SAS Macro #1
An Introduction

JC Wang

Western Michigan University
Department of Statistics
Outline

1. Macros
   - Introduction

2. Macro Statements
   - Special Characters
   - Macro Statement Syntax
   - More on Macro References
What is it

SAS macros provide
- facilities to customize codes
- tools to create user-defined *pseudo* subroutines/functions
- ways to develop user-friendly utilities (such as utility giving number of observations of a SAS data set among others)
What is it

SAS macros provide

- facilities to customize codes
- tools to create user-defined *pseudo* subroutines/functions
- ways to develop user-friendly utilities (such as utility giving number of observations of a SAS data set among others)
What is it

SAS macros provide
- facilities to customize codes
- tools to create user-defined *pseudo* subroutines/functions
- ways to develop user-friendly utilities (such as utility giving number of observations of a SAS data set among others)
SAS macros provide

- facilities to customize codes
- tools to create user-defined *pseudo* subroutines/functions
- ways to develop user-friendly utilities (such as utility giving number of observations of a SAS data set among others)
Why use it

Why one would use SAS macros?

- (#1 important reason) Hold variable values across data steps and procedure steps.
- Save time/effort by replacing repetitive codes (data steps, procedure steps, stand-alone statements, etc.) using macro facility.
- To extensively customize error checking and quality assurance (of codes).
Why use it

Why one would use SAS macros?

* (#1 important reason) Hold variable values across data steps and procedure steps.

* Save time/effort by replacing repetitive codes (data steps, procedure steps, stand-alone statements, etc.) using macro facility.

* To extensively customize error checking and quality assurance (of codes).
Why one would use SAS macros?

- (#1 important reason) Hold variable values across data steps and procedure steps.
- Save time/effort by replacing repetitive codes (data steps, procedure steps, stand-alone statements, etc.) using macro facility.
- To extensively customize error checking and quality assurance (of codes).
Why use it

Why one would use SAS macros?

- (#1 important reason) Hold variable values across data steps and procedure steps.
- Save time/effort by replacing repetitive codes (data steps, procedure steps, stand-alone statements, etc.) using macro facility.
- To extensively customize error checking and quality assurance (of codes).
Outline

1. Macros
   - Introduction

2. Macro Statements
   - Special Characters
   - Macro Statement Syntax
   - More on Macro References
Special Characters

- The percent sign (%) is to execute:
  - macro statements
  - macro (variables and function) definitions
  - macro functions

- The ampersand (&) is to reference macro variable
Special Characters

- The percent sign (%) is to execute
  - macro statements
  - macro (variables and function) definitions
  - macro functions
- The ampersand (&) is to reference macro variable
Special Characters

- The percent sign (%) is to execute
  - macro statements
  - macro (variables and function) definitions
  - macro functions

- The ampersand (&) is to reference macro variable
Special Characters

- The percent sign (%) is to execute
  - macro statements
  - macro (variables and function) definitions
  - macro functions
- The ampersand (&) is to reference macro variable
Special Characters

- The percent sign (%) is to execute
  - macro statements
  - macro (variables and function) definitions
  - macro functions

- The ampersand (&) is to reference macro variable
Special Characters

- The percent sign (%) is to execute
  - macro statements
  - macro (variables and function) definitions
  - macro functions

- The ampersand (&) is to reference macro variable
Macro Statement Syntax

- Macro definitions are enclosed within `%MACRO` and `%MEND` and are executed using the percent sign (%) followed immediately by macro name.
- Some macro statements can also be used in open codes (i.e., beyond macro definitions), such as `%LET`, `%GLOBAL` or `%PUT.`
Macro Statement Syntax

- Macro definitions are enclosed within `%MACRO` and `%MEND` and are executed using the percent sign (%) followed immediately by macro name.

- Some macro statements can also be used in open codes (i.e., beyond macro definitions), such as `%LET`, `%GLOBAL` or `%PUT.`
Macro Statement Syntax

- Macro definitions are enclosed within `%MACRO` and `%MEND` and are executed using the percent sign (%) followed immediately by macro name.

- Some macro statements can also be used in open codes (i.e., beyond macro definitions), such as `%LET`, `%GLOBAL` or `%PUT.`
Macro Variable References

- Statements that create macro variables:
  - `%LET x = someCharValue;`
  - `%DO i = 1 %TO &x;`  
  - `CALL SYMPUT (in data step)`

- The ampersand (&) references the macro variable:
  - `&SomeMacVar`
  - `&var&i`
  - `&&var`
  - `SAS makes multiple passes in resolution of macro references, && is to be resolved to &`
Macro Variable References

- **Statements that create macro variables:**
  - `%LET x = someCharValue;`
  - `%DO i = 1 %TO &x;`
  - `CALL SYMPUT (in data step)`

- **The ampersand (&) references the macro variable:**
  - `&SomeMacVar`
  - `&var&i`
  - `&&&var`
  - `SAS makes multiple passes in resolution of macro references; && is to be resolved to &`
Macro Variable References

- Statements that create macro variables:
  - `%LET x = someCharValue;`
  - `%DO i = 1 %TO &x;`
  - CALL SYMPUT (in data step)

- The ampersand (&) references the macro variable:
  - `&SomeMacVar`
  - `&var&i`
  - `&&var`
  - `&&&var`
  - SAS makes multiple passes in resolution of macro references; `&&&` is to be resolved to &
Macro Variable References

- Statements that create macro variables:
  - `%LET x = someCharValue;`
  - `%DO i = 1 %TO &x;`
  - CALL SYMPUT (in data step)

- The ampersand (&) references the macro variable:
  - `&SomeMacVar`
  - `&var&i`
  - `&&&var`
  - SAS makes multiple passes in resolution of macro references, `&&` is to be resolved to `&`
Macro Variable References

- Statements that create macro variables:
  - `%LET x = someCharValue;`
  - `%DO i = 1 %TO &x;`
  - `CALL SYMPUT (in data step)`

- The ampersand (&) references the macro variable:
  - `&SomeMacVar`
  - `&&var&i`
  - `&&&var`
  - `SAS makes multiple passes in resolution of macro references, && is to be resolved to &.`
Macro Variable References

- Statements that create macro variables:
  - `%LET x = someCharValue;
  - `%DO i = 1 %TO &x;
  - CALL SYMPUT (in data step)

- The ampersand (&) references the macro variable:
  - &SomeMacVar
  - &var&i
  - &/&var
  - SAS makes multiple passes in resolution of macro references, && is to be resolved to &
Macros

Macro Variable References

- Statements that create macro variables:
  - %LET x = someCharValue;
  - %DO i = 1 %TO &x;
  - CALL SYMPUT (in data step)

- The ampersand (&) references the macro variable:
  - &SomeMacVar
  - &var&i
  - &&var
  - SAS makes multiple passes in resolution of macro references, && is to be resolved to &
Macro Variable References

- Statements that create macro variables:
  - `%LET x = someCharValue;`
  - `%DO i = 1 %TO &x;`
  - `CALL SYMPUT (in data step)`

- The ampersand (`&`) references the macro variable:
  - `&SomeMacVar`
  - `&&var&i`
  - `&&&var`

  SAS makes multiple passes in resolution of macro references, `&&&var` is to be resolved to `&`
Macro Variable References

- **Statements that create macro variables:**
  - `%LET x = someCharValue;`
  - `%DO i = 1 %TO &x;`
  - CALL SYMPUT (in data step)

- **The ampersand (&) references the macro variable:**
  - &SomeMacVar
  - &var&i
  - &&&var

  SAS makes multiple passes in resolution of macro references, && is to be resolved to &
Macro Variable References

- Statements that create macro variables:
  - %LET x = someCharValue;
  - %DO i = 1 %TO &x;
  - CALL SYMPUT (in data step)

- The ampersand (&) references the macro variable:
  - %SomeMacVar
  - &var&i
  - &&var

- SAS makes multiple passes in resolution of macro references, && is to be resolved to &
Macro Reference Examples

- `%LET var1 = gender; %LET i = 1;
  &&&var&i is resolved to &var1 in first pass
  then is resolved to gender in second pass
- `%LET grpVar = gender; %LET analVar=ERA;
  %LET var = grpVar;
  &&&var is resolved to &grpVar in first pass
  then is resolved to gender
  %LET var = analVar;
  then &&&var is resolved to &analVar in first pass
  then is resolved to ERA
Macro Reference Examples

- `%LET var1 = gender; %LET i = 1;
  &&var&i is resolved to &var1 in first pass
  then is resolved to gender in second pass

- `%LET grpVar = gender; %LET analVar=ERA;
  %LET var = grpVar;
  &&&var is resolved to &grpVar in first pass
  then is resolved to gender
  %LET var = analVar;
  then &&&var is resolved to &analVar in first pass
  then is resolved to ERA
Macro Reference Examples

- `%LET var1 = gender; %LET i = 1;`  
  &&var&i is resolved to &var1 in first pass  
  then is resolved to gender in second pass

- `%LET grpVar = gender; %LET analVar=ERA;`  
  %LET var = grpVar;  
  &&&var is resolved to &grpVar in first pass  
  then is resolved to gender  
  %LET var = analVar;  
  then &&&var is resolved to &analVar in first pass  
  then is resolved to ERA
More on Macro References

A period (.) is used to connect text to the value of macro variable, examples:

- &somevariable.sometext
- &&var&i..moretext
- libref.&fname.moretext
- lib&i..data&&filenum&j
More on Macro References

A period (.) is used to connect text to the value of macro variable, examples:

- &somevariable.sometext
- &&var&i..moretext
- libref.&fname.moretext
- lib&i..data&&filenum&j
More on Macro References

A period (.) is used to connect text to the value of macro variable, examples:

- &somevariable.sometext
- &&var&i..moretext
- libref.&fname.moretex
- lib&i..data&&filenum&j
More on Macro References

A period (.) is used to connect text to the value of macro variable, examples:

- &somevariable.sometext
- &var&i..moretext
- libref.&fname.moretex
- lib&i..data&&filenum&j
More on Macro References

A period (.) is used to connect text to the value of macro variable, examples:

- &somevariable.sometext
- &&var&i..moretext
- libref.&fname.moretex
- lib&i..data&&filenum&j