

The Functionality of Arithmetic SAS® Functions for the CRO User

Phil d'Almada and Karen Graham, Rho, Inc.

ABSTRACT

The Contract Research Organisation (CRO) industry requires precision and accuracy in the handling of data. One tool to address this concern is the use of SAS functions. Arithmetic SAS functions are categorized, with other functions, by SAS into two categories entitled Truncation and Descriptive Statistics. The authors will discuss these types of functions in light of their application in the CRO industry and within the scope of the authors' experiences. This paper will illustrate the power of SAS functions as an option to arriving at the same result as some other SAS products, namely, SAS PROCEDURES or PROCs. Advantages from the use of SAS functions are a cleaner SAS log and a reduction in debugging time.

INTRODUCTION

In contrast to the realm of research, the business of the Contract Research Organization (CRO) industry requires precision and accuracy applied in a different manner as far as the management of databases. The SAS System is the prevalent tool of choice in this industry and allows facilitation of this concern for precision and accuracy through the power of SAS functions. A brief review of some of the functions that the user would apply in the CRO industry is the scope of this work. In particular, the authors will focus on arithmetic-type functions.

The schematic for this paper is to present discussion on groups of functions where a group is defined by the similarity in syntax and in the result returned from implementation of those functions within the group. The discussion begins with a working illustration or a reason from the CRO industry for the use of a group of functions along with definitions of the functions within the group. One or more examples will close the discussion on the particular group of functions.

TYPES OF FUNCTIONS

In the online users guide, SAS categorizes the arithmetic functions described herein as 'Truncation' functions or functions used to derive 'Descriptive Statistics'. Two lists of functions discussed in this paper are given below.

Truncation functions include

INT
ROUND
FLOOR
CEIL

Functions used for Descriptive Statistics include

SUM
MAX
MIN
MEAN
RANGE
STD
STDERR
VAR
N
NMISS
MISSING

TRUNCATION FUNCTIONS

In the CRO industry, data may come in to the CRO in various nonstandard formats depending on the data collection and keying sources. Since the industry demands some level of standardization, a set of data specifications are defined, sometimes by the client, and this results in the CRO programmer becoming relegated to abide by these specifications. Consider, for example, continuous data that are characterized by at least four decimal places and the specification calls for truncation, that is, reduction to a fewer number of decimal places. This could include 'rounding up,' or, reducing to the nearest higher integer; or 'rounding down,' or, reducing to the nearest lower integer.

INT, CEIL, FLOOR

The first group of functions to be discussed is those that return an integer value while ignoring the decimal portion of the data value. The syntax is given as

FUNCTION (argument)

where *argument* must be numeric and could be either a variable or a constant.

This syntax may be used as an expression in an evaluation as

IF FUNCTION (argument) ... THEN ...;

or in the derivation of a new variable such as

NewVar = FUNCTION (argument);

Consider, for example, a variable such as age at a visit to health care provider and is encoded in decimal form and from which integer values are to be derived. Typically, the INT function would be appropriate since it would return only the integer portion of the original data value. Since the decimal portion of *argument* is simply dropped, the rounding effect of the INT function is to round down a positive data value or round up a negative data value. Thus, from positive values such as 2.1 or 2.6, INT would return the value 2, and from negative values such as -2.1 or -2.6, INT would return the value -2. To illustrate,

$\text{INT}(2.1) = \text{INT}(2.6) = 2$

$\text{INT}(-2.1) = \text{INT}(-2.6) = -2$

The INT operation on negative values is identical to the CEIL function on negative values. They return the nearest integer that is greater than an original negative value. Thus, the CEIL function rounds up a data value:

$\text{INT}(-2.1) = \text{CEIL}(-2.1) = -2$

On original positive data values, CEIL returns the next higher positive integer. Thus, from 2.4 or 2.7, CEIL returns 3, that is,

$\text{CEIL}(2.4) = \text{CEIL}(2.7) = 3$

On the other hand, the INT operation on positive values is identical to the FLOOR function on positive values. They return the nearest integer that is less than an original positive value. Thus, the FLOOR function rounds down a data value:

$\text{INT}(2.6) = \text{FLOOR}(2.6) = 2$

On original negative data values, FLOOR returns the next lower negative integer. Thus, from -2.4 or -2.7, FLOOR returns -3, that is,

$\text{FLOOR}(-2.4) = \text{FLOOR}(-2.7) = -3$

For all three of these functions, the functions return the same original integers when operating on original data values that are integers. Thus,

$\text{INT}(-3) = \text{CEIL}(-3) = \text{FLOOR}(-3) = -3$

and

$\text{INT}(3) = \text{CEIL}(3) = \text{FLOOR}(3) = 3$

ROUND

The fourth truncation function to be discussed, ROUND, operates in the traditional or expected manner of mathematical rounding. That is, when applied to an original data value, ROUND will return the nearest integer after appropriately rounding up or down. This shows the uniqueness of the ROUND function where a rounding-off level or unit can be specified and this value will be positive. The syntax is given as

ROUND (argument, round-off-unit)

and may also be used as an expression in an evaluation as

IF ROUND (argument, round-off-unit) ... THEN ...;

or in the derivation of a new variable such as

NewVar = ROUND (argument, round-off-unit);

Thus, from a value in the range [2.0 to 2.5), ROUND will return the value 2, and from a value in the range [2.5 to 3.0], ROUND will return the value 3. From a value in the range [-3.0 to -2.5], ROUND will return the value -3, and from a

value in the range (-2.5 to -2.0], ROUND will return the value -2. These are examples given with data values of one decimal place with rounding to the nearest integer or unit.

The default round-off unit is unity, thus,

ROUND (64.4) and ROUND(64.4,1) will return the value 64.

The round-off unit may be hundredths, specified as .01; or hundreds, 100; or triples to one decimal place, .3; and so on. Respective examples of each of these three levels mentioned above are

ROUND (16.367,.01) returning 16.37

ROUND (1183,100) returning 1200

ROUND (25.22,.3) returning 25.2[♦]

With selection of a round-off unit of unity, the user could emulate the return from the CEIL, INT, or FLOOR functions.

FUNCTIONS FOR DESCRIPTIVE STATISTICS

To fit the requirements for summary reports, some simple statistics have to be produced and displayed on the reports. One way to do this using the SAS System is to implement one of a variety of PROCs or PROCEDURES such as MEANS, SUMMARY, or FREQ. These procedures will summarize response data that are arranged in columns across rows. Invariably, however, further manipulation such as transposition is to be expected before producing the final report. In another approach where the response data are arranged in rows across columns, the user could implement SAS functions in a DATA step. The functions to be described in this section are classified by SAS as those for Descriptive Statistics. The syntax for these functions can generally be represented in two ways as

FUNCTION (argument1 <,..., argumentn>)

FUNCTION (OF variablelist)

and may also be used as an expression in a conditional statement as

IF *FUNCTION (OF variablelist)* ... THEN DO;

...

END;

or in the derivation of a new variable such as

NewVar = *FUNCTION (OF variablelist)*;

where *argument* is a variable, constant or expression and *variablelist* is a list of variables separated by commas or spaces, or if the list is sequential with numeric suffices, then the list may be specified using the first and last elements of the selection of variables and separated by a hyphen. Using the latter style, it is possible to specify more than one list such as *FUNCTION (OF W5-W10 W20-W25)*, where *Wnumber* is a sequential list of variables each with a numeric suffix, *number*.

For all the functions described in this section, the arguments or expressions must be numeric except for the MISSING function where either numeric or character arguments or expressions could be evaluated.

MEAN, VAR, STD, STDERR

Consider, for example, a series of variables that denote weight of a subject at a number of time intervals, WGT1-WGT12, and simple statistics such as the mean, variance, standard deviation and standard error of the mean, need to be determined from these variables and included in a display. The respective functions would be implemented in the assignment of derived variables as

MWGT = MEAN (OF WGT1-WGT12);

VWGT = VAR (OF WGT1-WGT12);

SDWGT = STD (OF WGT1-WGT12);

SEWGT = STDERR (OF WGT1-WGT12);

and the derived variables may be presented in the report in sequence or the data set may be transposed and the new variables presented in column fashion.

SUM, MAX, MIN, RANGE

This group of functions will accomplish variable reduction when applied to a series of variables. One scenario involving the maximum or minimum value in a series could occur when producing a report using PUT statements and conditional processing that requires the monitoring of the number of lines written to an output page. More specifically, consider a scenario of three variables, each to be written in column fashion and each using a variable number of lines. Those numbers are held in the variables LINES1, LINES2, LINES3. In this scenario, a maximum or minimum number of lines could be assessed for conditional execution as

IF MAX (LINES1, LINES2, LINES3) >= *constant* THEN PUT _PAGE_;

or

IF MIN (LINES1, LINES2, LINES3) <= *constant* THEN PUT 'Insufficient data.';

[♦] Note: SAS Online Doc indicates that this value would resolve to 25.3, the nearest third.

where *constant* is some predetermined fixed value.

Clearly, these two functions, MAX and MIN, can return the upper and lower bounds of a series of variables or columns in a data set for a given observation or row. Although implementing the pair gives a sense of the range of value for each observation, the RANGE function would return the actual size of the range that is the difference between the MIN and MAX values. Thus, given a series of variables with values V1=2, V2=4, V3=6 then

MAX (OF V1-V3) returns 6

MIN (OF V1-V3) returns 2

and for

UVAL = MAX (OF V1-V3)

LVAL = MIN (OF V1-V3)

the size of the range is UVAL – LVAL = 4 = RANGE (OF V1 – V3).

The SUM function is oftentimes used to add data values derived from a series of variables or a series of constants. The user may forego summation programming since a missing value in at least one source variable will cause the summation statement to return a missing value whereas the SUM function will ignore any missing value. This can be stated as overcoming missingness. To summarize, when using the SUM function, missing values are ignored whereas when using summation programming, the presence of such input missing values will cause the return of an output missing value. Thus, given V1 = 2, V2 = 4, V3 = 6, the following statements return the same value, 12:

FSUM = SUM (OF V1-V3);

ASUM = V1 + V2 + V3;

However, given V1 = 2, V2 = ., V3 = 6, the SUM function returns the value 8, while the summation statement returns a missing value:

FSUM = SUM (OF V1-V3) = 8

ASUM = V1 + V2 + V3 = .

N, NMISS, MISSING

A final group of functions to be discussed is one where conditional processing may be controlled. Missingness or nonmissingness of data values provides the key in this group for conditional processing where, given a series of variables, the NMISS function returns the number of missing values in the series, and the N function returns the number of nonmissing values in the series. One example is to prevent the note being printed in the SAS log on output missing values being generated from input missing values. Thus, given two date variables, DATE1 and DATE2, and the likelihood exists that one or other of the two variables could be missing, and the need is to execute a DO loop, the following could be implemented:

IF NMISS (DATE1, DATE2) = 0 THEN DO;

...

END;

The DO loop will only execute if both dates are not missing.

Conversely, the assessment could be made on the basis of nonmissingness of date values using the statements

IF N (DATE1, DATE2) = 2 THEN DO;

...

END;

Again, the DO loop will execute when both dates are not missing.

In the event a scenario exists where a DO loop must execute when at least one date value always exists, these statements are adequate:

IF N (DATE1, DATE2) > 0 THEN DO;

...

END;

or

IF NMISS (DATE1, DATE2) < 2 THEN DO;

...

END;

Finally, an assessment on a single variable could be made for conditional processing using the MISSING function as

IF MISSING (DATE1) THEN DO;

...

END;

where the MISSING function returns unity if the expression or argument, DATE1 in this instance, is missing. Otherwise, 0 is returned. The argument or expression to which this function may be applied may be either character or numeric.

SUMMARY

In this paper the authors have attempted to give an insight in to some of the advantages of arithmetic-type SAS functions that they have encountered in the CRO industry. The notion was to demonstrate fluidity in developing code to result in a cleaner SAS log and to ensure tighter programmatic scrutiny of input variables before processing in order to reduce debugging time. The interlinking of some of these functions demonstrates the power of the SAS System and offers the user a preferred approach that may be different from another user.

REFERENCES

SAS Online Doc Version Eight, 1999 SAS Institute Inc., Cary, NC, USA.

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.

CONTACT INFORMATION

Phil d'Almada
Rho, Inc.
6330 Quadrangle Dr., Suite 500
Chapel Hill, NC 27517

Office: 919-595-6266
Facsimile: 919-408-0999
E-mail address: phil_dalmada@rhoworld.com

Karen T. Graham
Rho, Inc.
6330 Quadrangle Dr., Suite 500
Chapel Hill, NC 27517

Office: 919-595-6319
Facsimile: 919-408-0999
E-mail address: karen_graham@rhoworld.com