to Results for &project.
Title “Results for &project”;  
  Resolves to Results for Assignment 4.

Example

%Let refd = 01JAN2000;
%Let dob = 12APR1955;
age = int((intck("month","&dob"d,"&refd"d) - 
  (day("&refd"d)<day("&dob"d))))/12);
Scope of Macro Variables

- Local versus Global
  - Global variables may be used anywhere in your SAS program after they are defined.
  - Local variables are defined and used within a SAS macro – more about this later.
Global Macro Variables

- Global variables include
  - All automatic macro variables except SYSPBUFF. See Online Doc’s "Macro Language Dictionary" for more information on SYSPBUFF and other automatic macro variables.
  - Macro variables created outside of any macro, such as with a %let.
  - Macro variables created in %GLOBAL statements.
  - Most macro variables created by the CALL SYMPUT routine.
SAS Macros

- Define a SAS macro using the basic syntax
  \[
  \text{%MACRO macro-name;}
  \text{\hspace{1cm}macro definition}
  \text{%MEND macro-name;}
  \]

- Example
  \[
  \text{%macro whereby;}
  \text{\hspace{1cm}where (age ge 18 and eligible=1);}
  \text{\hspace{1cm}by ClinicId;}
  \text{%mend whereby;}
  \]

- Usage:
  \[
  \text{proc print data=Clinics;}
  \text{\hspace{1cm}%whereby}
  \text{run;}
  \]
Producing SAS code with Macros

%DO...%TO; %END;

%macro loopit;
  %let var1 = Age;
  %let var2 = Height;
  %let var3 = Weight;
  %do i = 1 %to 3;
    proc means;
      var &&var&i;
      Title "Analysis for the Variable &&var&i";
    %end;
  %end;
%mend loopit;

data one;
  input age height weight @@;
  datalines;
  34 60 130 45 70 201 50 68 188
 ;
%loopit
run;

This generates:
proc means;
var Age;
Title "Analysis for the Variable Age";
proc means;
var Height;
Title "Analysis for the Variable Height";
proc means;
var Weight;
Title "Analysis for the Variable Weight";

First time through the loop:
1. Resolves to &var1
2. Resolves to Age
Producing SAS code with Macros

%IF...%THEN; %ELSE;

%macro wantrslt;
   %let results = %upcase(&giverslt);
   %if &results = YES %then %do;
      proc means;
      var _numeric_;
      Title "Results for Numeric Variables";
   %end;
   %else %put No results requested; %* Appears in log;
%mend wantrslt;

%let giverslt = NO;
%wantrslt

Convert to upper case
Places text in SAS log.
Macro comment
Passing Parameters to Macros

- Character values may be passed to parameters that are local macro variables.

- Syntax

  ```
  %MACRO macro-name (parm1, parm2, ..., parmk);
  macro definition
  %MEND macro-name;
  ```
Passing Parameters to Macros

Example

%macro wantrslt (giverslt);
   %let results = %upcase(&giverslt);
   %if &results = YES %then %do;
      proc means;
      var _numeric_
      Title "Results for Numeric Variables"
   %end;
   %else %put No results requested; /* Appears in log;*/
%mend wantrslt;

%wantrslt(no);
%wantrslt(no);
%wantrslt(yes);

Local macro variable \textit{giverslt} is defined.

Values are passed to the local macro variable \textit{giverslt}.
Passing Parameters to Macros: An Alternative Method

- Character values may be passed to named parameters.
  - The named parameters may be placed in any order.
  - If omitted, the parameter receives a default value (that may be null).

```sas
%MACRO macro-name (parm1=deflt1, parm2=deflt2, …, parmk=defltk);
  macro definition
%MEND macro-name;
```
Passing Parameters to Macros: An Alternative Method

Example

```sas
%macro agecalc (dob=, refd=01JAN2000);
  %if &dob= %then %do;
    %put Date of birth is missing;
    age = .;
  %end;
  %else
    age = int((intck("month","&dob"d,"&refd"d) -
                (day("&refd"d)<day("&dob"d))))/12);
%mend agecalc;
```
Local Macro Variables

- A local macro variable is defined within a macro if
  - It is defined as a macro parameter.
  - It is used in a %LOCAL statement.
  - It is defined within the macro using a macro statement, assuming the variable does not already exist globally or a %GLOBAL statement is not used.
Assign a library reference to the directory that will hold the macro catalog
- Libname mymacs "c:\bios524\sasmacros";

Assign a file reference to the macro catalog (will create the catalog)
- Filename mymacros catalog "mymacs.stat524macros";

Set system options
- Options mstored=yes sasmstored=mymacs;
Compiling and Storing Macros

- Add the **store** option to the `%macro` statement.
  - `%macro example / store;
  - Run the macro to compile and store it.
  - A catalog named **Sasmacr** will be created in directory referred to by **mymacs**. This will contain the macros you compile and store.
Good Ideas about Stored Macros

- Store your macro source code in the same directory as your macro catalog. Use the file name extension *.sas*. You cannot reconstruct source code from compiled code.

- Define any macro variables used in your compiled macros as local using the %Local command. This avoids changing macros with the same name in the rest of your program.
Using Stored Compiled Macros

- Point to the directory containing your macro catalog and set the system options.
  - Libname mymacs "c:\bios524\sasmacros";
  - Options mstored=yes sasmstore=mymacs;
- Use the macro in your program.
Macro Error Messages and Debugging

- OnLine Documentation
  - Errors
    http://views.vcu.edu/ucsmcv/sas/sashtml/macro/z1302436.htm
  - Debugging
    http://views.vcu.edu/ucsmcv/sas/sashtml/macro/z1066200.htm
Select Macro Functions and Call Routines

- **CALL SYMPUT**(macro-variable, value);
  - [http://views.vcu.edu/ucsmcv/sas/sashtml/macro/z0210266.htm#znid-364](http://views.vcu.edu/ucsmcv/sas/sashtml/macro/z0210266.htm#znid-364)

- **Cautions:**
  - A macro reference resolves when the data or proc step is compiled, but symput assigns a value to the macro variable during execution. Thus you cannot refer to that macro variable in the same step.

- **SYMGET**(argument)
  - [http://views.vcu.edu/ucsmcv/sas/sashtml/macro/z0210322.htm](http://views.vcu.edu/ucsmcv/sas/sashtml/macro/z0210322.htm)
  - Use this to assign the value of a macro variable to a data step variable. This assignment takes place during execution.