Getting started Single variable insertion Macro modules SAS Macros Shannon Pileggi **STAT 330** STAT 330: Lecture 5 1 / 25 Getting started Single variable insertion Macro modules

OUTLINE

Getting started

Single variable insertion

Macro modules

 ✓□ > ✓♂ > ✓ 差 > ✓ 差 > 差
 シ へ ○

 STAT 330: Lecture 5
 2 / 25

SAS macros

STAT 330: Lecture 5

- ► The SAS macro facility is a *text processing facility*
- ▶ It allows us to insert/include line(s) of SAS code anywhere in the entire program
- ► This provides a **very** convenient way to automate many processes
- ► It is very much like having a handy recording of SAS code which you can play back whenever you need
- ▶ Macros can be broken down into two main types:
 - single variable insertion
 - multiple lines insertion

 Getting started
 Single variable insertion
 Macro modules

 0●00
 00000000
 000000000

The Macro Processor and the Standard SAS Processor

- ▶ In the presence of macro code, SAS will go through an initial scan of your code and 'resolve' any macros first.
- ► After the initial scan, the appropriate line(s) of SAS code are 'inserted' (it's like a program that writes a program)
- ▶ Finally, SAS compiles the full code and executes as usual

 Getting started
 Single variable insertion
 Macro modules

 ○○●○
 ○○○○○○○
 ○○○○○○○○○

Macro triggers

When a SAS program is submitted, two token sequences are recognized as *macro triggers*:

- 1. &name-token a macro variable reference
- 2. %name-token a macro statement, function, or call

(A token is a fundamental unit of text.)



STAT 330: Lecture 5

Getting started

Single variable insertion

Macro modules

Getting started

Single variable insertion

Macro modules

 Getting started
 Single variable insertion
 Macro modules

 000●
 00000000
 0000000000

Debugging macros

Items that are underlined represent the default SAS settings:

- ► MERROR | NOMERROR issues a warning in the log window when attempting to invoke a macro that does not exist.
- ► <u>SERROR</u> | NOSERROR issues a warning in the log window when attempting to use a macro variable that does not exist.
- ► MLOGIC | <u>NOMLOGIC</u> prints (in the log window) details of every macro step.
- ► MPRINT | NOMPRINT prints (in the log window) details of what SAS ultimately "sees" during the Standard SAS Processor stage.
- ► SYMBOLGEN | <u>NOSYMBOLGEN</u> prints (in the log window) the resolved values of any macro variables.

Use the following to ensure ALL your macro tools are made available to you:

0000000

OPTIONS MPRINT MLOGIC SYMBOLGEN;

STAT 330: Lecture 5

Getting started

←□ → ←□ → ← ≣ → □
 ←○ ←○
 6 / 25

Single variable insertion Macro modules

Macro Variable: The Single Variable Insertion

- Macro variables have a single value and do not belong to a data set
- ▶ When reference, macro variable names are prefixed with an ampersand (&)
- ▶ All macro variables are stored as *character* based variables
- ➤ You may name a macro variable whatever you wish, but do not use sys as the first three letters of a macro variable. Such variables are reserved for special purposes.
- ▶ A macro variable may have a **global scope** (can be used any where in the code) or a **local scope** (used only in a macro).

< □ ▷ < ② ▷ < 호 ▷ < 호 ▷ · 호 · ♡ Q ♡

8 / 25

7 / 25

Single variable insertion Getting started Macro modules 0000000

Automatic macro variables

- ▶ SAS has automatic macro variables that begin with the prefix
- ▶ http:

```
//support.sas.com/documentation/cdl/en/mcrolref/
61885/HTML/default/viewer.htm#a003167023.htm
```

```
___ SAS Code _____
TITLE "Contents of Baseball Data on &sysdate9";
PROC CONTENTS DATA = sashelp.baseball VARNUM ;
RUN ;
TITLE ;
           _____ SAS Code _
```

STAT 330: Lecture 5

```
◆□▶ ◆周▶ ◆重▶ ◆重 ◆ の○○
                    9 / 25
```

Single variable insertion Getting started Macro modules 00000000

Macro Variable: The Single Variable Insertion

- ▶ To create a basic macro variable we use %LET macro_variable_name = value ;
- ▶ %LET statements are valid in open code (any where in SAS program)
- ▶ When assigning a macro variable a value
 - do not do %LET ¯o_variable_= value;
 - do not put quotes around the value
- ▶ This is useful for changing values during a SAS program without having to change the entire program itself
- ▶ To use the macro variable that you've created call it with ¯o variable name

4□ > 4回 > 4 亘 > 4 亘 > □ ● 9 Q ○

Getting started Single variable insertion Macro modules 00000000

Resolving macro variables

```
_____ SAS Code _____
LET my_GPA = 3.3;
%LET country = New Zealand;
      _____ SAS Code ___
```

```
SAS Code
             IF GPA = \&my\_GPA;
Resolves to
             IF GPA = 3.3:
```

```
title "Addresses in &country";
SAS Code
Resolves to title "Addresses in New Zealand";
```

| 4 | ロ ト 4 | 日 ト 4 | 巨 ト 9 9 (で 11 / 25

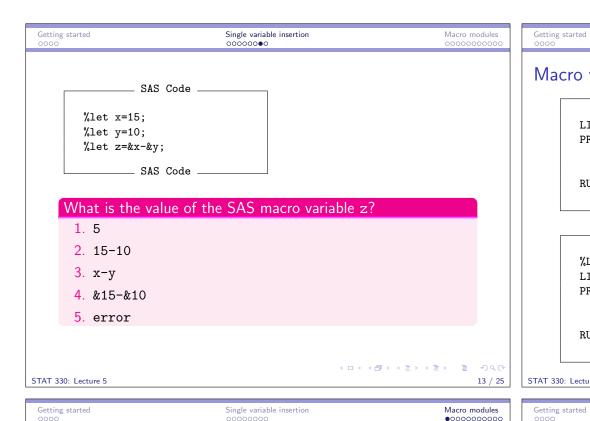
Getting started Single variable insertion Macro modules 00000000

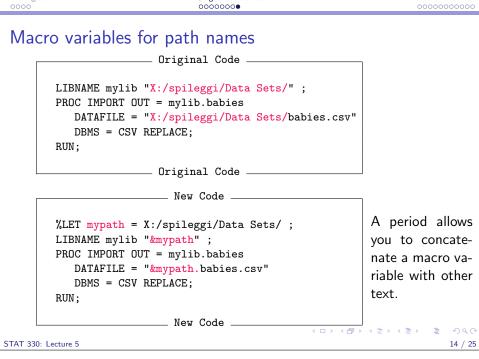
Baseball data

STAT 330: Lecture 5

```
__ SAS Code _____
TITLE "Data = sashelp.basesball, Obs = 10";
PROC PRINT DATA = sashelp.baseball (OBS = 10);
RUN ;
TITLE :
                      _ SAS Code ___
```

On your own: Convert the dataset name of sashelp.basesball and the number of observations printed 10 to macro variables named dsn and num.





Single variable insertion

Macro modules

Macro modules

Getting started

Single variable insertion

Macro modules

STAT 330: Lecture 5

```
Macro Modules: Multiple Lines Insertion

If you ever find yourself writing the same code over and over you should consider using a macro module.

Macro Definition

Macro Execution

Macro Execution

Macro Definition

Macro Execution

Macro Execution
```

Single variable insertion

Macro module, no parameters

```
Macro Definition

%MACRO myprint;
TITLE "DATA = &dsn, OBS = &num";
PROC PRINT DATA = &dsn (OBS=&num);
RUN;
TITLE;
%MEND;

Macro Execution

%LET num = 10;
%LET dsn = sashelp.baseball;
%myprint;
%myprint;
%myprint;
```

4□ ▷ ◆□ ▷ ◆□ ▷ ◆□ ▷ ◆○ ○
 17 / 25

STAT 330: Lecture 5

Getting started

Single variable insertion

Macro modules

Macro module, keyword parameters

```
Macro Definition

%MACRO myprint(dsn = sashelp.baseball, num = 5);
TITLE "DATA = &dsn, OBS = &num";
PROC PRINT DATA = &dsn (OBS=&num);
RUN;
TITLE;
%MEND;
Macro Definition
```

Macro Executions -

```
%myprint;
%myprint();
%myprint(dsn = sashelp.baseball, num = 5);
%myprint(num = 5, dsn = sashelp.baseball);
%myprint(num = 3);
%myprint(dsn = sashelp.class, num = 3);
$TAT 330: Lecture 5
```

- uses an equal sign in MACRO definition
- sets default values for parameters
- can replace all or some subset of default values
- the order of the parameter values matter does not matter

章 ▶ ◆ 臺 ▶ ■ ◆ 夕 ○ ○ 19 / 25

Getting started Single variable insertion Macro modules

Macro module, positional parameters

```
Macro Definition

Macro Executions

Macro Executions
```

- ▶ no equal sign in MACRO definition
- ▶ the parameter values match the order in which they are listed in the macro definition
- ▶ the order of the parameter values matters

←□ → ←□ → ← □ → ← □ → □
 18 / 25

STAT 330: Lecture 5

Getting started

Single variable insertion

Macro modules

Developing Macro Applications

Follow these steps to create and de-bug your SAS macros:

- 1. Write and debug a SAS program without macro coding.
- 2. Generalize the program by replacing hardcoded values with macro variable references.
- 3. Create a macro definition with macro parameters.
- 4. Add macro-level programming for conditional and iterative processing.

 4 □ ▶ 4 □ ▶ 4 □ ▶ 4 □ ▶ □
 20 / 25

Tips and warnings

Tips:

► ALWAYS include the macro debugging options in your SAS program when writing macros

OPTIONS MPRINT MLOGIC SYMBOLGEN;

▶ With these options, you should be able to see what values a macro parameter resolves to. Another way is with %PUT, which prints text to the LOG.

Warnings:

- ▶ use the * ... *\ commenting style when coding macros
- use double quotations (instead of single quotations) when calling macro variable names

21 / 25

◆□▶◆□▶◆■▶◆■▶ ■ 900

23 / 25

STAT 330: Lecture 5

Getting started Single variable insertion Macro modules 00000000000

Macro Conditional Logic

- ▶ We can use conditional logic **outside of data steps** within macros using %IF, %THEN, %DO -- %END, %ELSE
- ▶ These statements work like their counterparts IF, THEN, DO -- END, ELSE
- ► These conditional logic statements
 - can only be used within a macro module
 - ▶ are 'seen' only during the initial macro resolution scanning
 - are NOT included into the SAS code itself

```
Getting started
                             Single variable insertion
                                                                Macro modules
                                                                00000000000
                       %LET month = January:
   Which of the following produces the title
    The month is January?
     1. title "The month is &month";
     2. title 'The month is &month';
     3. title "The month is %month":
     4. title 'The month is %month';
```

Getting started Single variable insertion Macro modules 00000000000

22 / 25

24 / 25

%DO loop

STAT 330: Lecture 5

- ▶ The %DO ... %TO statement allows you to perform loops inside a macro.
- %DO loops in macros have the same kind of structure as standard DO loops in regular SAS code.
- ▶ As with the conditional logic statements in macros [i.e. %IF ... %THEN], these statements **must** be embedded within a macro module – they cannot be placed outside of a macro module and in "open SAS code".

Getting started Single variable insertion Macro modules 0000000000

Using CALL SYMPUTX

Using CALL SYMPUTX allows you to take a value from the data step and assign it to a macro variable.

CALL SYMPUTX("macrovariablename" ,value)

- ▶ macrovariablename **must** be surrounded by quotes
- ▶ value can be
 - ▶ a string in quotes (character or numeric)
 - ▶ the name of a variable that SAS will use to assign a value (in this case, do NOT use quotes)

