CHAPTER

6

Functions and CALL Routines

Definitions

Definition of Functions

A *SAS function* performs a computation or system manipulation on arguments and returns a value. Most functions use arguments supplied by the user, but a few obtain their arguments from the operating environment.

In base SAS software, you can use SAS functions in DATA step programming statements, in a WHERE expression, in macro language statements, in PROC REPORT, and in Structured Query Language (SQL).

Some statistical procedures also use SAS functions. In addition, some other SAS software products offer functions that you can use in the DATA step. Refer to the documentation that pertains to the specific SAS software product for additional information about these functions.

Definition of CALL Routines

A *CALL routine* alters variable values or performs other system functions. CALL routines are similar to functions, but differ from functions in that you cannot use them in assignment statements.

All SAS CALL routines are invoked with CALL statements; that is, the name of the routine must appear after the keyword CALL on the CALL statement.

Syntax

Syntax of Functions

The syntax of a function is

function-name (*argument-1*<. . .,*argument-n*>)

function-name (OF *variable-list*)

function-name (OF *array-name*{*})

where

function-name
names the function.

argument
can be a variable name, constant, or any SAS expression, including another function. The number and kind of arguments allowed are described with individual functions. Multiple arguments are separated by a comma.

Tip: If the value of an argument is invalid (for example, missing or outside the prescribed range), SAS prints a note to the log indicating that the argument is invalid, sets _ERROR_ to 1, and sets the result to a missing value.

Examples:

- `x=max(cash,credit);`
- `x=sqrt(1500);`
- `NewCity=left(upcase(City));`
- `x=min(YearTemperature-July,YearTemperature-Dec);`
- `s=repeat('----+',16);`
- `x=min((enroll-drop),(enroll-fail));`
- `dollars=int(cash);`
-
```
if sum(cash,credit)>1000 then
    put 'Goal reached';
```

(OF *variable-list*)
can be any form of a SAS variable list, including individual variable names. If more than one variable list appears, separate them with a space.

Examples:

- `a=sum(of x  y  z);`
- The following two examples are equivalent.
  - ```
    a=sum(of x1-x10 y1-y10 z1-z10);
    a=sum(of x1-x10, of y1-y10, of z1-z10);
    ```
- `z=sum(of y1-y10);`

(OF *array-name*{*})
names a currently defined array. Specifying an array in this way causes SAS to treat the array as a list of the variables instead of processing only one element of the array at a time.

Examples:

```
array y{10} y1-y10;
x=sum(of y{*});
```

Syntax of CALL Routines

The syntax of a CALL routine is

CALL *routine-name* (*argument-1*<. . .,*argument-n*>);

where

routine-name
names a SAS CALL routine.

argument
can be a variable name, a constant, any SAS expression, an external module name, an array reference, or a function. Multiple arguments are separated by a comma. The number and kind of arguments allowed are described with individual CALL routines in the "Functions and CALL Routines" section of *SAS Language Reference: Dictionary*.

Examples:

- `call rxsubstr(rx,string,position);`
- `call set(dsid);`
- `call ranbin(Seed_1,n,p,X1);`
- `call label(abc{j},lab);`

Using Functions

Restrictions on Function Arguments

If the value of an argument is invalid, SAS prints an error message and sets the result to a missing value. Here are some common restrictions on function arguments:
```

- Some functions require that their arguments be restricted within a certain range. For example, the argument of the LOG function must be greater than 0.
- Most functions do not permit missing values as arguments. Exceptions include some of the descriptive statistic functions and financial functions.
- In general, the allowed range of the arguments is platform-dependent, such as with the EXP function.
- For some probability functions, combinations of extreme values can cause convergence problems.

Characteristics of Target Variables

Some character functions produce resulting variables, or *target variables*, with a default length of 200 bytes. Numeric target variables have a default length of 8. Character functions to which the default target variable lengths do not apply are shown in the following table.

Table 6.1 Target Variables

Function	Target Variable Type	Target Variable Length (bytes)
BYTE	character	1
COMPRESS	character	length of first argument
INPUT	character	width of informat
	numeric	8
LEFT	character	length of argument
PUT	character	width of format
REVERSE	character	length of argument
RIGHT	character	length of argument
SUBSTR	character	length of first argument
TRANSLATE	character	length of first argument
TRIM	character	length of argument
UPCASE, LOWCASE	character	length of argument
VTYPE, VTYPEX	character	1

Notes on Descriptive Statistic Functions

SAS provides functions that return descriptive statistics. Except for the MISSING function, the functions correspond to the statistics produced by the MEANS procedure. The computing method for each statistic is discussed in "SAS Elementary Statistics Procedures" in the appendix of the *SAS Procedures Guide*. SAS calculates descriptive statistics for the nonmissing values of the arguments.

Notes on Financial Functions

SAS provides a group of functions that perform financial calculations. The functions are grouped into the following types:

Table 6.2 Types of Financial Functions

Function type	Functions	Description
Cashflow	CONVX, CONVXP	calculates convexity for cashflows
	DUR, DURP	calculates modified duration for cashflows
	PVP, YIELDP	calculates present value and yield-to-maturity for a periodic cashflow
Parameter calculations	COMPOUND	calculates compound interest parameters
	MORT	calculates amortization parameters
Internal rate of return	INTRR, IRR	calculates the internal rate of return
Net present and future value	NETPV, NPV	calculates net present and future values
	SAVING	calculates the future value of periodic saving
Depreciation	DACCxx	calculates the accumulated depreciation up to the specified period
	DEPxxx	calculates depreciation for a single period

Special Considerations for Depreciation Functions

The period argument for depreciation functions can be fractional for all of the functions except DEPDBSL and DACCDBSL. For fractional arguments, the depreciation is prorated between the two consecutive time periods preceding and following the fractional period.

CAUTION:

Verify the depreciation method for fractional periods. You must verify whether this method is appropriate to use with fractional periods because many depreciation schedules, specified as tables, have special rules for fractional periods. △

Using DATA Step Functions within Macro Functions

The macro functions %SYSFUNC and %QSYSFUNC can call DATA step functions to generate text in the macro facility. %SYSFUNC and %QSYSFUNC have one difference: %QSYSFUNC masks special characters and mnemonics and %SYSFUNC does not. For more information on these functions, see %QSYSFUNC and %SYSFUNC in *SAS Macro Language: Reference*.

%SYSFUNC arguments are a single DATA step function and an optional format, as shown in the following examples:

```
%sysfunc(date(),worddate.)
%sysfunc(attrn(&dsid,NOBS))
```

You cannot nest DATA step functions within %SYSFUNC. However, you can nest %SYSFUNC functions that call DATA step functions. For example:

```
%sysfunc(compress(%sysfunc(getoption(sasautos)),
   %str(%)%(%')));
```

All arguments in DATA step functions within %SYSFUNC must be separated by commas. You cannot use argument lists that are preceded by the word OF.

Because %SYSFUNC is a macro function, you do not need to enclose character values in quotation marks as you do in DATA step functions. For example, the arguments to the OPEN function are enclosed in quotation marks when you use the function alone, but the arguments do not require quotation marks when you use them within %SYSFUNC.

```
dsid=open("sasuser.houses","i");
dsid=open("&mydata","&mode");
%let dsid=%sysfunc(open(sasuser.houses,i));
%let dsid=%sysfunc(open(&mydata,&mode));
```

You can use these functions to call all of the DATA step SAS functions except those that pertain to DATA step variables or processing. These prohibited functions are: DIF, DIM, HBOUND, INPUT, IORCMSG, LAG, LBOUND, MISSING, PUT, RESOLVE, SYMGET, and all of the variable information functions (for example, VLABEL).

Using Functions to Manipulate Files

SAS manipulates files in different ways, depending on whether you use functions or statements. If you use functions such as FOPEN, FGET, and FCLOSE, you have more opportunity to examine and manipulate your data than when you use statements such as INFILE, INPUT, and PUT.

When you use external files, the FOPEN function allocates a buffer called the File Data Buffer (FDB) and opens the external file for reading or updating. The FREAD function reads a record from the external file and copies the data into the FDB. The FGET function then moves the data to the DATA step variables. The function returns a value that you can check with statements or other functions in the DATA step to determine how to further process your data. After the records are processed, the FWRITE function writes the contents of the FDB to the external file, and the FCLOSE function closes the file.

When you use SAS data sets, the OPEN function opens the data set. The FETCH and FETCHOBS functions read observations from an open SAS data set into the Data Set Data Vector (DDV). The GETVARC and GETVARN functions then move the data to DATA step variables. The functions return a value that you can check with statements or other functions in the DATA step to determine how you want to further process your data. After the data is processed, the CLOSE function closes the data set.

For a complete listing of functions and CALL routines, see Table 6.3 on page 51. For complete descriptions and examples, see *SAS Language Reference: Dictionary*.

Using Random-Number Functions and CALL Routines

Seed Values

Random–number functions and CALL routines generate streams of random numbers from an initial starting point, called a *seed*, that either the user or the computer clock supplies. A seed must be a nonnegative integer with a value less than $2^{31}-1$ (or

2,147,483,647). If you use a positive seed, you can always replicate the stream of random numbers by using the same DATA step. If you use zero as the seed, the computer clock initializes the stream, and the stream of random numbers is not replicable.

Each random-number function and CALL routine generates pseudo-random numbers from a specific statistical distribution. Every random-number function requires a seed value expressed as an integer constant, or a variable that contains the integer constant. Every CALL routine calls a variable that contains the seed value. Additionally, every CALL routine requires a variable that contains the generated random numbers.

The seed variable must be initialized prior to the first execution of the function or CALL statement. After each execution of a function, the current seed is updated internally, but the value of the seed argument remains unchanged. After each iteration of the CALL statement, however, the seed variable contains the current seed in the stream that generates the next random number. With a function, it is not possible to control the seed values, and, therefore, the random numbers after the initialization.

Comparison of Random-Number Functions and CALL Routines

Except for the NORMAL and UNIFORM functions, which are equivalent to the RANNOR and RANUNI functions, respectively, SAS provides a CALL routine that has the same name as each random-number function. Using CALL routines gives you greater control over the seed values.

With a CALL routine, you can generate multiple streams of random numbers within a single DATA step. If you supply a different seed value to initialize each of the seed variables, the streams of the generated random numbers are computationally independent. With a function, however, you cannot generate more than one stream by supplying multiple seeds within a DATA step. The following two examples illustrate the difference.

Examples

Example 1: Generating Multiple Streams from a CALL Routine

This example uses the CALL RANUNI routine to generate three streams of random numbers from the uniform distribution, with ten numbers each. See the results in Output 6.1 on page 50.

```
options nodate pageno=1 linesize=80 pagesize=60;

data multiple(drop=i);
   retain Seed_1 1298573062 Seed_2 447801538
          Seed_3 631280;
   do i=1 to 10;
      call ranuni (Seed_1,X1);
      call ranuni (Seed_2,X2);
      call ranuni (Seed_3,X3);
      output;
   end;
run;

proc print data=multiple;
   title 'Multiple Streams from a CALL Routine';
run;
```

Output 6.1 The CALL Routine Example

Multiple Streams from a CALL Routine 1

Obs	Seed_1	Seed_2	Seed_3	X1	X2	X3
1	1394231558	512727191	367385659	0.64924	0.23876	0.17108
2	1921384255	1857602268	1297973981	0.89471	0.86501	0.60442
3	902955627	422181009	188867073	0.42047	0.19659	0.08795
4	440711467	761747298	379789529	0.20522	0.35472	0.17685
5	1044485023	1703172173	591320717	0.48638	0.79310	0.27536
6	2136205611	2077746915	870485645	0.99475	0.96753	0.40535
7	1028417321	1800207034	1916469763	0.47889	0.83829	0.89243
8	1163276804	473335603	753297438	0.54169	0.22041	0.35078
9	176629027	1114889939	2089210809	0.08225	0.51916	0.97286
10	1587189112	399894790	284959446	0.73909	0.18622	0.13269

Example 2: Assigning Values from a Single Stream to Multiple Variables

Using the same three seeds that were used in Example 1, this example uses a function to create three variables. The results that are produced are different from those in Example 1 because the values of all three variables are generated by the first seed. When you use an individual function more than once in a DATA step, the function accepts only the first seed value that you supply and ignores the rest.

```
options nodate pageno=1 linesize=80 pagesize=60;

data single(drop=i);
   do i=1 to 3;
      Y1=ranuni(1298573062);
      Y2=ranuni(447801538);
      Y3=ranuni(631280);
      output;
   end;
run;

proc print data=single;
   title 'A Single Stream across Multiple Variables';
run;
```

The following example shows the results. The values of Y1, Y2, and Y3 in this example come from the same random-number stream that was generated from the first seed. You can see this by comparing the values by observation across these three variables, with the values of X1 in Output 6.1 on page 50.

Output 6.2 The Function Example

A Single Stream across Multiple Variables 1

Obs	Y1	Y2	Y3
1	0.64924	0.89471	0.42047
2	0.20522	0.48638	0.99475
3	0.47889	0.54169	0.08225

Pattern Matching Using Regular Expression (RX) Functions and CALL Routines

You can use a special group of functions and CALL routines to match or change data according to a specific pattern that you specify. By using these functions and CALL routines, you can determine whether a given character string is in a set denoted by a pattern, or you can search a given character string for a substring in a set denoted by a pattern. You can also change a matched substring to a different substring.

This group consists of CALL RXCHANGE, CALL RXFREE, CALL RXSUBSTR, RXMATCH, and RXPARSE, and comprises the character string matching category for functions and CALL routines. For a description, see "Functions and CALL Routines by Category" on page 51. For details about how to use these functions and CALL routines, see *SAS Language Reference: Dictionary*.

Base SAS Functions for Web Applications

Four functions that manipulate Web-related content are available in base SAS software. HTMLENCODE and URLENCODE return encoded strings. HTMLDECODE and URLDECODE return decoded strings. For information about Web-based SAS tools, follow the Web Enablement link on the SAS Institute home page, at www.sas.com.

Functions and CALL Routines by Category

Table 6.3 Categories and Descriptions of Functions

Category	Function	Description
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

Category	Function	Description
	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 (right of =)	Extracts a substring from an argument
	TRANSLATE	Replaces specific characters in a character expression
	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

Category	Function	Description
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

Category	Function	Description
	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

Category	Function	Description
	DINFO	Returns information about a directory
	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

Category	Function	Description
	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

Category	Function	Description
	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

Category	Function	Description
	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 bessel function
	JBESSEL	Returns the value of the bessel 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

Category	Function	Description
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
	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

Category	Function	Description
	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

Category	Function	Description
	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 and ZIP Code	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
	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

Category	Function	Description
	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

Category	Function	Description
	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

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS Language Reference: Concepts*, Cary, NC: SAS Institute Inc., 1999. 554 pages.

SAS Language Reference: Concepts

ISBN 1–58025–441–1

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, November 1999