USES OF SAS® MACRO STATEMENTS - %LET STATEMENT

Reiling Lee, Syntex Research

ABSTRACT

The %LET statement is commonly used to define a macro variable and assign it a value. Some SAS® users may be aware of the problem that a macro variable is unresolved when called outside the macro where it is defined. This paper presents two methods to allow users to call %LET-defined macro variables anywhere they want and avoid the problem of unresolved macro variables. Nine examples are presented. One of them is from the SAS® autolcall library.

INTRODUCTION

A SAS® user can define macro variables by using an iterative %DO statement, a %GLOBAL statement, a %INPUT statement, a %LET statement, a %LOCAL statement, a %MACRO statement, or the SYMPUT routine. However, only %DO, %LET, and %MACRO statements and the SYMPUT routine can assign values to macro variables. For example, %DO THISDAY = 1 %TO 31 assigns the macro variable, THISDAY, values from 1 to 31 iteratively; %LET REILING = FEMALE assigns the value of FEMALE to the macro variable, REILING; %MACRO CREATING(DSN=_LAST_, NEWDSN) defines two macro variables, DSN and NEWDSN, and only assigns DSN the value of _LAST_; CALL SYMPUT('CUTE', 'REILING') creates a macro variable, CUTE, and assigns the value of REILING to it. Among these macro program statements, the %LET statement is the simplest way to create a macro variable and assign it a value. A detailed description of defining macro variables using the %LET statement can be found in the SAS® Guide to Macro Processing (P. 23 - P. 48). This paper simply introduces the use of the %LET statement to those who are new to this subject and presents two ways to let users call a macro variable outside the macro where it is defined.

USES OF %LET STATEMENT

%LET is a SAS® macro statement, which defines a single macro variable or changes the value of an existing one. For example,

%LET TODAY = May 11, 1988;

defines a macro variable, TODAY, and assigns it a value of "May 11, 1988." SAS® substitutes the value of a macro variable wherever the macro variable is used, e.g.,

%PUT Today is &TODAY;

puts "Today is May 11, 1988." in the SAS® log. If more than one value is needed, a macro variable can be created for each value or different values can be assigned to the same macro variable. For example, the following SAS® macro statements:

%LET NAME1 = Wei-wei;
%PUT &NAME1 lives in Delaware.;
%LET NAME2 = Reiling;
%PUT &NAME2 lives in California.;
%PUT &NAME1 and &NAME2 are sisters.;

put 'Wei-wei lives in Delaware., 'Reiling lives in California., and 'Wei-wei and Reiling are sisters.' in the SAS® log. The value of an existing macro variable can be changed in the following way:

%LET NAME = Tan;
%PUT &NAME lives in Florida.;
%LET NAME = Reiling;
%PUT &NAME lives in California.;

which puts 'Tan lives in Florida., 'Reiling lives in California.' in the SAS® log (Note that only one macro variable is defined). The value of a macro variable remains constant until explicitly changed.

PROBLEM

When using a %LET statement to define a macro variable, a SAS® user may sometimes get a warning message saying that it is unresolved. The most common reason for this problem is that the macro variable has been called outside the macro where it is defined. A macro variable defined by a %LET statement inside a macro is a local macro variable, that is, it can be referenced within the macro where it has been defined but is unresolved outside that macro. However, a macro variable defined outside a macro can be called in the macro. How can we use a macro variable defined by a %LET statement inside a macro wherever we want?

SOLUTION

There are at least two ways to make macro variables global. The first is to use the %GLOBAL statement to define the macro variable as global before assigning it a value by a %LET statement. The second is to use a "&macrovariablename" statement, which is useful when the only purpose of the macro is to assign a value to a macro variable. Nine examples are presented here for demonstration:
Example 1: Macro variables are unresolved outside a macro when defined by %LET statements inside the macro:

Code:

```sas
%MACRO MIIJOR;
%IF "&NAME" EO "PE~PEI" %THEN %LET MAJOR = BIOLOGY;
%ELSE_IF "&NAME" EQ "WEI-WEI" %THEN %LET MAJOR = STATISTICS;
%ELSE_IF "&NAME" EQ "REI-CHI" %THEN %LET MAJOR = COMPUTER SCIENCES;
%ELSE_IF "&NAME" EQ "REI-CHANG" %THEN %LET MAJOR = NUTRITION;
%ELSE %IF "&NAME" EQ "REI-CHANG" %THEN %LET MAJOR = INFORMATION SCIENCES;
%MEND MAJOR;

%MACROJOB;
%IF "&MAJOR' EO 'BIOLOGY' %THEN %LET JOB. BIOLOGIST;
%ELSE %IF "&MAJOR" EO "STATISTICS' %THEN %LET JOB = STATISTICIAN;
%ELSE %IF "&MAJOR" EO "COMPUTER SCIENCES' %THEN
%LET JOB = COMPUTER PROGRAMMER;
%ELSE %IF "&MAJOR" EO "NUTRITION" %THEN %LET JOB = DIETITIAN;
%ELSE %IF "&MAJOR" EO "INFORMATION SCIENCES" %THEN
%LET JOB = LIBRARIAN;
%MEND JOB;

%LET NAME = PEI-PEI;
%MAJOR
%JOB
%PUT &NAME MAJORED IN &MAJOR BUT HER CURRENT JOB IS NOT A &JOB;
```

Results in the SAS®log:

PEI-PEI MAJORED IN MAJOR BUT HER CURRENT JOB IS NOT A JOB.

1301  1301
WARNING 1301: APPARENT SYMBOLIC REFERENCE NOT RESOLVED.

Example 2: A macro variable is defined as a global macro variable before being assigned a value by a %LET statement (%GLOBAL statement is used):

Code:

```sas
%MACRO WHO_DOES;
"" ---- make macro variable global (a macro variable is defined);
%GLOBAL JOB;
"" ---- assign values to macro variables;
%IF "&NAME" EQ "PEI-PEI" %THEN %LET MAJOR = BIOLOGY;
%ELSE_IF "&NAME" EQ "WEI-WEI" %THEN %LET MAJOR = STATISTICS;
%ELSE_IF "&NAME" EQ "REI-CHI" %THEN %LET MAJOR = COMPUTER SCIENCES;
%ELSE_IF "&NAME" EQ "REI-CHANG" %THEN %LET MAJOR = NUTRITION;
%ELSE %IF "&NAME" EQ "REI-CHANG" %THEN %LET MAJOR = INFORMATION SCIENCES;
%IF "&MAJOR" EQ "BIOLOGY" %THEN %LET JOB = BIOLOGIST;
%ELSE_IF "&MAJOR" EQ "STATISTICS" %THEN %LET JOB = STATISTICIAN;
%ELSE_IF "&MAJOR" EQ "COMPUTER SCIENCES" %THEN
%LET JOB = COMPUTER PROGRAMMER;
%ELSE_IF "&MAJOR" EQ "NUTRITION" %THEN %LET JOB = DIETITIAN;
%ELSE_IF "&MAJOR" EQ "INFORMATION SCIENCES" %THEN
%LET JOB = LIBRARIAN;
%MEND WHO_DOES;

%LET NAME = REI-CHI;
%WHO DOES
%PUT &NAME IS A &JOB __ ;
```

Results in the SAS®log:

REI-CHI IS A COMPUTER PROGRAMMER.
Example 3: If the only purpose of a macro is to assign a value to a macro variable, put "&macrovariablename" as the last line of the macro:

Code:  
\%MACRO DEFINE:  
%* define MAJOR and assign to a value to this macro variable;  
%IF '&NAME' EQ 'PEI-PEI' %THEN %LET MAJOR = BIOLOGY;  
%ELSE %IF '&NAME' EQ 'WEI-WEI' %THEN %LET MAJOR = STATISTICS;  
%ELSE %IF '&NAME' EQ 'REI-CHI' %THEN %LET MAJOR = COMPUTER SCIENCES;  
%ELSE %IF '&NAME' EQ 'REILING' %THEN %LET MAJOR = NUTRITION;  
%ELSE %IF '&NAME' EQ 'REI-CHANG' %THEN %LET MAJOR = INFORMATION SCIENCES;  
%* --- semicolon is not necessary for the following statement;  
&MAJOR  
%MEND DEFINE;  

%LET NAME = WEI-WEI;  
%WHO_IS  
%PUT &NAME MAJORED IN &MAJOR BUT HER CURRENT JOB IS NOT A &JOB;  

Results in the SAS® log:  
WEI-WEI MAJORED IN STATISTICS.

Example 4: Macro WHO_IS defines two macro variables and both will be used outside the macro:

Code:  
\%MACRO WHO_IS;  
%* the method of "&macrovariablename" is not applicable here because two macro variables are defined in this macro;  
%GLOBAL MAJOR JOB;  
%IF '&NAME' EQ 'PEI-PEI' %THEN %LET MAJOR = BIOLOGY;  
%ELSE %IF '&NAME' EQ 'WEI-WEI' %THEN %LET MAJOR = STATISTICS;  
%ELSE %IF '&NAME' EQ 'REI-CHI' %THEN %LET MAJOR = COMPUTER SCIENCES;  
%ELSE %IF '&NAME' EQ 'REILING' %THEN %LET MAJOR = NUTRITION;  
%ELSE %IF '&NAME' EQ 'REI-CHANG' %THEN %LET MAJOR = INFORMATION SCIENCES;  
%IF '&MAJOR' EQ 'BIOLOGY' %THEN %LET JOB = BIOLOGIST;  
%ELSE %IF '&MAJOR' EQ 'STATISTICS' %THEN %LET JOB = STATISTICIAN;  
%ELSE %IF '&MAJOR' EQ 'COMPUTER SCIENCES' %THEN %LET JOB = COMPUTER PROGRAMMER;  
%ELSE %IF '&MAJOR' EQ 'NUTRITION' %THEN %LET JOB = DIETITIAN;  
%ELSE %IF '&MAJOR' EQ 'INFORMATION SCIENCES' %THEN %LET JOB = LIBRARIAN;  
%MEND WHO_IS;  

%LET NAME = PEI-PEI;  
%WHO_IS  
%PUT &NAME MAJORED IN &MAJOR BUT HER CURRENT JOB IS NOT A &JOB;  

Results in the SAS® log:  
PEI-PEI MAJORED IN BIOLOGY BUT HER CURRENT JOB IS NOT A BIOLOGIST.
Example 5: Two macros, MAJOR and JOB, are defined. Each macro creates a macro variable which will be used outside the macro:

Code:

```sas
%MACRO MAJOR;
%GLOBAL MAJOR;
%IF "NAME" EQ "PEI-PEI" %THEN %LET MAJOR = BIOLOGY;
%ELSE %IF "NAME" EQ "WEI-WEI" %THEN %LET MAJOR = STATISTICS;
%ELSE %IF "NAME" EQ "REI-CHI" %THEN %LET MAJOR = COMPUTER SCIENCES;
%ELSE %IF "NAME" EQ "REI-CHANG" %THEN %LET MAJOR = NUTRITION;
%ELSE %IF "NAME" EQ "REI-CHEN" %THEN %LET MAJOR = INFORMATION SCIENCES;
%MEND MAJOR;

%MACRO JOB;
%GLOBAL JOB;
%IF "MAJOR" EQ "BIOLOGY" %THEN %LET JOB = BIOLOGIST;
%ELSE %IF "MAJOR" EQ "STATISTICS" %THEN %LET JOB = STATISTICIAN;
%ELSE %IF "MAJOR" EQ "COMPUTER SCIENCES" %THEN %LET JOB = COMPUTER PROGRAMMER;
%ELSE %IF "MAJOR" EQ "NUTRITION" %THEN %LET JOB = DIETITIAN;
%ELSE %IF "MAJOR" EQ "INFORMATION SCIENCES" %THEN %LET JOB = LIBRARIAN;
%MEND JOB;

%LET NAME = REILING;
%MAJOR
%JOB
%PUT &NAME MAJORED IN &MAJOR BUT HER CURRENT POSITION IS NOT A &JOB.;
```

Results in the SAS®log:

REILING MAJORED IN NUTRITION BUT HER CURRENT POSITION IS NOT A DIETITIAN.

Example 6: Use "&JOB" in the second macro. "&MAJOR" statement is not applicable in the first one because the macro variable, MAJOR, is called in the second macro:

Code:

```sas
%MACRO DEFINE;
%GLOBAL MAJOR;
%IF "NAME" EQ "PEI-PEI" %THEN %LET MAJOR = BIOLOGY;
%ELSE %IF "NAME" EQ "WEI-WEI" %THEN %LET MAJOR = STATISTICS;
%ELSE %IF "NAME" EQ "REI-CHI" %THEN %LET MAJOR = COMPUTER SCIENCES;
%ELSE %IF "NAME" EQ "REI-CHANG" %THEN %LET MAJOR = NUTRITION;
%ELSE %IF "NAME" EQ "REI-CHEN" %THEN %LET MAJOR = INFORMATION SCIENCES;
%MEND DEFINE;

%MACRO POSITION;
%IF "MAJOR" EQ "BIOLOGY" %THEN %LET JOB = BIOLOGIST;
%ELSE %IF "MAJOR" EQ "STATISTICS" %THEN %LET JOB = STATISTICIAN;
%ELSE %IF "MAJOR" EQ "COMPUTER SCIENCES" %THEN %LET JOB = COMPUTER PROGRAMMER;
%ELSE %IF "MAJOR" EQ "NUTRITION" %THEN %LET JOB = DIETITIAN;
%ELSE %IF "MAJOR" EQ "INFORMATION SCIENCES" %THEN %LET JOB = LIBRARIAN;
%JOB
%MEND POSITION;

%LET NAME = REI-CHANG;
%MAJOR
%PUT &NAME IS STUDYING IN &MAJOR AND WILL BECOME A %POSITION.;
```

Results in the SAS®log:

REI-CHANG IS STUDYING IN INFORMATION SCIENCES AND WILL BECOME A LIBRARIAN.
Example 7: If there is no semicolon to signal the end of a "&macrovariablename" statement, the macro can be called anywhere in a %PUT statement:

Code:
```
%MACRO DEFINE;
  %GLOBAL MAJOR;
  %IF "&NAME" EQ "PEI-PEI" %THEN %LET MAJOR = BIOLOGY;
  %ELSE %IF "&NAME" EQ "WEI-WEI" %THEN %LET MAJOR = STATISTICS;
  %ELSE %IF "&NAME" EQ "REI-CHI" %THEN %LET MAJOR = COMPUTER SCIENCES;
  %ELSE %IF "&NAME" EQ "REILING" %THEN %LET MAJOR = NUTRITION;
  %ELSE %IF "&NAME" EQ "REI-CHANG" %THEN %LET MAJOR = INFORMATION SCIENCES;
%MEND DEFINE;

%MACRO POSITION;
  %IF "&MAJOR" EQ "BIOLOGY" %THEN %LET JOB = BIOLOGIST;
  %ELSE %IF "&MAJOR" EQ "STATISTICS" %THEN %LET JOB = STATISTICIAN;
  %ELSE %IF "&MAJOR" EQ "COMPUTER SCIENCES" %THEN %LET JOB = COMPUTER PROGRAMMER;
  %ELSE %IF "&MAJOR" EQ "NUTRITION" %THEN %LET JOB = DIETITIAN;
  %ELSE %IF "&MAJOR" EQ "INFORMATION SCIENCES" %THEN %LET JOB = LIBRARIAN;
%JOB
%MEND POSITION;

%LET NAME = REILING;
%MAJOR
%PUT &NAME IS NOT A %POSITION BECAUSE SHE QUIT &MAJOR 4 YRS AGO.;
```

Results in the SAS®log:
REILING IS NOT A DIETITIAN BECAUSE SHE QUIT NUTRITION 4 YRS AGO.

Example 8: The macro variable can be called only at the end of the %PUT statement if a semicolon is used to end the "&macrovariablename" statement:

Code:
```
%MACRO DEFINE;
  %GLOBAL MAJOR;
  %IF "&NAME" EQ "PEI-PEI" %THEN %LET MAJOR = BIOLOGY;
  %ELSE %IF "&NAME" EQ "WEI-WEI" %THEN %LET MAJOR = STATISTICS;
  %ELSE %IF "&NAME" EQ "REI-CHI" %THEN %LET MAJOR = COMPUTER SCIENCES;
  %ELSE %IF "&NAME" EQ "REILING" %THEN %LET MAJOR = NUTRITION;
  %ELSE %IF "&NAME" EQ "REI-CHANG" %THEN %LET MAJOR = INFORMATION SCIENCES;
%MEND DEFINE;

%MACRO POSITION;
  %IF "&MAJOR" EQ "BIOLOGY" %THEN %LET JOB = BIOLOGIST;
  %ELSE %IF "&MAJOR" EQ "STATISTICS" %THEN %LET JOB = STATISTICIAN;
  %ELSE %IF "&MAJOR" EQ "COMPUTER SCIENCES" %THEN %LET JOB = COMPUTER PROGRAMMER;
  %ELSE %IF "&MAJOR" EQ "NUTRITION" %THEN %LET JOB = DIETITIAN;
  %ELSE %IF "&MAJOR" EQ "INFORMATION SCIENCES" %THEN %LET JOB = LIBRARIAN;
%JOB
%MEND POSITION;

%LET NAME = REI-CHANG;
%MAJOR
%PUT &NAME IS STUDYING IN &MAJOR AND WILL BECOME A %POSITION.;
```

Results in the SAS®log:
REI-CHANG IS STUDYING IN INFORMATION SCIENCES AND WILL BECOME A %JOB

ERROR 180: STATEMENT IS NOT VALID OR IT IS USED OUT OF PROPER ORDER.
WARNING 1301: APPARENT SYMBOLOIC REFERENCE NOT RESOLVED.

1063
If SAS® codes are changed to:

```sas
%LET NAME = REI-CHANG;
%LET MAJOR
* • -- period is deleted from %PUT statement;
%PUT NAME IS STUDYING IN &MAJOR AND WILL BECOME A &POSITION
*/

Results in the SAS® log:
REI-CHANG IS STUDYING IN INFORMATION SCIENCES AND WILL BECOME A LIBRARIAN
```

Example 9: An example from SAS® autocal library, Code:

```sas
%macro datatyp(parm);
%*******************************************************************;
%* MACRO: DATATYP
%* USAGE: %datatyp(parm)
%* DESCRIPTION:
%* The DATATYP macro determines if the input parameter is
%* NUMERIC or CHARACTER data, and returns either CHAR or NUMERIC
%* depending on the value passed in through parm.
%* PROEDURE:
%* This macro simply checks to make sure that all characters
%* passed in are between 0 and 9 after leading and trailing
%* blanks have been removed. If so, NUMERIC is returned,
%* otherwise CHAR is returned.
%* ERRORS/RESTRICTIONS:
%* This macro requires the %VERIFY macro.
%*******************************************************************;
%local type;
%let type=CHAR;
%let parm=%qleft(%qtrim(&parm));
%if %length(&parm)=0 | %length(&parm)10
  %then %goto endata;
%if %verify(&parm,'0123456789')=0
  %then %let type=NUMERIC;
%endata: &type
%mend datatyp;
```

REFERENCES
