Accessing a SAS® Variable Label: An Example of the Use of Macro Variables

Carl E. Pierchala, Washington, DC

ABSTRACT

In this paper, I give a short overview on the use of SAS macro variables. The latter can be a valuable tool for many SAS programmers. The paper introduces macro variables as a way to ease making routine changes to SAS programs. Then, I illustrate how to use the SYMPUT function in conjunction with PROC TRANSPOSE in order to put a SAS data set internal variable label into a macro variable, thus making the label available to be put into a TITLE statement. To illustrate the method, some random number functions are used to generate a test data set. A restriction on the use of the random number functions is coincidentally given. Finally, I close with a tip on macro variable syntax.

INTRODUCTION

Three methods for extracting the internal SAS variable label were described in the November (Fourth Quarter) 1994 issue of the DC SAS Users Group Newsletter. These methods were attributed to Phil Mason via SAS-L. Two of his methods use PROC SQL, while the third uses CALL LABEL in a DATA _NULL_ step. This paper gives another simple method using Base SAS software. Perhaps more importantly, it illustrates the value of SAS macro variables.

USING MACRO VARIABLES

Suppose you have a large data base stored as a SAS data set named MONSTER in a library with libref MUCHDATA. MONSTER consists of many observations and a large number of variables. You need to produce a histogram (bar chart) for variable A, so you write the following program:

```
TITLE "HISTOGRAM OF VARIABLE A" " IN MUCHDATA.MONSTER";
PROC CHART DATA=MUCHDATA.MONSTER (KEEP=A);
VBAR A;
RUN;
```

Now suppose you decide to run the program for another variable, B. You need to edit your program and change each appropriate instance of the string A to the string B. To later process another variable, C, you need to change each B to C. And so on as it becomes necessary to process other variables. A macro variable can ease making the necessary changes. Simply write the program as follows:

```
%LET CVAR=A;
TITLE "HISTOGRAM OF VARIABLE &CVAR" " IN MUCHDATA.MONSTER";
PROC CHART DATA=MUCHDATA.MONSTER (KEEP=&CVAR);
VBAR &CVAR;
RUN;
```

The %LET statement assigns the string A to the macro variable CVAR. Before submitting the program to SAS, the macro processor system substitutes the string A in every place where &CVAR occurs in the program. If you want to run the program for variable D after running it for variable A, just change the first statement to

```
%LET CVAR=D;
```

The above is a very short introduction into the use of macro variables. Read the chapter on macro variables in the SAS macro manual for more details. Special rules apply in various circumstances. For example, the double quotes in the TITLE statement above are needed; had single quotes been used, &CVAR would not have been replaced by the string A in the title.

ACCESSING A VARIABLE LABEL

Now suppose you know that the owner of MUCHDATA.MONSTER went to the trouble to give good descriptive names for each of the SAS variables, and stored these internally as SAS variable labels. It would be nice if the title of the histogram were to display these descriptive labels. How do you make the label available to the program?

One method in addition to those referred to above for extracting an internal variable label works as follows. First PROC TRANSPOSE is used to put the variable label into the variable _LABEL_ in an output data set (named VARLABL below), and then the SYMPUT function is used in a DATA _NULL_ step to store the value as a macro variable (named CVARLAB below). The code is as follows, and needs to be placed in the program after the macro variable CVAR is assigned.

```
PROC TRANSPOSE DATA=MUCHDATA.MONSTER (OBS=1 KEEP=&CVAR) OUT=VARLABL;
RUN;
DATA _NULL_; SET VARLABL; CALL SYMPUT(’CVARLAB’,TRIM(_LABEL_)); RUN;
```

For details about PROC TRANSPOSE, see the SAS Procedures guide. Since we are restricting ourselves to the first observation and only one variable in the data set, computational speed is fast. The transposed data set, VARLABL, has the variable _LABEL_ added to it, containing &CVAR’s label in the first (and only) observation. For details about CALL SYMPUT, see the SAS Guide to Macro Processing. We are using CALL SYMPUT to put the label into a macro variable, thus making it available in later PROC or DATA steps. Note that the TRIM function removes leading blanks that may exist in the internal label for &CVAR in MUCHDATA.MONSTER.

AN EXAMPLE

We first use some SAS random number functions to generate 100 observations with four variables:
%LET SEED=1447291;
DATA TEST;
  DO I=1 TO 100;
    U=UNIFORM(&SEED);
    LABEL U=
      'UNIFORMLY DISTRIBUTED OVER (0,1)';
    N=RANNOR(&SEED)*0.125+.5;
    LABEL N=
      'NORMALLY DISTRIBUTED: MEAN .5, S.D .125';
    T=RANTRI(&SEED,.5);
    LABEL T=
      'TRIANGULARLY DISTRIBUTED: PEAK .5';
    E=RANEXP(&SEED)*.2;
    LABEL E=
      'EXPONENTIALLY DISTRIBUTED: LAMBDA .2';
    OUTPUT;
  END;
RUN;

Note the use of the macro variable SEED for each of the four random number functions employed in the code above. A different choice in the %LET statement would give a different set of data. Rerunning the code without changing the %LET statement produces the same data.

(An important caveat is to be noted. While the programming technique above works for the purpose of generating test data, it is not adequate for performing valid simulations in SAS 6.10 and earlier due to a bug. This is documented via the world wide web in usage note A514 at the SAS web site http://www.sas.com/service/techsup/find_answer.html. Under SAS 6.10 some of the variables in this example are highly correlated. Between T and U the correlation is 0.989, between T and E it is -0.915, and between U and E it is -0.996. In each of these cases, p<.0001 was reported by PROC CORR. However, under SAS 6.12, this difficulty did not occur, as the correlations were small and none gave p<.05. Still, even under SAS 6.12, I recommended that careful testing of the random numbers be done to verify that simulations valid for a user’s purposes are being produced.)

Now suppose we wish to see a histogram of variable U, with its label in the title. The following code produces the histogram.

%LET VCHART=U;
PROC TRANSPOSE DATA=TEST
  (OBS=1 KEEP=&VCHART) OUT=RD;
RUN;
DATA _NULL_; Set RD;
  CALL SYMPUT('LCHART',TRIM(_LABEL_));
RUN;

To produce the labeled histogram for variable G, for example, we merely substitute G for U in the %LET statement in the preceding code.

A FINAL NOTE

As a novice user of macro variables, I often was confused as to when to use the & symbol with a macro variable. The answer is essentially as follows. When you want the macro variable’s current string value to be substituted into the code before it is submitted to SAS, use the & symbol directly before the macro variable name. But when you want a string value to be placed into the macro variable for use in a later PROC or DATA step, use an appropriate statement without the & symbol before the macro variable name.

CONCLUSIONS

Macro variables can be very useful to the SAS programmer, as illustrated in this paper.

REFERENCES


ACKNOWLEDGMENT

I thank Michael P. Cohen for reviewing and commenting on two drafts of this paper, and to SAS Institute’s Technical Support Division for information about the bug mentioned in this paper.

SAS is a registered trademark or trademark of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Carl Pierchala lives and works in Washington, DC. He can be contacted at 2400-16th St., NW #537, Washington, DC 20009-6629, or by phone at (202) 483-8131.