SAS: Macros

Computing for Research I
Spring 2011
January 19, 2011

What are Macros and why do we need /use them?

- A macro is a way to automate a task that you perform repeatedly or on a regular basis.
- It is a series of commands and actions that can be stored and run whenever you need to perform the task.
- Can you think of situations where you may want or have used a macro?

SAS Procs

- In essence all of the PROC statements in SAS are macros. They have been written, validated and incorporated into the system so that they can be used repeatedly when called to perform specific tasks.
- Example: PROC means

PROC means

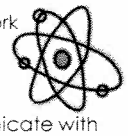
- Provides data summarization tools to compute descriptive statistics for variables across all observations and within groups of observations.
  - Calculates descriptive statistics based on moments
  - Estimates quantiles, including the median
  - Calculates confidence limits for the mean
  - Indentifies extreme values
  - Performs a t test

What is the macro facility?

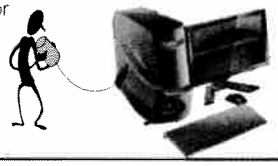
- A tool for extending and customizing SAS and for reducing the amount of text you must enter to do common tasks.
- Enables you to assign a name to character strings or groups of SAS programming statements.

Components of macro facility:

- The macro processor
  - The portion of SAS that does the work



- The macro language
  - The syntax that you use to communicate with the macro processor



How do we "trigger" the SAS macro processor?

- *&name*
  - Refers to a macro variable
  - The form *&name* is called a macro variable reference.
- *%name*
  - Refers to a macro
  - The form *%name* is called a macro call.

What we will cover:

- Macro Variables and Using them for specific functions
  - SAS defined macro variables
  - How do we use these in conjunction with SAS code?
- Creating SAS Code Using Macros
  - Defining SAS macros
  - SYNTEX for SAS macros
  - Calling SAS macros
- Applications
- Examples

Macro Variables

- An efficient way of replacing text strings in SAS code.
- Can be defined within a macro definition or within a statement that is outside a macro definition, referred to as OPEN code.
- Are independent of SAS data set variables.

Macro Variables defined by SAS

- When you invoke SAS, the macro processor creates automatic macro variables that supply information related to the SAS session.

Macro Variables defined by SAS

Some automatic SAS macro variables

SYSCMD	LAST NON-SAS COMMAND ENTERED
SYSDATE	CURRENT DATE IN DATE6. OR DATE7. FORMAT
SYSDAY	CURRENT DAY OF THE WEEK
SYSDEVIC	CURRENT GRAPHICS DEVICE
SYSDSN	LAST SAS DATASET BUILT
SYSINDEX	NUMBER OF MACROS STARTED IN JOB
SYSINFO	SYSTEM INFORMATION GIVEN BY SOME PROCS
SYSPROD	INDICATES WHETHER A SAS PRODUCT IS LICENSED
SYSSCP	OPERATING SYSTEM WHERE SAS IS RUNNING
SYSTIME	STARTING TIME OF JOB
SYSVER	SAS VERSION

Macro Variables defined by SAS

- To use an automatic macro variable, reference it with an ampersand followed by the macro variable name.
  - Example: SYSDATE and SYSDAY
    - SYSDATE contains a SAS date value in the DATE7. format, which displays a two-digit date, the first three letters of the month name, and a two-digit year.
    - SYSDAY contains a SAS day value

Macro Variables defined by Users

- Scope – Global vs Local
  - %GLOBAL – used to define a global macro variable
- Naming conventions

Macro Variables defined by Users

%LET

How do we use this?

%LET CITY=CHARLESTON;

Macro Program Statements

- %DO
- %GLOBAL
- %LOCAL
- %MACRO and %MEND

Macro Functions

- Processes one or more arguments and produces a result.
- Can use all macro functions in both macro definitions and open code.
- Examples: (%EVAL, %LENGTH, %UPCASE)

Using and Displaying Macro Variables

- & statement
- %PUT
  - Example:

```
%LET A=2;
%LET B=5;
%LET OPERATOR=+;
%PUT THE RESULT OF &A &OPERATOR &B IS %EVAL(&A &OPERATOR &B).;
```

Notes to consider about macro variables:

- Double quotation marks, i.e. " " vs ' '
- Variable length
- Special characters
- Condition operations
- Complex tasks

Generating SAS Code Using Macros

- Macro Processing
  - Need to define what the macro function will be
  - Process that SAS uses to program macro elements

Defining and Calling Macros

```
%MACRO macro-name;

  macro text


%MEND macro-name;
```

Application in SAS code:

- Vitals data over time
- Summary statistics using PROC TABULATE
  - Categorical variables
  - Continuous variables

Vitals Data over time:

```
DATA FORM05;
SET ALISAH.FORM05;
SUBJECTID=ZSUBJECTID;
MAX_CVPRESS=F05Q02;
WEIGHT=F05Q06;
MAX_SBP=F05Q07;
MAX_DBP=F05Q09;
MIN_DBP=F05Q10;
MAX_HR=F05Q13;

KEEP SUBJECTID ZVISITID MAX_CVPRESS WEIGHT MAX_SBP MAX_
    MAX_HR;
RUN;

PROC SORT DATA=FORM05 OUT=FORM05T;BY SUBJECTID ZVISITID;
RUN;
```

Vitals Data over time:

```
*MACRO TO CREATE MULTIPLE DATA SETS-ONE FOR EACH DAY OF VITALS;
%MACRO VITALS(DAY, VISIT);
DATA &DAY._VITAL (KEEP=SUBJECTID MAX_CVPRESS& WEIGHT&DAY
    MAX_SBP&DAY  MAX_DBP&DAY MIN_DBP&DAY MAX_HR&DAY);
SET FORM05T;
IF ZVISITID=&VISIT;
MAX_CVPRESS&DAY=MAX_CVPRESS;
WEIGHT&DAY=WEIGHT;
MAX_SBP&DAY=MAX_SBP;
MAX_DBP&DAY=MAX_DBP;
MAX_HR&DAY=MAX_HR;
```

Vitals Data over time:

```
LABEL MAX_CVPRESS&DAY="MAXIMUM CENTRAL VENOUS PRESSURE&DAY.";
LABEL WEIGHT&DAY="WEIGHT&DAY.";
LABEL MAX_SBP&DAY="MAX SYSTOLIC BLOOD PRESSURE&DAY.";
LABEL MAX_DBP&DAY="MAX DIASTOLIC BLOOD PRESSURE&DAY.";
LABEL MAX_HR&DAY="MAX HEART RATE&DAY.";
RUN;

PROC SORT DATA=&DAY._VITAL; BY SUBJECTID; RUN;
%MEND VITALS;
```

Vitals Data over time:

```
*CALLING MACRO TO CREATE THE DATA SETS FOR VITALS:
%VITALS (BASE, 1);
%VITALS (D1, 2);
%VITALS (D2, 3);
%VITALS (D3, 4);
%VITALS (D4, 5);
%VITALS (D5, 6);
%VITALS (D6, 7);
%VITALS (D7, 8);
```

Vitals Data over time:

```
/*MACRO TO CREATE NEW VARIABLES TO BE USED IN MERED DATA
   SET TO CALCULATE DIFFERENCE EACH DAY FROM BASELINE*/

%MACRO CHANGES(NEWVAR, VAR1, VAR2, VITAL, DAY);
IF &VAR1 NE . AND &VAR2 NE . THEN %DO;
   &NEWVAR=&VAR1-&VAR2;
   %END;
LABEL &NEWVAR="CHANGE IN &VITAL. FROM BASELINE TO &DAY.";
%MEND CHANGES;
```

PROC TABULATE for Categorical Variables:

```
*CATEGORICAL MACRO BY TREATMENT GROUP;
%MACRO CATEGORY(INDATA, MYVAR, MYTITLE);
DATA INDATA;SET &INDATA; RUN;
PROC TABULATE DATA=INDATA MISSING FORMAT=7.0 STYLE=[FONT_SIZE=1.5pt];
  CLASS &MYVAR TREATMENTCODE  STYLE=[FONT_SIZE=1.5 FOREGROUND=BLACK
     BACKGROUND=WHITE] ;
  TABLE ALL="ALL SUBJECTS" &MYVAR='  '
                              ,TREATMENTCODE ALL="ALL GROUPS"*(N PCTN<&myvar
     ALL*F=PCTFMT7. ;
      BOX=[LABEL="&MYTITLE"
      STYLE=[FONT_SIZE=1.5 FOREGROUND=BLACK BACKGROUND=WHITE]
      MISSTEXT='0' PRINTMISS;
  LABEL TREATMENTCODE="TREATMENT GROUP"
  KEYLABEL N="N" PCTN="PERCENT"
  CLASSLEV &MYVAR TREATMENTCODE /STYLE=[FONT_SIZE=1.5 FOREGROUND=BLACK
     BACKGROUND=WHITE]
  KEYWORD ALL/PCTN /STYLE=[FONT_SIZE=1.5 FOREGROUND=BLACK
     BACKGROUND=WHITE]
RUN;
%MEND
```

		TREATMENT GROUP						ALL GROUPS	
		TIER 1 (0.625 G/KG)		TIER 2 (1.25 G/KG)		TIER 3 (1.875 G/KG)			
		N	PERCENT	N	PERCENT	N	PERCENT	N	PERCENT
GENDER	ALL SUBJECTS	20	100.0%	20	100.0%	7	100.0%	47	100.0%
	MALE	5	25.0%	7	35.0%	1	14.3%	13	27.7%
	FEMALE	15	75.0%	13	65.0%	6	85.7%	34	72.3%

PROC TABULATE for Categorical Variables:

```
%MACRO CONTINUOUS(INDATA, MYVAR, MYTITLE);
PROC TABULATE DATA=&INDATA FORMAT=10.0 STYLE={FONT_SIZE=1.5};
  VAR &MYVAR/STYLE={FONT_SIZE=1.5 FOREGROUND=BLACK BACKGROUND=WHITE};
  CLASS _TREATMENTCODE /STYLE={FONT_SIZE=1.5 FOREGROUND=BLACK
     BACKGROUND=WHITE};
  TABLE ALL='ALL RANDOMIZED' &MYVAR*{IN='TOTAL_N' MEAN*F=6.2 STD='SD'*F=4.2
     MEDIAN MIN MAX}, (_TREATMENTCODE ALL='ALL GROUPS')
       /BOX={LABEL='&MYTITLE'
       STYLE={FONT_SIZE=1.5 FOREGROUND=BLACK BACKGROUND=WHITE}};
  KEYLABEL N=' ';
  CLASSLEV _TREATMENTCODE /STYLE={FONT_SIZE=1.5 FOREGROUND=BLACK
     BACKGROUND=WHITE};
  KEYWORD ALL 'N' MEAN STD MEDIAN MIN MAX
       /STYLE={FONT_SIZE=1.5 FOREGROUND=BLACK BACKGROUND=WHITE};
RUN;
%MEND;
```

		TREATMENT GROUP			ALL GROUPS
		TIER 1 (0.625 G/KG)	TIER 2 (1.25 G/KG)	TIER 3 (1.875 G/KG)	
AGE AT BASELINE	TOTAL N	20	20	7	47
	Mean	50.80	51.45	53.57	51.49
	SD	15.2	11.4	13.0	13.1
	Median	51	51	55	51
	Min	25	33	38	25
	Max	79	77	75	79