IN08_05

Fun With Functions

Marje Fecht, Prowerk Consulting Ltd, Mississauga, ON Canada

ABSTRACT

Functions can be fun (and useful) if you know when, how, and why to use them. Functions can be frustrating if you do not know the tricks behind them.

In this presentation, we will explore the most commonly used functions to streamline data processing and reduce your programming effort. You will learn how to use summary functions, string functions, conversion functions, and date functions. We will also explore some of the 60 functions added to SAS 9.

This presentation will also help you understand WHERE to use functions, since they are not "just for the DATA Step".

INTRODUCTION

The SAS® language is rich with functions that enable the savvy programmer to handle all of their data transformation and manipulation needs. However, the wealth of functions does have implications:

- How do you know **which function** to choose when so many are similar?
- Are there **tricks** for using the functions?
- Should you consider replacing "old code" with some of the new **SAS 9** functions?
- **Where** can functions be used in your SAS code?
- What is the difference between using **Summary** functions in the data step versus using **Summary** Procedures?

This Proceedings paper provides an overview of the topics that will be covered in the presentation. The detailed PowerPoint is available at the presentation, or from the author following the conference.

A QUIZ

Before diving into the topics, test your knowledge!

1. What is the value of Q1?

```
Data _null_;
   length C $8;
   C = '' ;  *** No blanks at all;
   Q1 = length( C );
run;
```

2. What is the value of Q2?

```
Data _null_;
   length C $8;
   C = ' ' ;  *** ONE blank;
   Q2 = length( C );
run;
```

3. What is the value of Q3?

```
Data _null_;
   length C $8;
   C = 'eight ' ;
   Q3 = length( C );
run;
```

4. What is the value of Q4?

```
Data _null_;
   m = . ;  ** missing;
   y = 4 ;
   z = 0 ;  ** zero;
   Q4 = m + y + z;
run;
```

5. What is the value of Q5?

```
Data _null_;
   m = . ;  ** missing;
   y = 4 ;
   z = 0 ;  ** zero;
   Q5 = mean(m -- z );
run;
```

How sure are you of your answers? Have you run into these questions before?

Less than 50% of the audience during my first two presentations (consisting of novice, intermediate, and advanced SAS users) got the correct answers.

The answers are:

1. Q1 = 1
2. Q2 = 1 (Length always returns 1 for a blank string)
3. Q3 = 5 (Length returns the position of the right most non-blank character)
4. Q4 = . (numeric missing value)
5. Q5 = . (numeric missing value)

SUMMARY FUNCTIONS

Key features of most SAS Summary functions include:

- "ignore" arguments containing missing data
- process data within an observation (row) of data
- use the OF keyword to specify ranges of variables.

If your goal is to summarize data within a column (variable), you should consider SAS Summary Procedures such as PROC MEANS or PROC SUMMARY.

EXAMPLES

The **SUM** function provides the total of all non-missing arguments.

```
data _null_;
   m = . ;      y = 4 ;      z = 0 ;
   Q1 = sum(m , y , z );
   Q2 = sum(of m -- z);
run;
```

The above program results in Q1 = 4 and Q2 = 4. Since the variable **m** contains a missing value, it is ignored in the computation. To sum an entire list of variables (where the variable order is dictated by order within the SAS data set), use the keyword OF and a double hyphen. If you exclude the keyword OF, SAS will subtract the value "negative **z**" from the value of **m** and return the result.

To determine the number of missing values that are excluded in a computation, use the **NMISS** function.

```
data _null_;
   m = . ;        y = 4 ;       z = 0 ;
   N = N(m , y,  z);
   NMISS = NMISS(m , y,  z);
run;
```

The above program results in N = 2 (Number of non missing values) and NMISS = 1 (number of missing values).

TRICKS

Listing all of the variables in a summary function can be time consuming. If you use some tricks in naming your variables, you can use shortcuts. For example, if you want to total all of the variables beginning with the string **amt**, simply specify **OF AMT:** as the argument for the summary function.

```
data _null_;
   AmtGroc = 220 ;  AmtCar = 180 ;
   AmtUtil = 270;       Month = 12;
   MthlyAmt = Sum( of Amt:);
   put MthlyAmt = ;
run;
```

The above program results in MthlyAmt = 670.

STRING AND SEARCH FUNCTIONS

The SAS Language has a full complement of character manipulation functions, including functions to

- Search
- Substring
- Join
- Compress
- and many others…..

COMBINING CHARACTER STRINGS

If you work with character data, you probably have to combine strings or extract strings. The coding can be cumbersome and there are many misunderstandings about how the functions work.

EXAMPLE

A standard example of string manipulation involves joining a first name, middle initial, and last name. Since all SAS variables are of fixed length, this results in extra blanks in the result by default. Thus, a standard solution is:

```
data cat;
   length fn  mi  ln  $10;
   fn = 'Mary';   mi = '';    ln = 'Smith';
   Name = trim(fn)  ||  ' '  ||  trim(mi)  ||  ' '  ||  ln;
   put name=;
run;
```

What would happen without the **trim** function? Since all SAS variables are of fixed length, Mary is actually stored in fn as Mary followed by 6 blanks. The result for **name** is **Mary Smith** (there are two blanks between Mary and Smith).

Unfortunately, if a person had no recorded middle initial, the result would be 2 blank spaces between first and last names. When in doubt, use TRIM**N** to remove trailing blanks from character expressions and **return a null string** if the expression is missing. The above example would be coded better as:

```
data cat;
   length fn  mi  ln  $10;
   fn = 'Mary';   mi = '';       ln = 'Smith';
   Name = trimN(fn) || trimN(' ' || mi ) || ' ' || ln;
   put name=;
run;
```

SAS 9 FUNCTIONS

Prior to SAS 9, you probably used complex sequences of functions or steps to accomplish standard tasks. When you concatenated 2 character strings, you likely had to left justify, trim, and include an extra blank to get the proper result. If you needed to determine the *second highest value* in a list of variables for each observation, you likely used quite a few comparative statements or maybe even used a PROC.

More than 60 analytic and data manipulation functions were added to SAS 9. Many functions were added to reduce code complexity. Additionally, the logic behind some functions was revised to improve performance (LENGTH and TRIM functions).

REDUCE CODE COMPLEXITY

Several character functions were added to reduce the complexity of common tasks. For example, the CATX function provides a straightforward solution to joining 2 strings so the result removes leading and trailing blanks and includes a designated character as a separator.

Version 8 Solution:

```
Combine = trim(left(Char1))  || ' ' || left(Char2);
```

SAS 9 Solution:

```
Combine = CATX(' ', Char1, Char2);
```

If you used repetitive INDEX and SUBSTR functions to determine how many times a "word" occurs in a character string, that task can be accomplished more easily in Version 9. The COUNT function determines the number of times *argument 2* occurs in *argument 1*.

SUBSTR has always been an effective function for extracting (or replacing) a portion of a character string. Two variations of SUBSTR are available in SAS 9:

CHAR - to extract a single character from a text string

```
JustOne = CHAR(String , 5);
```

FIRST – to grab just the first character from a text string

```
Initial = FIRST(name);
```

If you need to strip both leading and trailing blanks from a string, you can skip using the LEFT and TRIM functions and just use the new STRIP function.

COMPUTING STATISTICS OR QUANTITIES

If you work frequently with Medians or Percentiles, you can now obtain those quantities in the DATA step. The MEDIAN function determines the median of non-missing values. The PCTL function accepts a percentage for *argument 1* to specify the percentile of interest. The function then computes the percentile for the list of values.

If you need to find the *second largest* value in a list of values, or perhaps the *third smallest* value, the LARGEST and SMALLEST functions are available. In addition to supplying a list of values, you supply *which* placement you are looking for.

```
ThirdLargest = LARGEST(3 , a, b, c, d, e, f);
```

WHERE TO USE FUNCTIONS

Do you typically create a subset of your data before processing the data in a PROC? If you answered YES, you may be "reading" your data twice unnecessarily. Most functions can be used in a WHERE statement which means a lot of processing can occur in PROCs.

Suppose you want to look at all July sales. Instead of a two-step "subset – print" process, you can handle the subsetting in the same step as the print.

```
proc print data=sashelp.retail;
  where month(date) = 7;
  title 'Sandal Sales in July';
run;
```

If the above example were handled as a DATA step to subset, followed by a PROC step to Print, the data would have been processed twice. If you work with large volumes of data, the savings in resources would be considerable.

CONCLUSION

Before you write "messy code" to solve your data transformation/manipulation needs, review the existing functions. You may be able to solve most tasks in one line rather than many, which will also make your code easier to understand (and thus easier to "inherit").

REFERENCES

The SAS online documentation provides both syntax and examples for all of the available functions.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Marje Fecht
Prowerk Consulting

Email: marje.fecht@prowerk.com
Web: www.prowerk.com