STA 311 Statistical Computing and Data Management

Instructor: Cheng Peng, Ph.D.
Department of Mathematics
West Chester University
West Chester, PA 19383

Office: 25 University Avenue, RM 111
Phone: 610-436-2369
Email: cpeng@wcupa.edu

14. Introduction to SAS MACRO facility

What is SAS Macro?

- The SAS Macro language is another language that rests on top of regular SAS code.
- It allows you to assign a name to groups of SAS programming statements. From that point on, you can work with the name rather than with the text itself.
- Macros are particularly useful if you want to make your SAS programs more flexible and allow them to be used in different situations without having to rewrite entire programs

What is SAS Macro?

SAS MACROs consist of collections of

a) regular SAS program statements,

b) macro variables,

c) macro statements and

d) macro functions

contained within a **%MACRO** and a **%MEND**.

Part I: Macro Variables

- can be defined and used anywhere in a SAS program, **except data lines** (i.e. raw data).
- contain one value that **remains constant** until explicitly changed.
- We can also consider a MACRO variable as a string replacement method!

Defining Macro Variables (%LET statement)

%LET *macrovar* = *value*;

- ***macrovar*** : MACRO variable name that follows the regular SAS naming conventions
- ***value*** : any string or macro expression
  - stored as character type
  - length 0-32K characters
  - math expressions are not evaluated
  - quotations not needed
- **leading/trailing blanks are removed**

%LET statement: Examples

Assignment statement

```
%let name = Ed;
%let name2 = ‘Ed’;
```

Macro Variable	Value
name	Ed
name2	‘Ed’

```
%let title = “Ed’s report”;
%let start = 3+4;
```

Macro Variable	Value
title	“Ed’s report”
start	3+4

Using Macro Variables

To use a macro variable, precede the macro variable name with an ampersand (&).

Example: &macrovar

&macrovar will be replaced with whatever text that was assigned to the MACRO variable.

Using Macro Variables: Example1

```
%let var_list = RBC  WBC  cholesterol;

title "Using a Macro Variable List";
proc means data=blood n mean min max ;
   var &var_list;
run;
```

Using Macro Variables: Example

```
%let n = 3;

data generate;
   do Subj = 1 to &n;
      x = int(100*ranuni(0) + 1);
      output;
   end;
run;
```

Combining Macro Variables with Text

- Macro variables can be appended to text or variable names:

```
varname&macrovar
```

- Or prepended:

```
&macrovar.text
```

  you need to insert a . between the macro variable and the text so SAS knows where the text begins.

- Or connected to other macro variables:

```
&macvar1&macvar2
```

Combining Macro Variables with Text (Examples)

```
%let graphics = g;
%let file=blood;

PROC &graphics.plot data=&file;
   PLOT RBC * WBC;
RUN;
```

```
proc gplot data = blood;
   plot RBC * WBC;
run;
```

Combining Macro Variables with Text (Examples)

When concatenating a dataset to a macro variable containing the libname, you need to use double decimal points instead of one.

```
%let lib = d;
data data1;
  set &lib..clinic;
run;
```

```
data data1;
   set d.clinic;
run;
```

Macro Quoting: Examples

macro variables can not resolve single quotes -- **we must use double quotes**.

- **%let sitename = Maryland;**

title1 “site: &sitename”; $\Rightarrow$ title1 “site: Maryland”;

title2 ‘site: &sitename’; $\Rightarrow$ title2 ‘site: &sitename’;

Using SAS automatic Macro Variables: Example

```
title "The Date is &sysdate9 - the Time is
  &systime";
proc print data=blood (obs=5) ;
run;

title 'The Date is &sysdate9 - the
        Time is &systime';
proc print data=blood (obs=5) ;
run;
```

Part II: Macros

“A macro is stored text identified by a name.”

Uses:

- generating repetitive pieces of SAS code
- conditionally executing code

Defining a Macro - 1

Begin with a %MACRO statement and end with a %MEND statement. Coding beginning with % signs is evaluated by the macro processor

Execute the macro with a macro call statement.

```
%MACRO macro-name;
      macro text;
%MEND macro-name;
```
} Define a macro

```
%macro-name
```
} Execute (call) a macro

Defining a Macro - 2

- %MACRO macro-name;
  - begins the macro definition and assigns a name to the macro
- macro text
  - any text
  - regular SAS statements or steps
  - macro variables, functions or statements
  - combination of the above
- %MEND macro-name;
  - ends the macro definition
  - macro-name is optional

Calling a Macro

- %macro-name
  - not a SAS statement, therefore, doesn’t require a semi-colon
  - can be placed anywhere in a program except in data lines
  - Submitting a macro definition complies the macro.
  - If the macro compiles successfully, SAS will run the generated SAS code.

Defining a Macro: Example

```
%let a=blood;
%let x=sex;

%MACRO sortx;
  proc sort data=&a;
  by &x;
  run;

  proc means data=&a;
  by &x;
  run;
%MEND sortx;

%sortx;
```

Part III: Macros with Parameters

We can use %let statement to define a macro variable. Another way to define macro variable is to use parameters.

Parameters pass variable values to macros in the form of macro variables. There are 2 types of parameters:

positional

keyword

Positional Parameters: Syntax

```
%MACRO macro-name (p1, …, pn);
   text
%MEND macro-name;


%macro-name(v1, …, vn)
```

Parameters p1 up to pn.

Values for parameters p1 up to pn.

Positional Parameters -1

The parameter list is defined in the %MACRO statement. This list contains names of macro variables referenced only within the macro.

To call this macro and provide values for the macro parameters you list the values you want to substitute for each parameter in the macro call statement. Values are substituted for the parameter variables using a one-to-one correspondence.

Positional Parameters- 2

Macro parameters:

%MACRO macro-name(p1, …, pn);

- enclosed in parentheses
- separated with commas
- Following SAS naming conventions
- referenced as a macro variable within the macro (&p1, &p2, etc)

Defining a Macro - 3

Macro call values: %macro-name(v1, …, vn)

- enclosed in parentheses
- separated with commas
- are substituted for the parameters using a one-to-one correspondence
  v1 ==> p1
- use commas as placeholders to substitute a null value for positional parameters.

WCU WEST CHESTER UNIVERSITY

Positional Parameters: Example

```
%MACRO sortvar (file,
  byvar);
  proc sort
  data=&file;
    by &byvar;
  run;
%MEND sortvar;

%sortvar (blood, sex)
```

```
proc sort data=blood;
  by sex;
run;
```

Part III: Keyword Parameters

```
%MACRO macro-name (KEY1=, ...,KEYn=);
   text
%MEND macro-name;

%macro-name (KEY1=val1, ...,KEYn=valn)
```

Keyword Parameters - 1

The **keyword parameters** are defined in the %MACRO statement. KEYn is the name of the macro variable followed by an =. VALUE is the default value assigned to the macro variable.

To call this macro and substitute different values you must indicate the macro variable (KEYn) followed by an = and the value you want to substitute. You can specify keyword parameters in any order. Parameters not defined are assigned the default value.

Keyword Parameters - 2

Macro parameters:

%MACRO macro-name(KEY1=value, ..., KEYn=value);

- enclosed in parentheses, separated by commas
- assigned a default value (null is allowed)
- referred to as a macro variable within the macro (&KEYn)
- keyword complies with SAS naming conventions

Keyword Parameters - 3

Macro call values:

%macro-name (KEY1=value, …, KEYn=value)

- enclosed in parentheses, separated by commas
- can be null values, text, macro variable references
- can be listed in **any** order
- can be omitted from the call (default used)

Keyword Parameters: Example

```
%MACRO sortvar (file= mylib.clinic, byvar= gender);
    proc sort data=&file;
            by &byvar;
    run;
%MEND sortvar;


%sortvar (file=blood, byvar=sex)


%sortvar
```

```
proc sort data=blood;
  by sex;
run;

proc sort
    data=mylib.clinic;
  by gender;
run;
```

Mixed Parameter List

```
%MACRO macro-name
                (p1, …, pn, key=value, …,key=value);
    text
%MEND macro-name;

%macro-name
                (v1, …, vn, key=value, …, key=value)
```

- all positional parameters must be listed in order before any keyword parameters

Mixed Parameter List: Example

```
%MACRO sortx4 (LIB, DATANAME, file=&LIB..&dataname,
    byvar=gender);
    proc sort data=&file;
    by &byvar;
    run;

    TITLE "MEANS PROCEDURE OF &DATANAME DATA";
    proc means data=&file;
    by &byvar;
    run;
%MEND sortx4;
```

```
%sortx4 (WORK, blood, byvar=gender);

%sortx4 (d, clinic);
```