FROM %LET TO %LOCAL; METHODS, USE AND SCOPE OF MACRO VARIABLES IN SAS PROGRAMMING

Jay Iyengar is director of Data Systems Consultants LLC. He is a SAS consultant, trainer, and SAS Certified Advanced Programmer. He was the co-leader and organizer of the Chicago SAS Users Group (WCSUG) for the last 5 years.

He's presented papers and training seminars at SAS Global Forum (SGF), and other regional and local SAS users groups. He has been using SAS since 1997. His industry experience includes Health care International Trade, Database Marketing and Educational Testing.

From %LET to %LOCAL; Methods, Use and Scope of Macro Variables in SAS programming

Jay Iyengar, Data Systems Consultants LLC

INTRODUCTION

- SAS macro language
  - Macro variables
- Data Step
  - SAS data set variables – Numeric or Character
- Macro variables
  - Character Strings

PROCESSING FLOW OF SAS CODE

- Flow of SAS code during the compilation phase

- The word scanner breaks code into tokens.

MACRO PROCESSOR

SAS code with macro tokens (%/&) are sent to the macro processor.

INPUT STACK		WORD SCANNER		MACRO PROCESSOR
%LET CAR = MAZDA	→	CAR = MAZDA	→	%LET

TYPES OF MACRO VARIABLES

- AUTOMATIC
- USER-DEFINED

Automatic macro variable	Definition	Values
SYSDATE	The Date of the SAS Invocation (Date7. format)	FIXED
SYSTIME	The Time of the SAS Invocation	FIXED
SYSSCP	The Operating System being used, WIN, HPUX, etc	FIXED
SYSVER	The release of SAS that is being used	FIXED
SYSLAST	The name of the most recently created SAS data set	VARIABLE
SYSPARM	Contains text specified when SAS is invoked	VARIABLE
SYSERR	Return which indicates execution status of SAS code	N/A

%PUT statement

Prints macro variables with values to SAS log

```
73          %Put _Automatic_;
AUTOMATIC SYSDATE 03FEB21
AUTOMATIC SYSDATE9 03FEB2021
AUTOMATIC SYSDAY Wednesday
AUTOMATIC SYSJOBID 18882
AUTOMATIC SYSLAST WORK.VERSION_161238
AUTOMATIC SYSPARM
AUTOMATIC SYSRC 0
AUTOMATIC SYSSCP LIN X64
AUTOMATIC SYSTIME 21:52
AUTOMATIC SYSVER 9.4
```

%PUT KEYWORD	DESCRIPTION
AUTOMATIC	AUTOMATIC MACRO VARIABLES
USER	USER DEFINED MACRO VARIABLES
ALL	ALL MACRO VARIABLES
LOCAL	LOCAL MACRO VARIABLES
GLOBAL	GLOBAL MACRO VARIABLES

SCOPE OF MACRO VARIABLES

- GLOBAL – Used in open code, in a macro, any SAS programs
- LOCAL – used within the macro only
- Global Symbol Table – created at SAS initialization
- Local Symbol Table – created at macro execution

METHODS FOR DEFINING MACROVARIABLES

- %LET
- CALL SYMPUT/CALL SYMPUTX
- PROC SQL INTO clause

%LET statement

- Created during the compilation phase.
- Creates a global macro variable.
- Leading and trailing blanks are removed.

%STR and %NRSTR

- Macro quoting functions

```
%Let Make = Audi;
%Let Model = 5000S;

%Let T1 = %STR (Vonstuben%'s Audi Dealership);
%Let T2 = %NRSTR (Invoice AS A % OF Retail Price);

%Put MAKE=&MAKE T1=&T1 T2=&T2;

Proc Print Data=SASHELP.CARS;
   Var Make Model Type DriveTrain InvoiceP MSRP PCT_MSRP;
   Where Make="&Make" and Model="&Model";
   Title1 "&T1";
   Title2 "&T2";
Run;
```

CALL SYMPUT routine

- Creates macro variables during DATA STEP execution.
- Creates a global macro variable.
- Preserves leading and trailing blanks.

```
DATA _NULL_;
        SET SDATASET1 NOBS=TOBS;
        CALL SYMPUT ('TOT_OBS', TOBS);
RUN;
```

PROC SQL INTO clause

- Creates global macro variables during execution.

```
Proc Sql Inobs=1 Noprint;
        Select Distinct Make Into: Make
        From SASHELP.CARS;
Quit;
```

- Can create multiple macro variables, and a concatenated value list using SEPARATED BY

```
Proc Sql;
    Select MSRP into : PRICELIST Separated By ‘ ‘
    From SASHELP.CARS;
Quit;
```

OTHER METHODS

- %MACRO Statement
  - Parameters – (Keyword or Positional)
  - Local macro variables

```
%MACRO TEST (VARLIST, DSN=SASHELP.CARS, LIB1=WORK);
        TITLE "DATASET: &DSN WITH VARIABLES &VARLIST, IN LIBRARY &LIB1;
%MEND TEST;
```

- %GLOBAL / %LOCAL – macro variables with Null values.

COMPARISON

Advantages and disadvantages to each method

Method	Scope of Variable	Processing	Leading\Trailing Blanks
%LET	GLOBAL	COMPILATION	REMOVED
CALL SYMPUT\SYMPUTX	GLOBAL	EXECUTION	PRESERVED
PROC SQL INTO CLAUSE	GLOBAL	EXECUTION	PRESERVED
%MACRO STATEMENT	LOCAL	COMPILATION	N/A
%GLOBAL\%LOCAL	GLOBAL\LOCAL	COMPILATION	N/A

%LET – inside macro

```
%Macro Autos;
  %Let Make = Audi;
  %Let Model = 5000S;

  %Put Make=&Make Model=&Model;

  Proc Print Data=SASHELP.CARS;
    Var Make Model Type DriveTrain MSRP;
    Where Make="&Make" and Model="&Model";
    Title "Type, Drivetrain, and MSRP &Make &Models Models";
  Run;

  %Put _LOCAL_;

%Mend Autos;

%Autos
```

LOG

```
AUTOS MAKE Audi
AUTOS MODEL 5000S
```

%LET – inside macro (con't)

```
%Macro Autos;

  %Global Make Model;

  %Let Make = Audi;
  %Let Model = 5000S;
  %Put Make=&Make   Model=&Model;

  Proc Print Data=SASHELP.CARS;
        Var Make Model Type DriveTrain MSRP;
       Where Make="&Make" and Model="Model";
        Title "Type, Drivetrain, and MSRP &Make &Model Models";
  Run;

  %Put _GLOBAL_;

%Mend Autos;

%Autos
```

LOG

```
GLOBAL MAKE Audi
GLOBAL MODEL 5000S
```

CALL SYMPUT – inside macro

```
%Macro Vehicle;

  Data _Null_;
          Set SASHELP.Cars (Obs=1);
          Call Symput('Automake', Make);
          Call Symput('Automodel', Model);
  Run;

  %Put Automake=&Automake Automodel=&Automodel;

  Proc Means Data=SASHELP.CARS Mean;
          Var MSRP;
          Class DriveTrain;
          Where Make="&Automake" and Model="&Automodel";
          Title1"Average Price for &Automake &Automodel Models";
          Title2"By Drivetrain";
  Run;

  %Put _Global_;
%Mend Vehicle;
%Vehicle
```

LOG

```
GLOBAL AUTOMAKE Acura
GLOBAL AUTOMODEL MDX
```

CALL SYMPUT – inside macro (con't)

```
%Macro Vehicle;
  %Local Automake Automodel;
          Data _Null_;
            Set SASHELP.Cars (Obs=1);
            Call Symput('Automake', Make);
            Call Symput('Automodel', Model);
          Run;
  %Put Automake=&Automake Automodel=&Automodel;
          Proc Means Data=SASHELP.CARS Mean;
            Var MSRP;
            Class DriveTrain;
            Where Make="&Automake";
            Title1"Average Price for &Automake Models";
            Title2"By Drivetrain";
          Run;
  %Put _LOCAL_;
%Mend Vehicle;
%Vehicle
```

LOG

```
VEHICLE AUTOMAKE Acura
VEHICLE AUTOMODEL MDX
```

PROC SQL INTO – inside macro

```
%Macro Automaker;
  %Local Make Model;
    Proc Sql Inobs=1 Noprint;
          Select Distinct Make, Model Into :Make, :Model
          From SASHELP.CARS;
    Quit;
  %PUT MAKE=&MAKE MODEL=&MODEL;
    Proc Means Data=SASHELP.CARS Mean;
          Var MSRP;
          Class DriveTrain
          Where Make="&Make" and Model="&Model";
          Title1"Average Price for &Make &Model";
          Title2"By Drivetrain";
    Run;
  %Put _LOCAL_;
%Mend Automaker;
%Automaker
```

LOG

```
AUTOMAKER MAKE Acura
AUTOMAKER MODEL MDX
```

COMPARISON

Inside a macro, the scope of a macro variable depends on technique, and the use of parameters.

Method	Scope in Open Code	Scope inside Macro	Scope Inside Macro with Parameters	Scope Inside Macro with %LOCAL or %GLOBAL
%LET	GLOBAL	LOCAL	LOCAL	GLOBAL
CALL SYMPUT\X	GLOBAL	GLOBAL	LOCAL	LOCAL
PROC SQL INTO	GLOBAL	GLOBAL	LOCAL	LOCAL

CONCLUSION

- Knowing capabilities of macro variable methods and their consequences makes your code more versatile.
- Scope is determined by method, but also location in code where macro variable is defined.
- Open code vs. inside macro

Contact Information

Name: Jay Iyengar

Affiliation: Data Systems Consultants LLC

Contact Number:

E-mail: datasyscon@gmail.com

Website:

Twitter:

LinkedIn: https://www.linkedin.com/in/datasysconsult