PharmaSUG China 2017 - Paper 5

SYMply PUT: GET the most out of SYMPUTX and SYMGETN

Robert Howard, Veridical Solutions, San Diego, CA

ABSTRACT

SYMPUTX and SYMGETN are simple, yet extremely powerful and useful functions. The CALL SYMPUTX routine allows you to quickly and easily store dataset values into macro variables, while the SYMGETN function retrieves these values. Using these functions saves time and reduces the chance of data calculation errors in even the most complex programs. By working through a practical example, you will see the utility and flexibility of these indispensable functions.

INTRODUCTION

Though easily implemented, CALL SYMPUTX and SYMGETN are relatively unknown or under-used by many beginning-to-intermediate-level SAS® software programmers. These versatile tools can simplify program structures, reduce errors and allow for increased efficiency and economy in SAS coding. In this paper, we will explore the CALL SYMPUTX routine and the SYMGETN function as they are used in SAS Version 9. We will begin with simple examples that will introduce the syntax of each function. Later we will work through a more sophisticated program to illustrate the utility and efficiency of these powerful tools.

CALL SYMPUTX ROUTINE

The CALL SYMPUTX routine is a quick and simple way to store or assign values into macro variables. Once stored, these values can be accessed globally throughout a program and can be displayed in generated reports. The CALL SYMPUTX routine is used in a DATA STEP with the following syntax:

```
call symputx(macro-variable,value);
```

The ***macro-variable*** argument can be:

a) A character string in quotation marks
b) A character variable name

Values of the ***macro-variable*** argument must follow existing naming rules for macro variables.

The ***value*** argument can be:

a) A character or numeric value
b) A character or numeric variable name

Note: When using the CALL SYMPUTX routine, all leading and trailing spaces are truncated from both the ***macro-variable*** and the ***value***. The CALL SYMPUTX routine can be used repeatedly, at any point in the DATA step.

EXAMPLE 1: CALL SYMPUTX

In this example we will create a macro variable named ***&thisyear*** which will store the numeric value of **2017** and the macro variable named ***&trt0*** which will store the character value of **"Placebo".**

```
data _null_;
  call symputx('thisyear',2017);
  call symputx('trt0','Placebo');
run;
```

A %PUT statement will show the value of ***&thisyear*** and ***&trt0***.

```
%put &thisyear &trt0;
```

The log result displays: 2017 Placebo

EXAMPLE 2: CALL SYMPUTX

Now let's examine a more sophisticated usage of the CALL SYMPUTX routine. We will use a character variable as the ***macro-variable*** argument. Consider the following dataset named DSET1:

DSET1

	name	age
1	Savannah	13
2	Jillian	13
3	Ayden	12
4	Quinn	11
5	Evan	7

NAME is a character variable and the variable AGE is numeric.

Using the CALL SYMPUTX routine, we can use the variable NAME as the argument for the ***macro-variable***. The actual values of NAME will become the macro variable name. If we use AGE as the argument for the ***value***, then the values of AGE will be stored in corresponding values of NAME. Looking at this closer:

```
data _null_;
  set dset1;
  call symputx(name,age);
run;
```

We now have the macro variables for each value of NAME: *&savannah, &jillian, &ayden, &quinn* and *&evan* which contain the corresponding values of AGE.

A %PUT statement will return the value:

```
%put &savannah &jillian &ayden &quinn &evan;
```

The log result displays: `13 13 12 11 7`

SYMGET AND SYMGETN FUNCTIONS

The SYMGET and SYMGETN functions easily retrieve the values of previously stored macro variables and assigns these values in new programmer-defined variables. The choice of SYMGET or SYMGETN is dictated by the data type of the original macro variable value; SYMGET is used to store character values, while numeric values require SYMGETN.

With SYMGET, the programmer-defined variable will be a *character* string; retrieving a *character* value from the given **macro-variable**:

```
<variable> = symget('macro-variable');
```

With SYMGETN, the programmer-defined variable will be *numeric*; retrieving a *numeric* value from the given **macro-variable**:

```
<variable> = symgetn('macro-variable');
```

As with the CALL SYMPUT routine, the ***macro-variable*** for both SYMGET and SYMGETN can either be:

a) The name of a macro variable with no ampersand (within the single quotation marks)
b) The name of a variable with values that have been assigned as macro variables. (Note that, with this usage, the ***macro-variable*** will *not* be surrounded with single quotes.)

EXAMPLE 3: SYMGETN

First, let's recall our example of the CALL SYMPUT routine where we assigned the macro variable ***&thisyear*** with the numeric value **2017**. We now want to create a new variable called **YEAR** that will be assigned the value of ***&thisyear***. Executing the following code will produce the dataset, DSET2, as displayed below:

```
data dset2;
  year=symgetn('thisyear');
run;
```

DSET2

	year
1	2017

YEAR is a numeric variable with value 2017.

EXAMPLE 4: SYMGETN

Now let's look at another example – one that uses a variable and its corresponding values. Previously, in Example 2, the CALL SYMPUTX routine was used to create macro variables using the following values for the variable NAME: *&savannah, &jillian, &ayden, &quinn* and *&evan*. The corresponding numeric AGE value has been stored in each of the macro variables. Now, consider the following dataset, labeled DSET3:

DSET3

	name
1	Savannah
2	Jillian
3	Ayden
4	Quinn
5	Evan

We can use the SYMGETN function to create a numeric variable AGE.

```
data dset4;
  set dset3;
  age=symgetn(name);
run;
```

The result is a new dataset, DSET4 (displayed to the right), with the newly created numeric variable, AGE, which contains the corresponding values associated with each instance of NAME.

DSET4

	name	age
1	Savannah	13
2	Jillian	13
3	Ayden	12
4	Quinn	11
5	Evan	7

Clearly, this is a simple exercise to illustrate the basic usage of this function. We can now look at a more practical example.

PUT IT ALL TOGETHER: A PRACTICAL EXAMPLE COMBINING CALL SYMPUT AND SYMGETN

Consider the following dataset, DSET5, with 11 observations:

DSET5

	USUBJID	TREAT	GENDER
1	1	A	Female
2	4	A	Male
3	5	A	Female
4	6	A	Female
5	8	A	Male
6	11	A	Female
7	12	A	Male
8	14	A	Male
9	2	B	Female
10	3	B	Female
11	9	B	Male

TREAT is a character variable with values of either "A" or "B". GENDER is also character variable with values of either "Male" or "Female". USUBJID is a numeric unique subject identifier variable.

Suppose we want to calculate the number and percentage of subjects by gender for each treatment group. First we count the number of subjects in each treatment group; this is accomplished through use of the PROC FREQ function. This result will serve as our denominator:

```
proc freq data=dset5 noprint;
  tables treat / out=trtfreq;
run;
```

The dataset TRTFREQ is created:

There are 12 subjects assigned to Treatment Group A and 3 subjects assigned to Treatment Group B.

TRTFREQ

	TREAT	COUNT
1	A	8
2	B	3

Using the CALL SYMPUT routine, we assign macro variables for the value of TREAT.

```
data _null_;
  set trtfreq;
  call symputx(treat,count);
run;
```

We now have the macro variable ***&A***, which contains the value of **8**, and the macro variable ***&B***, which contains the value **3**. Both **&A** and **&B** are numeric because COUNT is numeric. We now have our denominators for both treatment groups stored in macro variables.

To obtain the number of “Male” and “Female” subjects by Treatment Group, we will again use PROC FREQ. The resulting dataset FREQ0 is shown on the left:

```
proc freq data=dset5 noprint;
  tables treat*gender / out=freq0;
run;
```

FREQ0

	TREAT	GENDER	COUNT
1	A	Female	4
2	A	Male	4
3	B	Female	2
4	B	Male	1

To calculate the percentages of male and female subjects in each treatment group, we divide COUNT by the number of subjects in each treatment group. We use the SYMGETN function to retrieve the total number of subjects (the denominator) in each Treatment Group in a new DATA step:

```
<new variable>=symgetn(treat);
```

Alternatively, we can use the SYMGETN function within the calculation to obtain the percentages:

```
data final;
  set freq0;
  percent=put(count/symgetn(treat),percent7.1);
run;
```

The dataset FINAL is created with the new variable PERCENT. The SYMGETN function was used to retrieve the denominator for the corresponding value of TREAT to calculate the percentages:

FINAL

	TREAT	GENDER	COUNT	PERCENT
1	A	Female	4	50.0%
2	A	Male	4	50.0%
3	B	Female	2	66.7%
4	B	Male	1	33.3%

CONCLUSION

The CALL SYMPUTX routine and the SYMGETN and SYMGET functions are simple, yet powerful tools, readily available within SAS Version 9. When used in concert, CALL SYMPUTX and SYMGETN prove to be an easy, efficient, and effective method to store and retrieve macro variables, saving the programmer time, simplifying code, and reducing data calculation errors. With these benefits, the use of the CALL SMPUTX routine and the SYMGETN and SYMGET functions are sure to become of standard component of your programming practices.

CONTACT INFORMATION

Thank you for your time and interest. If you have any comments or questions please feel free to contact me:

Rob Howard
Veridical Solutions
P.O. Box 656
Del Mar, CA 92014
858.205.8284
E-mail: rob.howard@veridicalsolutions.com
Web: www.veridicalsolutions.com