SAS Macro Programming for Beginners

Susan J. Slaughter
Avocet Solutions
Lora D. Delwiche
University of California, Davis

What is SAS macro language?

- Programming language for string manipulation
- Strings are characters
- Usually SAS statements or pieces of SAS statements
- Normally considered advanced, but concepts are not difficult

Why use macros?

- Harder to write than standard code
- But can save time and effort
  - Make one change, SAS echoes
  - Reusable code
  - Make programs data driven

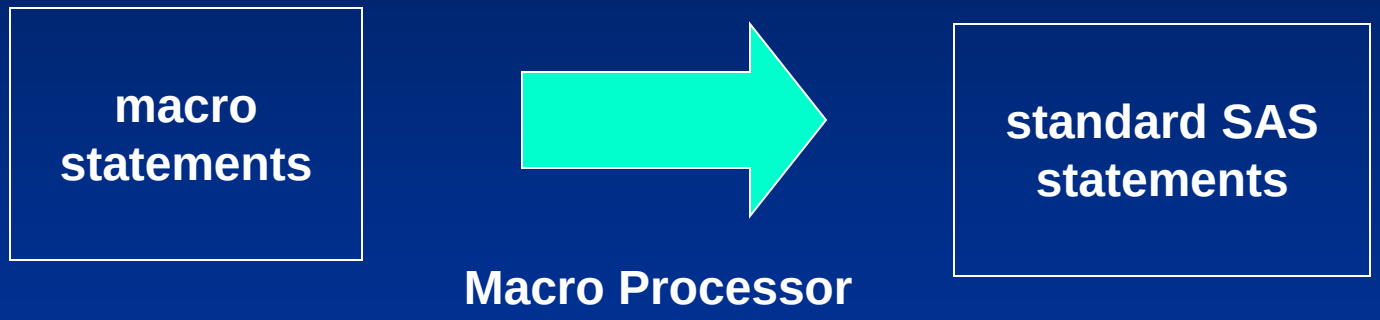

TREK

Did you know?

There are a billion bicycles in the world, twice as many as motorcars.

didyouknow.org/bicycles

SAS Macro Processor



You are writing a program that writes a program.

Macros vs. Macro Variables

Macro variables

- Start with &
- Single character value

Macros

- Start with %
- Piece of a program
- May use macro statements
- Often use macro variables

Scope of macro variables

- Local macro variables
  - Defined within a macro
  - Can only be used in that macro
- Global macro variables
  - Defined in “open code”
  - Can be used anywhere
- Think globally and locally

Use of Quotes

- Macro processor does not check inside ‘single quotes’

```
TITLE ‘Report for &Region’;
```

- Use “double quotes” for code containing macro variables

```
TITLE “Report for &Region”;
TITLE “Report for Northwest”;
```

UC Davis

Source: City of Davis

Substituting text with %LET

- Simplest macro statement
- Assigns a value to a macro variable

```
%LET macro-variable-name = value;
```

Substituting text with %LET

- Define macro variable

```
%LET iterations = 5;
```

- Use macro variable

```
DO i=1 TO &iterations;
```

- Resolves to standard SAS code

```
DO i=1 TO 5;
```

Substituting text with %LET

- Define macro variable

```
%LET winner = Bradley Wiggins;
```

- Use macro variable

```
TITLE "First: &winner";
```

- Resolves to standard SAS code

```
TITLE "First: Bradley Wiggins";
```

Models data

Model	Class	Price	Frame
Black Bora	Track	796	Aluminum
Delta Breeze	Road	699	CroMoly
Jet Stream	Track	1130	CroMoly
Mistral	Road	1995	Carbon Comp
Nor'easter	Mountain	899	Aluminum
Santa Ana	Mountain	459	Aluminum
Scirocco	Mountain	2256	Titanium
Trade Wind	Road	759	Aluminum

Substituting text with %LET

- Define and use macro variable

```
%LET bikeclass = Mountain;
PROC PRINT DATA = models NOOBS;
   WHERE Class = "&bikeclass";
   FORMAT Price DOLLAR6.;
   TITLE "Current Models "
      "of &bikeclass Bicycles";
RUN;
```

Substituting text with %LET

- Resolves to standard SAS code

```
PROC PRINT DATA = models NOOBS;
    WHERE Type = "Mountain";
    FORMAT Price DOLLAR6.;
    TITLE "Current Models "
        "of Mountain Bicycles";
RUN;
```

Current Models of Mountain Bicycles

Model	Class	Price	Frame
Nor'easter	Mountain	$899	Aluminum
Santa Ana	Mountain	$459	Aluminum
Scirocco	Mountain	$2,256	Titanium

American River Parkway

Photo by Rome Aban

What is a macro?

- A group of statements with a name
- To call or invoke a macro
  - Use its name
  - SAS substitutes the statements for the name

Creating modular code

- Define macro

```
%MACRO macro-name;
        macro-text
%MEND macro-name;
```

- Call macro

```
%macro-name
```

Creating modular code

- Define macro

```
%MACRO printit;
  PROC PRINT DATA = models NOOBS;
    TITLE 'Current Models';
    VAR Model Class Frame Price;
    FORMAT Price DOLLAR6.;
  RUN;
%MEND printit;
```

Creating modular code

- Call macro

```
%printit
  PROC SORT DATA = models;
    BY Price;
%printit
```

Creating modular code

- Resolves to standard SAS code

```
PROC PRINT DATA = models NOOBS;
  TITLE 'Current Models';
  VAR Model Class Frame Price;
  FORMAT Price DOLLAR6.;
RUN;
PROC SORT DATA = models;
  BY Price;
PROC PRINT DATA = models NOOBS;
  TITLE 'Current Models';
  VAR Model Class Frame Price;
  FORMAT Price DOLLAR6.;
RUN;
```

Current Models

Model	Class	Frame	Price
Black Bora	Track	Aluminum	$796
Delta Breeze	Road	CroMoly	$699
Jet Stream	Track	CroMoly	$1,130
Mistral	Road	Carbon Comp	$1,995
Nor'easter	Mountain	Aluminum	$899
Santa Ana	Mountain	Aluminum	$459
Scirocco	Mountain	Titanium	$2,256
Trade Wind	Road	Aluminum	$759

Current Models

Model	Class	Frame	Price
Santa Ana	Mountain	Aluminum	$459
Delta Breeze	Road	CroMoly	$699
Trade Wind	Road	Aluminum	$759
Black Bora	Track	Aluminum	$796
Nor'easter	Mountain	Aluminum	$899
Jet Stream	Track	CroMoly	$1,130
Mistral	Road	Carbon Comp	$1,995
Scirocco	Mountain	Titanium	$2,256

Photo by Eric Norris

Adding parameters to macros

- Parameters are macro variables
- Defined in macro

```
%MACRO macro-name
    (parameter-1=, parameter-n=);
   macro-text
%MEND macro-name;
```

Adding parameters to macros

- Define macro

```
%MACRO monthlyreport (month=, region=);
    macro-text
%MEND monthlyreport;
```

- Call macro

```
%monthlyreport (month=May, region=West)
```

Adding parameters to macros

- Define macro

```
%MACRO sortandprint (sortseq=, sortvar=);
  PROC SORT DATA = models;
    BY &sortseq &sortvar;
  PROC PRINT DATA = models NOOBS;
    TITLE 'Current Models';
    TITLE2 "Sorted by &sortseq &sortvar";
    VAR Model Class Frame Price;
    FORMAT Price DOLLAR6.;
  RUN;
%MEND sortandprint;
```

Adding parameters to macros

- Call macro

```
%sortandprint
  (sortseq=Descending, sortvar=Price)
```

Adding parameters to macros

- Resolves to standard SAS code

```
PROC SORT DATA = models;
  BY Descending Price;
PROC PRINT DATA = models NOOBS;
  TITLE 'Current Models';
  TITLE2 "Sorted by Descending Price";
  VAR Model Class Frame Price;
  FORMAT Price DOLLAR6.;
RUN;
```

Current Models
Sorted by Descending Price

Model	Class	Frame	Price
Scirocco	Mountain	Titanium	$2,256
Mistral	Road	Carbon Comp	$1,995
Jet Stream	Track	CroMoly	$1,130
Nor'easter	Mountain	Aluminum	$899
Black Bora	Track	Aluminum	$796
Trade Wind	Road	Aluminum	$759
Delta Breeze	Road	CroMoly	$699
Santa Ana	Mountain	Aluminum	$459

Adding parameters to macros

- Call macro again

```
%sortandprint (sortseq=, sortvar=Class)
```

Adding parameters to macros

- Resolves to standard SAS code

```
PROC SORT DATA = models;
  BY  Class;
PROC PRINT DATA = models NOOBS;
  TITLE 'Current Models';
  TITLE2 "Sorted by  Class";
  VAR Model Class Frame Price;
  FORMAT Price DOLLAR6.;
RUN;
```

Current Models
Sorted by Class

Model	Class	Frame	Price
Scirocco	Mountain	Titanium	$2,256
Nor'easter	Mountain	Aluminum	$899
Santa Ana	Mountain	Aluminum	$459
Mistral	Road	Carbon Comp	$1,995
Trade Wind	Road	Aluminum	$759
Delta Breeze	Road	CroMoly	$699
Jet Stream	Track	CroMoly	$1,130
Black Bora	Track	Aluminum	$796

MPRINT option

- Normally you don’t see resolved macro statements
- To see them use MPRINT system option

```
OPTIONS MPRINT;
```

MPRINT option: SAS log

```
16 OPTIONS MPRINT;
17 %sortandprint(sortseq=, sortvar=Class)

MPRINT(SORTANDPRINT):   PROC SORT DATA=models;
MPRINT(SORTANDPRINT):   BY  Class;

MPRINT(SORTANDPRINT):   PROC PRINT DATA=models NOOBS;
MPRINT(SORTANDPRINT):   TITLE 'Current Models';
MPRINT(SORTANDPRINT):   TITLE2 "Sorted by  Class";
MPRINT(SORTANDPRINT):   VAR Model Class Frame Price;
MPRINT(SORTANDPRINT):   FORMAT Price DOLLAR6.;
MPRINT(SORTANDPRINT):   RUN;
```

Did you know?

Source: *Major Taylor* by Andrew Richie

American Major Taylor won the Bicycling World Championship in 1899 in Montreal.

Conditional Logic

- Increase flexibility of macros
- Use macro statements:

  %IF %THEN %ELSE

  %IF %THEN %DO %END

Conditional Logic

```
%IF condition %THEN action;
    %ELSE %IF condition %THEN action;
    %ELSE action;
```

Conditional Logic

```
%IF condition %THEN %DO;
    action;
%END;
```

%IF vs. IF

- Different from standard IF statement
- Can only be used inside a macro
- These statements won’t appear in standard SAS code
- Remember you are writing a program that writes a program

Automatic Macro Variables

Variable Name	Example	Description
&SYSDATE	01MAR13	Character value of the date that job or session began
&SYSDAY	Friday	Day of the week that job or session began

Orders data

ID	Date	Model	Quantity
287	15FEB13	Delta Breeze	15
287	15FEB13	Santa Ana	15
274	16FEB13	Jet Stream	1
174	17FEB13	Santa Ana	20
174	17FEB13	Nor'easter	5
174	17FEB13	Scirocco	1
347	18FEB13	Mistral	1
287	21FEB13	Delta Breeze	30
287	21FEB13	Santa Ana	25

Conditional logic

- Define macro

```
%MACRO reports;
    %IF &SYSDAY = Monday %THEN %DO;
    PROC PRINT DATA = orders NOOBS;
        FORMAT OrderDate DATE7.;
        TITLE "&SYSDAY Report: "
            "Current Orders";
    %END;
```

Conditional logic

```
%ELSE %IF &SYSDAY = Friday %THEN %DO;
    PROC TABULATE DATA = orders;
        CLASS CustomerID;
        VAR Quantity;
        TABLE CustomerID ALL, Quantity;
        TITLE "&SYSDAY Report: Summary "
            "of Orders";
    %END;
  RUN;
%MEND reports;
```

Conditional logic

- Call macro

```
%reports
```

Conditional logic

- On Monday resolves to

```
PROC PRINT DATA = orders NOOBS;
    FORMAT OrderDate DATE7.;
    TITLE "Monday Report: "
        "Current Orders";
```

Monday Report: Current Orders

Customer ID	Order Date	Model	Quantity
287	15FEB13	Delta Breeze	15
287	15FEB13	Santa Ana	15
274	16FEB13	Jet Stream	1
174	17FEB13	Santa Ana	20
174	17FEB13	Nor'easter	5
174	17FEB13	Scirocco	1
347	18FEB13	Mistral	1
287	21FEB13	Delta Breeze	30
287	21FEB13	Santa Ana	25

Conditional logic

- On Friday resolves to

```
PROC TABULATE DATA = orders;
    CLASS CustomerID;
    VAR Quantity;
    TABLE CustomerID ALL, Quantity;
    TITLE "Friday Report: Summary "
        "of Orders";
```

Friday Report: Summary of Orders

	Quantity
	Sum
CustomerID	
174	26.00
274	1.00
287	85.00
347	1.00
All	113.00

Sacramento Valley

Photo by Eric Norris

Data-Driven Programs

- Let data determine values of macro variables
- Problem—SAS doesn’t see data until execution phase
- Macro variables resolved before execution
- Solution—Use CALL SYMPUT in a DATA step and pass value to a later step

CALL SYMPUT routine

- Used in DATA step
- Assigns a value to a macro variable

```
CALL SYMPUT("macro-variable", value);
```

- Value is name of a variable

CALL SYMPUT routine

- Example value as variable name

```
IF Place = 1 THEN
    CALL SYMPUT("WinningTime", Time);
```

CALL SYMPUT routine

```
PROC SORT DATA = orders;
  BY DESCENDING Quantity;
DATA _NULL_;
  SET orders;
  IF _N_ = 1 THEN
   CALL SYMPUT("biggest", CustomerID);
  STOP;
```

Use Macro Variable

```
PROC PRINT DATA = orders NOOBS;
  WHERE CustomerID = "&biggest";
  FORMAT OrderDate DATE7.;
  TITLE "Customer &biggest Had the "
     "Single Largest Order";
RUN;
```

Data-Driven Program

- Resolves to

```
PROC PRINT DATA = orders NOOBS;
    WHERE CustomerID = "287";
    FORMAT OrderDate DATE7.;
    TITLE "Customer 287 Had the "
        "Single Largest Order";
RUN;
```

Customer 287 Had the Single Largest Order

Customer ID	Order Date	Model	Quantity
287	21FEB13	Delta Breeze	30
287	21FEB13	Santa Ana	25
287	15FEB13	Delta Breeze	15
287	15FEB13	Santa Ana	15

Sacramento Valley

Photo by Jeff Hall

Avoiding problems

- Start simple and build piece by piece
- First write your program in standard SAS code
- Then add macro features one at a time

Conclusions

- Macros can be complicated
- Macros can make your work easier
- Remember you are writing a program that writes a program

Thank you!

Contact author

susan@avocetsolutions.com

Download paper

www.avocetsolutions.com

The Little SAS Book: A Primer

Fifth Edition