greatlearning
Learning for Life

Functions in SAS

Neelesh Singh
July 2020

greatlearning
Learning for Life

Module Content

- Operators
- Numerical functions
- Date Functions
- String Functions
- Financial functions
- Creating Random Variables

Operators

- Arithmetic operators to perform arithmetic calculations-
  - +,-,*,/,**
- Comparison operators to form Boolean expressions which take value 0 or 1
  - EQ, NE, GT, LT, GE, LE, IN
  - =, ^=, >, <, >=, <=
- Boolean operators
  - OR, AND , NOT

Example

- Arithmetic
  A = B+2;
  A = B*2;
  A = B**2;
- Comparison
  A NE 5;
  A > 5;
  A > (B+C)/500;
  A EQ B+45;
  A in (1,2,3,4,5)
- Boolean
  A > 5 and B > 10;
  A or B;
  Not A ;

Functions

greatlearning
Learning for Life

- A function takes one or more arguments and provides us with an output value

- Syntax
  - Functionname(argument1, argument2,..)
    - Where argument could be constant or variable/expression

Example

A=SQRT(64);
A=SUM(B,C);
A=ABS(B+C);

A= Substr(X1,1,2);
A= length(X1);

Functions – Categorization based on Data type

greatlearning
Learning for Life

Functions in SAS can be categorised in these categories:

- Numeric Functions (Mathematical / Statistical)
- Character Functions
- Date Functions (Date & Time)
- Financial Functions (Cash Flow etc)
- Call Routines & Special Functions

Example

- Numeric– MEAN, SUM, AVG, ABS
- Character – CONCAT, LENGTH, LEFT, SUBSTR
- Special Functions – RANUNI, RANNOR
- Date Time Functions – MONTH, YEAR

SAS Functions: Numeric

greatlearning
Learning for Life

Arithmetic Functions:

Function name	Description	Syntax
MAX	Returns the maximum value of arguments	*max*(*x*, *y*, *z*)
MIN	Returns the minimum value of arguments	*min*(*x*, *y*, *z*)
Mean	Returns the average value of arguments	*mean*(*x*, *y*, *z*)
N	returns the number of nonmissing values	*n*(*x*)
NMISS	NMISS function returns the number of missing values	*nmiss*(*x*)
COUNT	Counts the number of observations in the variable	count(x)
ABS	The absolute value of the argument	abs(x)
SUM	Sum of the variables	sum(x,y,z)
MOD	Returns the remainder of the argument	MOD(value, divisor)
RANUNI	Returns a random variate from a uniform distribution.	RANUNI(Seed)

Date Functions:

Function name	Description	Syntax
MONTH	Returns month value from date input arguments	Month(x)
DAY	Returns day value from date input arguments	Day(x)
YEAR	Returns year value from date input arguments	Year(x)
MDY	converts into date value	MDY(x)

Examples of some Important Frequently Used Functions

greatlearning
Learning for Life

NUMERIC FUNCTIONS

```
data  Example1;
SET  Example3;

        TotAmt =  SUM (of TranAmt1);
        TotAmt =  SUM (of vol freq);
        TotAmt =  SUM (of vol, of freq);
        TotAmt =  SUM (vol);
        TotAmt =  SUM (vol, freq);
run;
```

OTHER NUMERIC FUNCTIONS :

MIN, MAX, MEAN, MEDIAN, ABS, LOG, LOG10, LOG2, FACT, Quantile, Ranuni etc..

Use of Numeric functions

greatlearning
Learning for Life

```
data output(drop=e1 e2 e3 e4 e5 p1);
Set grades;
Total_marks=sum(e1,e2,e3,e4,e5);
Average_marks=mean(e1,e2,e3,e4,e5);
Max_marks=max(e1,e2,e3,e4,e5);
Min_marks=min(e1,e2,e3,e4,e5);
count_nonmissing=n(e1,e2,e3,e4,e5);
count_missing=nmiss(e1,e2,e3,e4,e5);
avg_marks=round(mean(of e1-e5),0.1);
Total_m=sum(of e1--p1);
run;
```

Name	Total_marks	Average_marks	Max_marks	Min_marks	count_nonmissing	count_missing	avg_marks	Total_m
Alexander Smith	412	82.4	97	69	5	0	82.4	492
John Simon	346	86.5	100	72	4	1	86.5	431
Patricia jones	480	96	99	92	5	0	96	573
Jack Benedict	319	63.8	82	49	5	0	63.8	388
Rene Porter	422	84.4	100	62	5	0	84.4	514

SAS Functions: Character Manipulation

greatlearning
Learning for Life

Function name	Description	Syntax
SUBSTR	To extract part of a string. When the SUBSTR function is used on the left side of the equal sign, it can place specified characters into an existing string.	SUBSTR(character-value, start ,<length>)
TRIM	To remove trailing blanks from a character value. This is especially useful when you want to concatenate several strings together and each string may contain trailing blanks.	TRIM(*character-value)*
COMPRESS	To remove specified characters from a character value.	COMPRESS(character-value <,'compress-list'>)
COMPBL	Converting multiple blanks to a single blank	COMPBL(character-value)
SCAN	Extracts a specified word from a character expression, where word isdefined as the characters separated by a set of specified delimiters. The length of the returned variable is 200, unless previously defined.	SCAN(character-value, n-word <,'delimiter-list'>)
CAT	To concatenate (join) two or more character strings, leaving leading and/or trailing blanks unchanged.	CAT(*string-1, string-2 <,string-n>)*
STRIP	Removes trailing and leading blanks.	STRIP(character-value)
LEFT	Aligns the character value to the left	LEFT(character-value)
TRANWRD	To substitute one or more words in a string with a replacement word or words. It works like the find and replace feature of most word processors.	TRANWRD(*character-value, from-string, to-string)*
TRANSLATE	To exchange one character value for another.	TRANSLATE(*character-value, to-1, from-1 <,… to-n, from-n>)*
UPCASE	To change all letters to uppercase.	UPCASE(character-value)
LENGTH	Length of the variable	LENGTH(character-value)

Functions That Change the Case of Characters

greatlearning
Learning for Life

Function: UPCASE
Purpose: To change all letters to uppercase.
Syntax: UPCASE(*character-value)*

Function: LOWCASE
Purpose: To change all letters to lowercase.
Syntax: LOWCASE(*character-value)*

Function: PROPCASE
Purpose: To change all words to propercase.
Syntax: PROPCASE(*character-value)*

character-value is any SAS character expression.
If a length has not been previously assigned, the length of the resulting variable will be the length of the argument.
Examples
For these examples CHAR = "ABCxyz"

Function	**Returns**
UPCASE(CHAR)	"ABCXYZ"
LOWCASE(CHAR)	“abcxyz“
PROPCASE(CHAR)	“Abcxyz”

Functions That Remove Characters from Strings

greatlearning
Learning for Life

Function: COMPRESS

Purpose: To remove specified characters from a character value.

Syntax: COMPRESS(*character-value* <,*'compress-list'*>) ; *where, character-value is any SAS character expression. compress-list is an optional list of the characters you want to remove.* If this argument is omitted, the default character to be removed is a blank. If you include a list of values to remove, only those characters will be removed. If a blank is not included in the list, blanks will not be removed.

Examples

In the examples below, CHAR = "A C123XYZ"

Function	Returns	Function	Returns
COMPRESS("A C XYZ")	"ACXYZ“	**COMPRESS**(CHAR,"0123456789")	"A CXYZ"
COMPRESS("(908) 777-1234"," (-)")	"9087771234“		

Function: COMPBL

Purpose: To replace all occurrences of two or more blanks with a single blank character. This is particularly useful for standardizing addresses and names where multiple blanks may have been entered.

Syntax: COMPBL(*character-value)*

character-value is any SAS character expression.

Example

For these examples CHAR = "A C XYZ"

Function	Returns
COMPBL(CHAR)	"A C XYZ”

Functions That Substitute Letters or Words in Strings

Function: TRANSLATE

Purpose: To exchange one character value for another. For example, you might want to change values 1–5 to the values A–E.

Syntax: TRANSLATE(*character-value, to-1, from-1 <,… to-n, from-n>)*

character-value is any SAS character expression. to-n is a single character or a list of character values. from-n is a single character or a list of characters. Each character listed in *from-n is changed to the corresponding value in to-n. If a* character value is not listed in from-n, it will be unaffected.

Examples: In these examples, CHAR = "12X45", ANS = "Y"

Function	Returns	Function	Returns
TRANSLATE(CHAR,"ABCDE","12345")	"ABXDE“	**TRANSLATE**(CHAR,'A','1','B','2','C','3','D','4','E','5')	"ABXDE"
TRANSLATE(ANS,"10","YN")	"1"		

Function: TRANWRD

Purpose: To substitute one or more words in a string with a replacement word or words. It works like the find and replace feature of most word processors.

Syntax: TRANWRD(*character-value, from-string, to-string)*

character-value is any SAS character expression. from-string is one or more characters that you want to replace with the character or characters in the to-string. *to-string is one or more characters that replace the entire from-string.*

Examples: For these examples STRING = "123 Elm Road", FROM = "Road" and TO = "Rd."

Function	Returns	Function	Returns
TRANWRD(STRING,FROM,TO)	"123 Elm Rd.“	**TRANWRD**("Now is the time","is","is not")	"Now is not the time"
TRANWRD("Mr. Rogers","Mr."," ")	" Rogers”	**TRANWRD**("one two three","four","4")	"one two three“

Functions That Extract Parts of Strings

greatlearning
Learning for Life

Function: SUBSTR

Purpose: To extract part of a string. When the SUBSTR function is used on the left side of the equal sign, it can place specified characters into an existing string.

Syntax: SUBSTR(character-value, start <,length>)

character-value is any SAS character expression.

start is the starting position within the string.length if specified, is the number of characters to include in the substring. If this argument is omitted, the SUBSTR function will return all the characters from the start position to the end of the string. If a length has not been previously assigned, the length of the resulting variable will be the length of the character-value.

Examples

For these examples, let STRING = "ABC123XYZ“

Function	**Returns**
SUBSTR(STRING,4,2)	"12"
SUBSTR(STRING,4)	"123XYZ"
SUBSTR(STRING,**LENGTH**(STRING))	"Z" (last character in the string)

Functions That Remove/Align Blanks from Strings

greatlearning
Learning for Life

Function: LEFT
Purpose: To left-align text values. A subtle but important point: LEFT doesn't "remove" the leading blanks; it moves them to the end of the string.
Syntax: LEFT(character-value) ;character-value is any SAS character expression.
Example
In these examples STRING = " ABC“ **LEFT**(STRING) returns "ABC “

Function: TRIM
Purpose: To remove trailing blanks from a character value. This is especially useful when you want to concatenate several strings together and each string may contain trailing blanks.
Syntax: TRIM(character-value) **;**character-value is any SAS character expression.
Examples
For these examples, STRING1 = "ABC " and STRING2 = " XYZ“
TRIM(STRING1) returns "ABC“ ; **TRIM**(STRING2) returns “ XYZ”

Function: STRIP
Purpose: To strip leading and trailing blanks from character variables or strings.
Syntax: STRIP(*character-value)* ***;****character-value is any SAS character expression.*
If the STRIP function is used to create a new variable, the length of that new variable will be equal to the length of the argument of the STRIP function. The STRIP function is useful when using the concatenation operator.
Examples
For these examples, let STRING = " abc “
STRIP(STRING) returns "abc" (if result was previously assigned a length of three, otherwise trailing blanks would be added)

Some more functions

greatlearning
Learning for Life

Function: SCAN (Function That Divide Strings into "Words“)
Purpose: Extracts a specified word from a character expression, where word is defined as the characters separated by a set of specified delimiters. The length of the returned variable is 200, unless previously defined.
Syntax: SCAN(*character-value, n-word <,'delimiter-list'>)*
character-value is any SAS character expression. n-word is the nth "word" in the string. If n is greater than the number of words, the SCAN function returns a value that contains no characters. If *n* is negative, the character value is scanned from right to left. A value of zero is invalid.
Examples
For these examples **STRING1 = "ABC DEF"** and **STRING2 = "ONE?TWO THREE+FOUR|FIVE"**.

Function	Returns	Function	Returns
SCAN(STRING1,2)	"DEF“	**SCAN**(STRING2,4)	"FIVE“
SCAN(STRING1,-1)	"DEF“	**SCAN**(STRING2,2," ")	"THREE+FOUR|FIVE“
SCAN(STRING1,3)	no characters	**SCAN**(STRING1,0)	An error in the SAS log

Function: LENGTH
Purpose: To determine the length of a character value, not counting trailing blanks**.** A null argument returns a value of 1.
Syntax: LENGTH(*character-value)* *;* *character-value is any SAS character expression.*
Examples
For these examples CHAR = "ABC “
LENGTH("ABC") returns 3 LENGTH(CHAR) returns 3 LENGTH(" ") returns 1

Functions That Join Two or More Strings Together

greatlearning
Learning for Life

Function: CAT

Purpose: To concatenate (join) two or more character strings, leaving leading and/or trailing blanks unchanged. This function accomplishes the same task as the concatenation operator (||).

Syntax: CAT(*string-1, string-2 <,string-n>)* *;where, *string-1, string-2 <,string-n> are the character strings to be* concatenated.

These arguments can also be written as: CAT(OF C1-C5)where C1 to C5 are character variables.

Function: CATX

Purpose: To concatenate (join) two or more character strings, stripping both leading and trailing blanks and inserting one or more separator characters between the strings.

Syntax: CATX(*separator, string-1, string-2 <,string-n>); where, *separator is one or more characters, placed in single or double* quotation marks, to be used as separators between the concatenated strings. *string-1, string-2,string-n are the character strings to be* concatenated.

Note: It is **very important to set the length of the resulting character string, using a** LENGTH statement (or other method), before using any of the concatenation functions. Otherwise, the length of the resulting string will default to 200.

For these examples

A = "Bilbo" (no blanks) ; B = " Frodo" (leading blanks) ; C = "Hobbit " (trailing blanks)

D = " Gandalf " (leading and trailing blanks) ; C1-C5 are five character variables, with the values 'A', 'B', 'C', 'D', and 'E' respectively.

Function	Returns	Function	Returns
CAT(A, B)	"Bilbo Frodo"	**CATX**(" ", A, B)	"Bilbo Frodo"
CAT(B, C, D)	" FrodoHobbit Gandalf "	**CATX(":"**B, C, D)	"Frodo:Hobbit:Gandalf"
CAT("Hello", D)	"Hello Gandalf "	**CATX**("***", "Hello", D)	"Hello***Gandalf"
CAT(OF C1-C5)	"ABCDE"	**CATX**("," ,OF C1-C5)	"A,B,C,D,E"

Financial Functions

greatlearning
Learning for Life

Unknown variable	Excel function	SAS FINANCE function call
Present value	=PV(rate,nper,pmt,fv)	=FINANCE('PV',rate,nper,pmt,fv)
Number of periods	=NPER(rate,pmt,pv,fv)	=FINANCE('NPER',rate,pmt,pv,fv)
Rate of return	=RATE(nper,pmt,pv,fv)	=FINANCE('RATE',nper,pmt,pv,fv)
Periodic payment	=PMT(rate,nper,pv,fv)	=FINANCE('PMT',rate,nper,pv,fv)
Future value	=FV(rate,nper,pmt,pv)	=FINANCE('FV',rate,nper,pmt,pv)

SAS Date Functions

greatlearning
Learning for Life

- Get todays Date

Functions	Description
date();	Get Todays Date
today();	Get Todays Date
"&sysdate"d	Get System Date

- Extract the components of the dates

Function	Syntax	Result
DAY	day=day(*date*);	day of month (1-31)
QTR	quarter=qtr(*date*);	quarter (1-4)
WEEKDAY	wkday=weekday(*date*)	day of week (1-7)
MONTH	month=month(*date*);	month (1-12)
YEAR	yr=year(*date*);	year (4 digits)

- INTCK function: returns the number of time units between two dates. For the time unit, you can choose years, months, weeks, days.

```
newdate = intck('year',birthdate, date());
```

- INTNX function: returns a SAS date that is a specified number of time units away from a specified date.

```
Retirement_Date = intnx('year',birthdate, 65);
```

SAS Date Functions

The YRDIF function can be used in calculating interest for fixed income securities when the third argument, *basis*, is present. YRDIF returns the difference between two dates according to specified day count conventions.

```
data _null_;
startdate=’01jan2013’d;
enddate=‘07feb2020’d;
y30360=yrdif(sdate, edate, '30/360’);
yactact=yrdif(sdate, edate, 'ACT/ACT’);
yact360=yrdif(sdate, edate, 'ACT/360’);
yact365=yrdif(sdate, edate, 'ACT/365’);
put y30360= / yactact= / yact360= / yact365= ;
run;
```

In ACT/ACT basis, both a 365–day year and 366–day year are taken into account.

Generating Random variables

greatlearning
Learning for Life

SAS can generate RV's from a given dist.

Binomial distribution b(n,p)	ranbin(seed,n,p)
Exponential distribution	ranexp(seed) Exp(1)
Standard normal N(0,1)	rannor(seed)
Poisson distribution Poisson(m)	ranpoi(seed,m)
Uniform distribution Unif(0,1)	ranuni(seed)

```
DATA random;
        DO i = 1 TO 100;
        r = RANNOR(0);
        y=exp(r);
        output;
        END;
RUN;
```

Generating Random variables

greatlearning
Learning for Life

Using RAND Function

```
data Rand;
length d $7;
label d="Distribution" x="Result";
call streaminit(12345);
d='EXPO';   x=rand('EXPO');           output;
d='NORMAL'; x=rand('NORMAL');         output;
d='UNIFORM';x=rand('UNIFORM');        output;
d='WEIB';   x=rand('WEIB', 1, 2);     output;
run;
```

Calculating Moving Average using LAG function

- A LAG*n* function stores a value in a queue and returns a value stored previously in that queue. Each occurrence of a LAG*n* function in a program generates its own queue of values.

Obs	s	x	a
1	1	1	1.0
2	3	2	1.5
3	6	3	2.0
4	10	4	2.5
5	15	5	3.0
6	21	6	3.5
7	28	7	4.0
8	36	8	4.5
9	45	9	5.0
10	55	10	5.5

```
data x;
do x=1 to 10;
  output;
  end;
run;
data x;
  do x=1 to 10;
  output;
  end;
run;
%let n=5;
data avg (drop=s); retain s; set x;
s=sum (s, x, -lag&n(x)) ;
a=s / min(_n_, &n); run;
proc print; run;
```

Converting variable types using PUT and INPUT

greatlearning
Learning for Life

PUT and INPUT functions are used to convert numeric to character and viceversa.

- PUT() always creates character variables
- INPUT() can create character or numeric variables based on the informat
- The source format must match the source variable type in PUT()
- The source variable type for INPUT() must always be character variables

Function Call	Raw Type	Raw Value	Returned Type	Returned Value
A PUT(name, $10.);	**char, char format**	'Amitabh'	**char always**	'Amitabh '
B PUT(age, 4.);	num, num format	50	**char always**	' 50'
C PUT(name, $nickname.);	**char, char format**	'Amitabh'	**char always**	'Amit'
D INPUT(agechar, 4.);	**char always**	'50'	num, num informat	50
E INPUT(agechar, $4.);	**char always**	'50'	**char, char informat**	' 50'
F INPUT(cost,comma7.);	**char always**	'500,652'	num, num informat	500652

greatlearning
Learning for Life

Thank You