Understanding and Using Functions

Frank C. DiIorio
Advanced Integrated Manufacturing Solutions, Co.
Durham NC

Introduction

Let's start by admitting that programmers are, at heart, rather lazy. We want procedures to do our sorting, printing, and analysis. We want formats to display the number '5' as 'Well above average'. In essence, we don't want to code any more statements than we have to. With that in mind, let's look at another code and time-saving feature of the SAS System.

This paper gives an overview of functions, a powerful set of tools in the SAS System. Functions are a set of predefined routines that come with the SAS System. They perform a wide range of activities, and often reduce complex computations that would require arduous and error-prone DATA step coding to a single, simple statement. Not even a novice SAS programmer's toolbox is complete without a basic knowledge of system functions.

This paper introduces SAS novices to functions. Basic terminology is reviewed first, followed by usage issues common to nearly all functions. The last section of the paper describes the purpose and syntax of some of the more commonly used functions. Bear in mind that this paper is simply an overview of a broad and sometimes complex topic. The reader should consult SAS Institute documentation for the definitive, exhaustive description of the purpose, limitations, and uses of the functions.

Fundamentals

The logic that's common to all functions is straightforward. Two components of function syntax – the function name and its parameters – identify what is to be performed. The first component is the function name. It identifies the action that the function performs – identify the minimum of a list of numbers (the MIN function), locate the third word in a character variable (the SCAN function), and so on. The name usually gives some idea of the activities performed by the function.

The second function component is a list of *parameters* (sometimes referred to as *arguments*) enclosed in parentheses. Notice in the above description of the function names we said "minimum *of* a list" and "third word *in* a character variable." The "of" and "in" identify *what* the function should operate on. Putting these two pieces together, let's look at two complete uses of these functions:

```
min_year = min(fy2001q1, fy2001q2,
               fy2001q3);
file_name = scan(dir_line, 3, ' ');
```

The first statement creates the numeric variable MIN_YEAR, which is the minimum of three arguments, fy2001q1, fy2001q2, and fy2001q3. The second statement creates a character variable, FILE_NAME, which is the third piece of variable DIR_LINE, the pieces being delimited by blanks.

The functions are said to be "called" – in the first statement, we called the MIN function, in the second, we called the SCAN function. Functions are said to "return" a value – in the first statement, we call the MIN function and it returns a value, stored in MIN_YEAR, that is the minimum of the three arguments passed to it.

Both the idea and the syntax are simple – specify the appropriate function name and give the function parameters and results are returned. Just as procedures perform a great deal of work with relatively few statements, so do functions simplify potentially tedious calculations. Before looking at what specific functions do, let's look at some logical and syntactical issues common to all of them.

Syntax and Usage

Before using this paper or the SAS documentation's description of the *many* available functions, consider the following points carefully:

Functions Can Be Used Pretty Much Anywhere. Functions usually perform some form of calculation, and calculations are usually considered in the context of the DATA step. Keep in mind that, with some documented exceptions, functions can be used anywhere an evaluation of constants and/or variables can take place.

Here are some non-DATA step examples. Look at them more for their use outside the DATA step than for the actual operation they perform:

```
proc print data=subset
     (where=(index(vin, 'NR') > 0));

proc freq data=mast01;
tables grp / missing;
where nmiss(of fy1999q1–fy2001q4) > 2;
```

Names Matter (Gotcha #1). There are specific names for specific function activities. These cannot be reassigned (changing MIN to MINIMUM, for example). What you can do, and probably won't want to, once you see the result, is define an array with the same name as a function. Watch what happens:

```
data phase1;
set master;
array min(4);  /* each group's minimum */
do i = 1 to 4;
   min(i) = min(of group1-group4);
end;
```

SAS gets confused – is MIN a reference to the function or the array? The default action is to recognize MIN as an array, effectively disabling the MIN function. The following message is printed in the SAS Log:

```
WARNING: An array is being defined with the
same name as a SAS-supplied or user-defined
function. Parenthesized references involving
this name will be treated as array refer-
ences and not function references.
```

Bottom line: there are lots of words in the language. Define arrays with names that don't conflict with function names.

Names Matter (Gotcha #2). Programmers new to SAS, or those regularly moving back and forth between SAS and other languages, must be careful not to assume that function *x* in one language does the same thing in another language. Don't make assumptions. Read the documentation carefully, and be sure the functionality is identical.

If you need motivation in this regard, consider the subtle difference in this example: the TRIM function is in EXCEL and SAS, and performs basically the same activity (trimming blanks from a character variable). In EXCEL, both leading and trailing blanks are trimmed, but SAS trims only the *trailing* blanks. It's usually easier to review documentation for the functions than it is to debug their unadvised usage.

Look at Data Types Carefully. Some functions require numeric arguments, others require character arguments. Yet others require a mix of these data types. The type(s) of the argument(s) does not influence the value returned by the function. The LENGTH function, for example, returns the location of the last non-blank character in a variable. The function requires a character argument but returns a numeric value.

The Number of Arguments Varies. The number of arguments required by a function will, of course, depend on the type of work the function performs. Even using the same function, though, the number of arguments can vary. MIN and other descriptive statistic functions can handle a varying number of arguments, provided there are enough values to perform the required task (you need at least three arguments to calculate skewness, for example). Other functions expect "n" arguments and will make assumptions if they do not receive the full "n" – the third argument to the SUBSTR function, for example, is the number of positions to extract from a character variable. If omitted from the SUBSTR call, the default behavior is to subset to the rightmost position in the variable.

Parameter Order May Matter. Some functions do not care about the order in which arguments are specified. The SUM function, for example, performs an action (addition) that is, by nature, indifferent to order. Other functions are not so forgiving, and assign specific meanings to arguments. The first argument to the ROUND function, for example, is a numeric value. The second argument is the rounding unit ("round to the nearest …"). SAS is often unable to detect misspecified parameters because they may make *syntactical* sense but do not have *logical* validity. Consider the following statements:

```
inc1 = round(income, 1000);
inc2 = round(1000, income);
```

INC1 is variable INCOME rounded to the nearest 1000, while INC2 is 1000 rounded to the nearest INCOME. Both statements are syntactically valid, but only INC1 makes sense. It's important to realize that from SAS' perspective, *both* statements are acceptable. It's up to the programmer to become familiar with parameter order and meaning (and then, of course, follow through and write the statement correctly!).

Watch Out for Range Restrictions. Some functions will process any values, provided their data types are correct. Others require one or all values to be in a range of values. The restriction may be known prior to coding (the square root function SQRT cannot process a negative value), while other limits are imposed by the nature of the operation (you cannot use SUBSTR to go beyond the length of a character variable). In all cases, if you specify one or more invalid arguments, SAS will issue a message in the Log and the function will return a missing value. Suppose we specify this statement:

```
rounded = round(rate, rate_factor);
```

If RATE_FACTOR is a missing value or negative, ROUNDED will be set to missing, and the SAS Log will contain a message similar to:

```
NOTE: Argument 2 to function ROUND at line
1449 column 11 is invalid.
```

Missing Values Sometimes Matter. Missing values in one or more arguments may influence the value returned by the function. If we specify a missing value where we should have entered the starting location of a substring, then SAS will display an error message and the function will return a missing value.

Other functions are not as fussy. Most descriptive statistic functions – SUM, MEAN, RANGE, and the like – will operate on any non-missing arguments. This is an important distinction, since a simple assignment statement not using functions will create a missing value if *any* of its operands is missing. Examine the following code:

```
q1 = 300;  q2 = 350;  q3 = 250;  q4 = .;
year_tot_1 = q1 + q2 + q3 + q4;
year_tot_2 = sum(of q1-q4);
```

The first assignment statement has a form which requires all operands to be numeric and non-missing. Since Q4 is missing, the result, YEAR_TOT_1, will be missing. The second assignment uses the SUM function. Its parameters match the operands of the previous statement, but it returns a value because the function uses only non-missing values. YEAR_TOT_2 is 900. The impact of missing values is significant here and in other statistical functions.

Here, as in the points noted above, we emphasize the need to carefully review the function's documentation prior to writing the program.

Specify Character Variable Lengths. If the function is returning a character variable, specify the length of the variable to avoid unanticipated padding. In this example, variable QUOTED is length 200, regardless of the length of TEXT.

```
data revised;
set temp2;
```

```
quoted = quote(text);
run;
```

Adding a LENGTH statement to the program brings QUOTED to a more reasonable length (TEXT's length – assume $20 – plus two positions for quotes):

```
data revised;
set temp2;
length quoted $22;
quoted = quote(text);
run;
```

Parameter Specification Can Vary Greatly. Arguments to most functions can be constants, variables, or expressions (including other function calls). In general, the function will accept a value as long as the specification results in a value that is appropriate. Here are some examples. As before, look at them for style rather than exact meaning:

```
small_pair = min(min(c1,c2), min(d1,d2));
piece = substr(line, 1, length(line) - 3);
tot = sub + reg + sum(ot1, ot2, ot3);
```

The first two statements are examples of functions being used as arguments to other functions. This is commonly referred to as "nesting." The statements are concise, but bear in mind the difficulty of debugging them. What is the minimum of C1 and C2? Of D1 and D2? With the statement written as is, you can't tell. It may be easier in the long run to break up the statement:

```
min_c = min(c1, c2);
min_d = min(d1, d2);
small_pair = min(min_c, min_d);
```

Finally, remember that SAS will do its best to reduce each argument to its simplest form. This behavior is predictable, but can lead to unexpected results if only casually recalled. Consider this code fragment:

```
v1 = 3;  v2 = 4;  v3 = 10;  v4 = 6;
max_v = max(v1-v4);
```

Looking at the assignment statements, you would expect the value of MAX_V to be 10. Instead, it is –3. Why? Because SAS will resolve the argument before it is passed to the function. Instead of seeing a list – "V1 through V4" – SAS sees an arithmetic expression – "V1 minus V4." Thus only 3 minus 6, or –3, is passed to the MAX function and –3, by definition, is the maximum, since no other parameters are available for evaluation. Fortunately, MAX and other statistical functions have a way to specify V1-V4 as a list, rather than an expression. It is shown below and also noted in the last section's description of various functions:

```
max_v = max(of v1-v4);
```

Functions Can Be Used "On the Fly." The previous point's examples hinted at a powerful capability of using functions within SAS. Rather than store a function result in a variable, it's possible to make it transient, available only for purposes of evaluation and not for storage as a variable. Here we show the ability to use functions in decision-making statements.

```
if index(line, '.txt') > 0 then do;
   code, code, and more code

select (quarter(start_date));
  when (1) do;
   code, code, and more code
```

CALL Routines Are Close Cousins of Functions. A separate set of routines, called CALL routines (for obvious reasons that we'll soon see) are equivalent in spirit, if not syntax, to functions. The idea is the same as functions – create a value by passing a certain number of parameters of a certain data type in a certain order. The principal difference is in their invocation, as shown in the examples below:

```
call label(var_names(i), label_text);
if eof then call symput
   ('count', put(_nread, 3.));
```

Functions would return a transient, "on the fly" value or have their value stored in a variable. By contrast, CALL routines typically specify operands and results in the parameter list.

For the purposes of this paper, CALL routines and functions are treated identically. Their syntactical differences and distinctions are clearly highlighted in SAS documentation.

Commonly-Used Functions

A complete list of functions, grouped by category, is found in the Appendix. This section takes a closer look at some of the more commonly used functions and gives examples of their use. The order and category names correspond to the SAS Online Doc. Yet again – refer to SAS Institute documentation for a description of parameters and other usage notes.

Category/Name	Description and example of usage
array	
dim	Number of elements in an array. do i = 1 to dim(list);
character	
compbl	Removes consecutive blanks from a string. old = 'much extra space'; new = compbl(old); NEW becomes 'much extra space'

compress	Removes characters from a string.
	old = 'Chapel Hill, NC – 27516';
	new1 = compress(old);
	new2 = compress(old, ',-');
	NEW1 becomes 'ChapelHill,NC-27516'
	NEW2 becomes 'Chapel Hill NC 27516'
	You could use COMPBL on NEW2 to remove consecutive blanks.

index	Gives the starting position of a string within a string.
	string = '\temp\examples1.sas';
	loc1 = index(string, '.sas');
	loc2 = index(string, '.SAS');
	loc3 = index(upcase(string), '.SAS');
	LOC1 equals 1, LOC2 equals 0 (not found), LOC3 equals 16

left	Left-aligns a string old = ' leading blanks'; new = left(old); NEW equals 'leading blanks'
length	Returns the length (rightmost non-blank character) of a string. length old $40; old = 'Short'; len = length(old);

Category/Name	Description and example of usage
	LEN equals 5.
lowcase	Lower-cases a string. old = 'Mixed Case'; new = lowcase(old); NEW equals 'mixed case'
quote	Encloses a string in double quotes. old = 'WUNC'; /* Length of OLD = 10 */ length new $12; /* +2 to allow for quotes */ new = quote("WUNC"); NEW equals "WUNC " /* quotes are part of the value */
repeat	Repeats a string "n" times. old = 'dog'; new = repeat(old, 4); NEW equals 'dogdogdogdogdog' Remember to set a length for NEW!
reverse	Reverses a string. old = 'Looks OK'; new = reverse(old); NEW equals 'KO skooL'
right	Right justifies a string. old = 'Cats and Dogs'; /* OLD is $15 */ new = right(old); NEW equals ' Cats and Dogs'

scan	Scans a string for a character expression ('word') using a default or user-specified word.
	old = 'temp\filexxx.dat';
	new1 = scan(old, 1);
	new2 = scan(old, 2, '.\');
	new3 = scan(old, -1, '.\');
	NEW1 equals 'temp\filexxx'
	NEW2 equals 'filexxx'
	NEW3 equals 'dat'

substr	Extracts or replaces a portion of a string.
	old = 'Chapel Hill NC 27516';
	new = substr(old, length(old)-4);
	NEW equals '27516'
	old = 'J*V87';
	if substr(old, 2, 1) = '*' then do;

translate	Changes all occurrences of one character in a string to another. old = 'Line1*Line2*Line3'; new = translate(old, '/', '*'); NEW equals 'Line1/Line2/Line3'
tranwrd	Similar to TRANSLATE, but at the word level. Parameter order is different (from-to, rather than to-from!) old = 'Mrs. Smith'; new = tranwrd(old, 'Mrs.', 'Sra.'); NEW equals "Sra. Smith"
upcase	Upper-cases a string. old = 'Mixed Case'; new = upcase(old); NEW equals 'MIXED CASE'
date and time	
date	Returns the current date as a SAS date value today = date() TODAY is 15138 (June 12, 2001 if formatted)
datetime	Returns the current date-time as a SAS datetime value. rightnow =datetime(); RIGHTNOW is 1307985980.7 (12JUN2001:17:26:21 if formatted)
day / month / year	Extract the day, month, and year numbers from a SAS date value. curr_day = day(today()); CURR_DAY equals 12 if today is June 12, 2001.
hour / minute / second	Extracts the hour, minute, and second from a SAS time value. curr_hr = hour(onset); CURR_HR equals 7 if ONSET is 7:04.

Category/Name	Description and example of usage
intck	Returns the number of intervals between two dates, times, or date-times. qtr=intck('qtr','01jan2001'd,'01dec2001'd); QTR equals 3
intnx	Advances a date, time, or date-time by a specified interval. yr=intnx('year','15mar99'd,2); YR equals 14976 (01JAN01 if formatted)
mdy	Creates a SAS date value from user-specified month, day, and year. sas_date = mdy(1, 1, 2001); SAS_DATE is 14976

time /
today
	Return the current time and date. Note that these functions do not require parameters. The parentheses are necessary for SAS to make the distinction between the function call to DATE and TIME and variables of the same name.
	curr_time = time();
	curr_date = date();
	CURR_TIME is 63666.44 (17:41 if formatted)
	CURR_DATE is 15138 (June 12, 2001 if formatted)

descriptive statistics	

all functions	See Appendix A for details.
	The function names correspond to statistics available in the MEANS procedure.
	General form of usage is:
	function_name(arg1, arg2, …)
	function_name(OF base1-baseN)
	For example:
	tot = sum(ne, se, nw, sc, pc, mw);
	tot = sum(of r1-r5);
	tot = sum(of r1-r5, intl);

macro	
symput	CALL routine if eof then call symput('goodones', put(_n,3.)); Macro variable GOODONES contains the value of variable _N.
symget	Retrieves the value of a macro variable. status = symget('stat'); DATA step variable STATUS equals the value of macro variable STAT.
math	

abs	Returns the absolute value of a numeric variable.
	old = -3;
	new = abs(old);
	NEW equals 3

fact	Returns the factorial of an integer. fact = fact(5); FACT equals 120
log / log10 / log2	Return the natural, base 10, and base 2 logarithms of a positive number.

mod	Returns the remainder of a division.
	old = 100;
	new1 = mod(old, 10);
	new2 = mod(old, 8);
	NEW1 equals 0 (no remainder)
	NEW2 equals 4 (4 left over when 100 is divided by 8)

random number	
rannor / ranuni	As CALL routines, they return random variates from normal and uniform distributions. call ranuni(-1);
rannor / ranuni	As functions, they return random variates from normal and uniform distributions. The CALL routines greater control over seed values. call rannor(-1);
special	
system	CALL routine, it submits a host operating system command for execution.

Category/Name	Description and example of usage
	call system("dir p:\qc\meas*.txt /s > c:\temp\dir.txt"); Creates text file c:\temp\dir.txt, which contains the output from a directory command.
input	Allows a character variable to be read using standard SAS informats. This is a way to convert character values to numeric. char_date = '2001/08/19'; num_date = input(char_date, yymmdd10.); NUM_DATE has a numeric data type, with a value of 15206.
put	The reverse of INPUT, it writes formatted character or numeric variables to a target character variable. sales_c = put(sales, dollar8.) || ' - ' || put(sales, salefmt.); User-written format SALEFMT might result in a SALES_C value such as "120,300 – Time to buy a Lexus!"
system	Issues an operating system command and captures its return code. rc = system('cd t:\prod\rpts\graphs');
state and ZIP code	

FIPNAME / FIPSTATE /	Converts FIPS codes to state names and postal codes.
	fip_state = 37;
	name = fipname(fip_state);
	postal = fipstate(fip_state);
	NAME equals "NORTH CAROLINA"
	POSTAL equals "NC"

STFIPS / STNAME	Converts state postal codes to FIPS codes and state names.
	postal = 'NC';
	name = stname(postal);
	postal = stfips(postal);
	NAME equals "NORTH CAROLINA"
	POSTAL equals "NC"

ZIPFIPS / ZIPNAME / ZIPSTATE	Converts ZIP codes to FIPS codes, state names, and state postal codes.
	zip = '27516';
	name = zipname(zip);
	postal = zipfips(zip);
	fips = zipfips(zip);
	NAME equals "NORTH CAROLINA"
	POSTAL equals "NC"
	FIPS = 37;

truncation	

ceil	Rounds up to the nearest integer.
	old = 2.3;
	new = ceil(old);
	NEW equals 3.

floor	Rounds down to the nearest integer.
	old = 2.3;
	new = ceil(old);
	NEW equals 2.

int	Returns the integer portion of a number.
	old = -8.9;
	new = int(old);
	NEW equals –8.

round	Rounds to the nearest rounding unit (default rounding unit is 1).
	old = 100.53;
	new1 = round(old);
	new2 = round(old, .1);
	NEW1 equals 101
	NEW2 equals 100.5

variable control	
label	CALL routine, it returns the value of a variable's label. label old = "Old's nondescript label"; call label(old, label_value); LABEL_VALUE equals "Old's nondescript label"

vname	CALL routine, it returns the name of a variable.
	length name $32;
	array temp(*) st-block;
	do i = 1 to dim(temp);
	call vname(temp(i), name);
	put name=;
	end;
	This code writes the name of each element in TEMP.

Questions? Comments?

Your feedback is always welcome. Contact the author at fcd1@mindspring.com.

Appendix A: Functions and CALL Routines by Category

The table in this appendix is taken directly from the Version 8.0 SAS Online Doc. It gives an idea of the power and versatility of the functions and CALL routines that come with the SAS System. For details, of course, refer to the specific help file or other SAS documentation.

Array	
DIM	Returns the number of elements in an array
HBOUND	Returns the upper bound of an array
LBOUND	Returns the lower bound of an array

Bitwise Logical Operations	
BAND	Returns the bitwise logical AND of two arguments
BLSHIFT	Returns the bitwise logical left shift of two arguments
BNOT	Returns the bitwise logical NOT of an argument
BOR	Returns the bitwise logical OR of two arguments
BRSHIFT	Returns the bitwise logical right shift of two arguments
BXOR	Returns the bitwise logical EXCLUSIVE OR of two arguments

Character String Matching	
CALL RXCHANGE	Changes one or more substrings that match a pattern
CALL RXFREE	Frees memory allocated by other regular expression (RX) functions and CALL routines
CALL RXSUBSTR	Finds the position, length, and score of a substring that matches a pattern
RXMATCH	Finds the beginning of a substring that matches a pattern and returns a value
RXPARSE	Parses a pattern and returns a value

Character	
BYTE	Returns one character in the ASCII or the EBCDIC collating sequence
COLLATE	Returns an ASCII or EBCDIC collating sequence character string
COMPBL	Removes multiple blanks from a character string
COMPRESS	Removes specific characters from a character string
DEQUOTE	Removes quotation marks from a character value
INDEX	Searches a character expression for a string of characters
INDEXC	Searches a character expression for specific characters
INDEXW	Searches a character expression for a specified string as a word
LEFT	Left aligns a SAS character expression
LENGTH	Returns the length of an argument
LOWCASE	Converts all letters in an argument to lowercase
MISSING	Returns a numeric result that indicates whether the argument contains a missing value
QUOTE	Adds double quotation marks to a character value
RANK	Returns the position of a character in the ASCII or EBCDIC collating sequence
REPEAT	Repeats a character expression
REVERSE	Reverses a character expression
RIGHT	Right aligns a character expression
SCAN	Selects a given word from a character expression
SOUNDEX	Encodes a string to facilitate searching
SPEDIS	Determines the likelihood of two words matching, expressed as the asymmetric spelling distance between the two words
SUBSTR (left of =)	Replaces character value contents
SUBSTR	Extracts a substring from an argument
TRANSLATE	Replaces specific characters in a character expression

Character

TRANWRD	Replaces or removes all occurrences of a word in a character string
TRIM	Removes trailing blanks from character expressions and returns one blank if the expression is missing
TRIMN	Removes trailing blanks from character expressions and returns a null string (zero blanks) if the expression is missing
UPCASE	Converts all letters in an argument to uppercase
VERIFY	Returns the position of the first character that is unique to an expression

Double-Byte Character Set (DBCS)

KCOMPARE	Returns the result of a comparison of character strings
KCOMPRESS	Removes specific characters from a character string
KCOUNT	Returns the number of double-byte characters in a string
KINDEX	Searches a character expression for a string of characters
KINDEXC	Searches a character expression for specific characters
KLEFT	Left aligns a SAS character expression by removing unnecessary leading DBCS blanks and SO/SI
KLENGTH	Returns the length of an argument
KLOWCASE	Converts all letters in an argument to lowercase
KREVERSE	Reverses a character expression
KRIGHT	Right aligns a character expression by trimming trailing DBCS blanks and SO/SI
KSCAN	Selects a given word from a character expression
KSTRCAT	Concatenates two or more character strings
KSUBSTR	Extracts a substring from an argument
KSUBSTRB	Extracts a substring from an argument based on byte position
KTRANSLATE	Replaces specific characters in a character expression
KTRIM	Removes trailing DBCS blanks and SO/SI from character expressions
KTRUNCATE	Truncates a numeric value to a specified length
KUPCASE	Converts all single-byte letters in an argument to uppercase
KUPDATE	Inserts, deletes, and replaces character value contents
KUPDATEB	Inserts, deletes, and replaces character value contents based on byte unit
KVERIFY	Returns the position of the first character that is unique to an expression

Date and Time

DATDIF	Returns the number of days between two dates
DATE	Returns the current date as a SAS date value
DATEJUL	Converts a Julian date to a SAS date value
DATEPART	Extracts the date from a SAS datetime value
DATETIME	Returns the current date and time of day as a SAS datetime value
DAY	Returns the day of the month from a SAS date value
DHMS	Returns a SAS datetime value from date, hour, minute, and second
HMS	Returns a SAS time value from hour, minute, and second values
HOUR	Returns the hour from a SAS time or datetime value
INTCK	Returns the integer number of time intervals in a given time span
INTNX	Advances a date, time, or datetime value by a given interval, and returns a date, time, or datetime value
JULDATE	Returns the Julian date from a SAS date value
JULDATE7	Returns a seven-digit Julian date from a SAS date value
MDY	Returns a SAS date value from month, day, and year values
MINUTE	Returns the minute from a SAS time or datetime value
MONTH	Returns the month from a SAS date value
QTR	Returns the quarter of the year from a SAS date value
SECOND	Returns the second from a SAS time or datetime value
TIME	Returns the current time of day
TIMEPART	Extracts a time value from a SAS datetime value
TODAY	Returns the current date as a SAS date value
WEEKDAY	Returns the day of the week from a SAS date value
YEAR	Returns the year from a SAS date value
YRDIF	Returns the difference in years between two dates
YYQ	Returns a SAS date value from the year and quarter

Descriptive Statistics

CSS	Returns the corrected sum of squares
CV	Returns the coefficient of variation
KURTOSIS	Returns the kurtosis
MAX	Returns the largest value
MEAN	Returns the arithmetic mean (average)
MIN	Returns the smallest value
MISSING	Returns a numeric result that indicates whether the argument contains a missing value
N	Returns the number of nonmissing values
NMISS	Returns the number of missing values
ORDINAL	Returns any specified order statistic
RANGE	Returns the range of values
SKEWNESS	Returns the skewness
STD	Returns the standard deviation
STDERR	Returns the standard error of the mean
SUM	Returns the sum of the nonmissing arguments
USS	Returns the uncorrected sum of squares
VAR	Returns the variance

External Files

DCLOSE	Closes a directory that was opened by the DOPEN function and returns a value
DINFO	Returns information about a directory

External Files

DNUM	Returns the number of members in a directory
DOPEN	Opens a directory and returns a directory identifier value
DOPTNAME	Returns directory attribute information
DOPTNUM	Returns the number of information items that are available for a directory
DREAD	Returns the name of a directory member
DROPNOTE	Deletes a note marker from a SAS data set or an external file and returns a value
FAPPEND	Appends the current record to the end of an external file and returns a value
FCLOSE	Closes an external file, directory, or directory member, and returns a value
FCOL	Returns the current column position in the File Data Buffer (FDB)
FDELETE	Deletes an external file or an empty directory
FEXIST	Verifies the existence of an external file associated with a fileref and returns a value
FGET	Copies data from the File Data Buffer (FDB) into a variable and returns a value
FILEEXIST	Verifies the existence of an external file by its physical name and returns a value
FILENAME	Assigns or deassigns a fileref for an external file, directory, or output device and returns a value
FILEREF	Verifies that a fileref has been assigned for the current SAS session and returns a value
FINFO	Returns the value of a file information item
FNOTE	Identifies the last record that was read and returns a value that FPOINT can use
FOPEN	Opens an external file and returns a file identifier value
FOPTNAME	Returns the name of an item of information about a file
FOPTNUM	Returns the number of information items that are available for an external file
FPOINT	Positions the read pointer on the next record to be read and returns a value
FPOS	Sets the position of the column pointer in the File Data Buffer (FDB) and returns a value
FPUT	Moves data to the File Data Buffer (FDB) of an external file, starting at the FDB's current column position, and returns a value
FREAD	Reads a record from an external file into the File Data Buffer (FDB) and returns a value
FREWIND	Positions the file pointer to the start of the file and returns a value
FRLEN	Returns the size of the last record read, or, if the file is opened for output, returns the current record size
FSEP	Sets the token delimiters for the FGET function and returns a value
FWRITE	Writes a record to an external file and returns a value
MOPEN	Opens a file by directory id and member name, and returns the file identifier or a 0
PATHNAME	Returns the physical name of a SAS data library or of an external file, or returns a blank
SYSMSG	Returns the text of error messages or warning messages from the last data set or external file function execution
SYSRC	Returns a system error number

External Routines

CALL MODULE	Calls the external routine without any return code
CALL MODULEI	Calls the external routine without any return code (in IML environment only)
MODULEC	Calls an external routine and returns a character value
MODULEIC	Calls an external routine and returns a character value (in IML environment only)
MODULEIN	Calls an external routine and returns a numeric value (in IML environment only)
MODULEN	Calls an external routine and returns a numeric value

Financial

COMPOUND	Returns compound interest parameters
CONVX	Returns the convexity for an enumerated cashflow
CONVXP	Returns the convexity for a periodic cashflow stream, such as a bond
DACCDB	Returns the accumulated declining balance depreciation
DACCDBSL	Returns the accumulated declining balance with conversion to a straight-line depreciation
DACCSL	Returns the accumulated straight-line depreciation
DACCSYD	Returns the accumulated sum-of-years-digits depreciation
DACCTAB	Returns the accumulated depreciation from specified tables
DEPDB	Returns the declining balance depreciation
DEPDBSL	Returns the declining balance with conversion to a straight-line depreciation
DEPSL	Returns the straight-line depreciation
DEPSYD	Returns the sum-of-years-digits depreciation
DEPTAB	Returns the depreciation from specified tables
DUR	Returns the modified duration for an enumerated cashflow
DURP	Returns the modified duration for a periodic cashflow stream, such as a bond
INTRR	Returns the internal rate of return as a fraction
IRR	Returns the internal rate of return as a percentage
MORT	Returns amortization parameters
NETPV	Returns the net present value as a fraction
NPV	Returns the net present value with the rate expressed as a percentage
PVP	Returns the present value for a periodic cashflow stream, such as a bond
SAVING	Returns the future value of a periodic saving
YIELDP	Returns the yield-to-maturity for a periodic cashflow stream, such as a bond

Hyperbolic

COSH	Returns the hyperbolic cosine
SINH	Returns the hyperbolic sine
TANH	Returns the hyperbolic tangent

Macro

CALL EXECUTE	Resolves an argument and issues the resolved value for execution
CALL SYMPUT	Assigns DATA step information to a macro variable
RESOLVE	Returns the resolved value of an argument after it has been processed by the macro facility
SYMGET	Returns the value of a macro variable during DATA step execution

Mathematical

ABS	Returns the absolute value
AIRY	Returns the value of the airy function
CNONCT	Returns the noncentrality parameter from a chi-squared distribution
COMB	Computes the number of combinations of **n** elements taken **r** at a time and returns a value
CONSTANT	Computes some machine and mathematical constants and returns a value
DAIRY	Returns the derivative of the airy function
DEVIANCE	Computes the deviance and returns a value
DIGAMMA	Returns the value of the DIGAMMA function
ERF	Returns the value of the (normal) error function
ERFC	Returns the value of the complementary (normal) error function
EXP	Returns the value of the exponential function
FACT	Computes a factorial and returns a value
FNONCT	Returns the value of the noncentrality parameter of an F distribution
GAMMA	Returns the value of the Gamma function
IBESSEL	Returns the value of the modified □essel function
JBESSEL	Returns the value of the □essel function
LGAMMA	Returns the natural logarithm of the Gamma function
LOG	Returns the natural (base e) logarithm
LOG10	Returns the logarithm to the base 10
LOG2	Returns the logarithm to the base 2
MOD	Returns the remainder value
PERM	Computes the number of permutations of **n** items taken **r** at a time and returns a value
SIGN	Returns the sign of a value
SQRT	Returns the square root of a value
TNONCT	Returns the value of the noncentrality parameter from the student's **t** distribution
TRIGAMMA	Returns the value of the TRIGAMMA function

Probability

CDF	Computes cumulative distribution functions
LOGPDF	Computes the logarithm of a probability (mass) function
LOGSDF	Computes the logarithm of a survival function
PDF	Computes probability density (mass) functions
POISSON	Returns the probability from a Poisson distribution
PROBBETA	Returns the probability from a beta distribution
PROBBNML	Returns the probability from a binomial distribution
PROBBNRM	Computes a probability from the bivariate normal distribution and returns a value
PROBCHI	Returns the probability from a chi-squared distribution
PROBF	Returns the probability from an **F** distribution
PROBGAM	Returns the probability from a gamma distribution
PROBHYPR	Returns the probability from a hypergeometric distribution
PROBMC	Computes a probability or a quantile from various distributions for multiple comparisons of means, and returns a value
PROBNEGB	Returns the probability from a negative binomial distribution
PROBNORM	Returns the probability from the standard normal distribution
PROBT	Returns the probability from a **t** distribution
SDF	Computes a survival function

Quantile

BETAINV	Returns a quantile from the beta distribution
CINV	Returns a quantile from the chi-squared distribution
FINV	Returns a quantile from the **F** distribution
GAMINV	Returns a quantile from the gamma distribution
PROBIT	Returns a quantile from the standard normal distribution
TINV	Returns a quantile from the **t** distribution

Random Number

CALL RANBIN	Returns a random variate from a binomial distribution
CALL RANCAU	Returns a random variate from a Cauchy distribution
CALL RANEXP	Returns a random variate from an exponential distribution
CALL RANGAM	Returns a random variate from a gamma distribution
CALL RANNOR	Returns a random variate from a normal distribution
CALL RANPOI	Returns a random variate from a Poisson distribution
CALL RANTBL	Returns a random variate from a tabled probability distribution
CALL RANTRI	Returns a random variate from a triangular distribution
CALL RANUNI	Returns a random variate from a uniform distribution
NORMAL	Returns a random variate from a normal distribution
RANBIN	Returns a random variate from a binomial distribution
RANCAU	Returns a random variate from a Cauchy distribution

Random Number

RANEXP	Returns a random variate from an exponential distribution
RANGAM	Returns a random variate from a gamma distribution
RANNOR	Returns a random variate from a normal distribution
RANPOI	Returns a random variate from a Poisson distribution
RANTBL	Returns a random variate from a tabled probability
RANTRI	Random variate from a triangular distribution
RANUNI	Returns a random variate from a uniform distribution
UNIFORM	Random variate from a uniform distribution

SAS File I/O

ATTRC	Returns the value of a character attribute for a SAS data set
ATTRN	Returns the value of a numeric attribute for the specified SAS data set
CEXIST	Verifies the existence of a SAS catalog or SAS catalog entry and returns a value
CLOSE	Closes a SAS data set and returns a value
CUROBS	Returns the observation number of the current observation
DROPNOTE	Deletes a note marker from a SAS data set or an external file and returns a value
DSNAME	Returns the SAS data set name that is associated with a data set identifier
EXIST	Verifies the existence of a SAS data library member
FETCH	Reads the next nondeleted observation from a SAS data set into the Data Set Data Vector (DDV) and returns a value
FETCHOBS	Reads a specified observation from a SAS data set into the Data Set Data Vector (DDV) and returns a value
GETVARC	Returns the value of a SAS data set character variable
GETVARN	Returns the value of a SAS data set numeric variable
IORCMSG	Returns a formatted error message for _IORC_
LIBNAME	Assigns or deassigns a libref for a SAS data library and returns a value
LIBREF	Verifies that a libref has been assigned and returns a value
NOTE	Returns an observation ID for the current observation of a SAS data set
OPEN	Opens a SAS data set and returns a value
PATHNAME	Returns the physical name of a SAS data library or of an external file, or returns a blank
POINT	Locates an observation identified by the NOTE function and returns a value
REWIND	Positions the data set pointer at the beginning of a SAS data set and returns a value
SYSMSG	Returns the text of error messages or warning messages from the last data set or external file function execution
SYSRC	Returns a system error number
VARFMT	Returns the format assigned to a SAS data set variable
VARINFMT	Returns the informat assigned to a SAS data set variable
VARLABEL	Returns the label assigned to a SAS data set variable
VARLEN	Returns the length of a SAS data set variable
VARNAME	Returns the name of a SAS data set variable
VARNUM	Returns the number of a variable's position in a SAS data set
VARTYPE	Returns the data type of a SAS data set variable

Special

ADDR	Returns the memory address of a variable
CALL POKE	Writes a value directly into memory
CALL SYSTEM	Submits an operating environment command for execution
DIF	Returns differences between the argument and its **n**th lag
GETOPTION	Returns the value of a SAS system or graphics option
INPUT	Returns the value produced when a SAS expression that uses a specified informat expression is read
INPUTC	Enables you to specify a character informat at run time
INPUTN	Enables you to specify a numeric informat at run time
LAG	Returns values from a queue
PEEK	Stores the contents of a memory address into a numeric variable
PEEKC	Stores the contents of a memory address into a character variable
POKE	Writes a value directly into memory
PUT	Returns a value using a specified format
PUTC	Enables you to specify a character format at run time
PUTN	Enables you to specify a numeric format at run time
SYSGET	Returns the value of the specified operating environment variable
SYSPARM	Returns the system parameter string
SYSPROD	Determines if a product is licensed
SYSTEM	Issues an operating environment command during a SAS session

State Postal, FIPS, and ZIP Codes

FIPNAME	Converts FIPS codes to uppercase state names
FIPNAMEL	Converts FIPS codes to mixed case state names
FIPSTATE	Converts FIPS codes to two-character postal codes
STFIPS	Converts state postal codes to FIPS state codes
STNAME	Converts state postal codes to uppercase state names
STNAMEL	Converts state postal codes to mixed case state names
ZIPFIPS	Converts ZIP codes to FIPS state codes
ZIPNAME	Converts ZIP codes to uppercase state names
ZIPNAMEL	Converts ZIP codes to mixed case state names
ZIPSTATE	Converts ZIP codes to state postal codes

Trigonometric

ARCOS	Returns the arccosine
ARSIN	Returns the arcsine

Trigonometric	
ATAN	Returns the arctangent
COS	Returns the cosine
SIN	Returns the sine
TAN	Returns the tangent

Truncation	
CEIL	Returns the smallest integer that is greater than or equal to the argument
FLOOR	Returns the largest integer that is less than or equal to the argument
FUZZ	Returns the nearest integer if the argument is within 1E-12
INT	Returns the integer value
ROUND	Rounds to the nearest round-off unit
TRUNC	Truncates a numeric value to a specified length

Variable Control	
CALL LABEL	Assigns a variable label to a specified character variable
CALL SET	Links SAS data set variables to DATA step or macro variables that have the same name and data type
CALL VNAME	Assigns a variable name as the value of a specified variable

Variable Information	
VARRAY	Returns a value that indicates whether the specified name is an array
VARRAYX	Returns a value that indicates whether the value of the specified argument is an array
VFORMAT	Returns the format that is associated with the specified variable
VFORMATD	Returns the format decimal value that is associated with the specified variable
VFORMATDX	Returns the format decimal value that is associated with the value of the specified argument
VFORMATN	Returns the format name that is associated with the specified variable
VFORMATNX	Returns the format name that is associated with the value of the specified argument
VFORMATW	Returns the format width that is associated with the specified variable
VFORMATWX	Returns the format width that is associated with the value of the specified argument
VFORMATX	Returns the format that is associated with the value of the specified argument
VINARRAY	Returns a value that indicates whether the specified variable is a member of an array
VINARRAYX	Returns a value that indicates whether the value of the specified argument is a member of an array
VINFORMAT	Returns the informat that is associated with the specified variable
VINFORMATD	Returns the informat decimal value that is associated with the specified variable
VINFORMATDX	Returns the informat decimal value that is associated with the value of the specified argument
VINFORMATN	Returns the informat name that is associated with the specified variable
VINFORMATNX	Returns the informat name that is associated with the value of the specified argument
VINFORMATW	Returns the informat width that is associated with the specified variable
VINFORMATWX	Returns the informat width that is associated with the value of the specified argument
VINFORMATX	Returns the informat that is associated with the value of the specified argument
VLABEL	Returns the label that is associated with the specified variable
VLABELX	Returns the variable label for the value of a specified argument
VLENGTH	Returns the compile-time (allocated) size of the specified variable
VLENGTHX	Returns the compile-time (allocated) size for the value of the specified argument
VNAME	Returns the name of the specified variable
VNAMEX	Validates the value of the specified argument as a variable name
VTYPE	Returns the type (character or numeric) of the specified variable
VTYPEX	Returns the type (character or numeric) for the value of the specified argument

Web Tools	
HTMLDECODE	Decodes a string containing HTML numeric character references or HTML character entity references and returns the decoded string
HTMLENCODE	Encodes characters using HTML character entity references and returns the encoded string
URLDECODE	Returns a string that was decoded using the URL escape syntax
URLENCODE	Returns a string that was encoded using the URL escape syntax