

Functions by category

Contents

Date and time functions
Mathematical functions
Matrix functions
Programming functions
Random-number functions
Selecting time-span functions
Statistical functions
String functions
Trigonometric functions

Date and time functions

<code>age($e_{d\text{DOB}}$, e_d [, s_{nl}])</code>	the age in integer years on e_d for date of birth $e_{d\text{DOB}}$ with s_{nl} the nonleap-year birthday for 29feb birthdates
<code>age_frac($e_{d\text{DOB}}$, e_d [, s_{nl}])</code>	the age in years, including the fractional part, on e_d for date of birth $e_{d\text{DOB}}$ with s_{nl} the nonleap-year birthday for 29feb birthdates
<code>birthday($e_{d\text{DOB}}$, Y [, s_{nl}])</code>	the e_d date of the birthday in year Y for date of birth $e_{d\text{DOB}}$ with s_{nl} the nonleap-year birthday for 29feb birthdates
<code>bofd("cal", e_d)</code>	the e_b business date corresponding to e_d
<code>Cdhms(e_d, h, m, s)</code>	the e_{tC} datetime (ms. with leap seconds since 01jan1960 00:00:00.000) corresponding to e_d , h , m , s
<code>Chms(h, m, s)</code>	the e_{tC} datetime (ms. with leap seconds since 01jan1960 00:00:00.000) corresponding to h , m , s on 01jan1960
<code>Clock(s_1, s_2 [, Y])</code>	the e_{tC} datetime (ms. with leap seconds since 01jan1960 00:00:00.000) corresponding to s_1 based on s_2 and Y
<code>clock(s_1, s_2 [, Y])</code>	the e_{tc} datetime (ms. since 01jan1960 00:00:00.000) corresponding to s_1 based on s_2 and Y
<code>Clockdiff(e_{tC1}, e_{tC2}, s_u)</code>	the e_{tC} datetime difference, rounded down to an integer, from e_{tC1} to e_{tC2} in s_u units of days, hours, minutes, seconds, or milliseconds
<code>clockdiff(e_{tc1}, e_{tc2}, s_u)</code>	the e_{tc} datetime difference, rounded down to an integer, from e_{tc1} to e_{tc2} in s_u units of days, hours, minutes, seconds, or milliseconds
<code>Clockdiff_frac(e_{tC1}, e_{tC2}, s_u)</code>	the e_{tC} datetime difference, including the fractional part, from e_{tC1} to e_{tC2} in s_u units of days, hours, minutes, seconds, or milliseconds
<code>clockdiff_frac(e_{tc1}, e_{tc2}, s_u)</code>	the e_{tc} datetime difference, including the fractional part, from e_{tc1} to e_{tc2} in s_u units of days, hours, minutes, seconds, or milliseconds
<code>Clockpart(e_{tC}, s_u)</code>	the integer year, month, day, hour, minute, second, or millisecond of e_{tC} with s_u specifying which time part
<code>clockpart(e_{tc}, s_u)</code>	the integer year, month, day, hour, minute, second, or millisecond of e_{tc} with s_u specifying which time part

2 Functions by category

<code>CmDyhms(M, D, Y, h, m, s)</code>	the e_{tC} datetime (ms. with leap seconds since 01jan1960 00:00:00.000) corresponding to M, D, Y, h, m, s
<code>Cofc(e_{tC})</code>	the e_{tC} datetime (ms. with leap seconds since 01jan1960 00:00:00.000) of e_{tc} (ms. without leap seconds since 01jan1960 00:00:00.000)
<code>cofC(e_{tC})</code>	the e_{tc} datetime (ms. without leap seconds since 01jan1960 00:00:00.000) of e_{tC} (ms. with leap seconds since 01jan1960 00:00:00.000)
<code>Cofd(e_d)</code>	the e_{tC} datetime (ms. with leap seconds since 01jan1960 00:00:00.000) of date e_d at time 00:00:00.000
<code>cofd(e_d)</code>	the e_{tc} datetime (ms. since 01jan1960 00:00:00.000) of date e_d at time 00:00:00.000
<code>daily(s_1, s_2 [, Y])</code>	a synonym for <code>date(s_1, s_2 [, Y])</code>
<code>date(s_1, s_2 [, Y])</code>	the e_d date (days since 01jan1960) corresponding to s_1 based on s_2 and Y
<code>datediff(e_{d1}, e_{d2}, s_u [, s_{nl}])</code>	the difference, rounded down to an integer, from e_{d1} to e_{d2} in s_u units of days, months, or years with s_{nl} the nonleap-year anniversary for e_{d1} on 29feb
<code>datediff_frac(e_{d1}, e_{d2}, s_u [, s_{nl}])</code>	the difference, including the fractional part, from e_{d1} to e_{d2} in s_u units of days, months, or years with s_{nl} the nonleap-year anniversary for e_{d1} on 29feb
<code>datepart(e_d, s_u)</code>	the integer year, month, or day of e_d with s_u specifying year, month, or day
<code>day(e_d)</code>	the numeric day of the month corresponding to e_d
<code>daysinmonth(e_d)</code>	the number of days in the month of e_d
<code>dhms(e_d, h, m, s)</code>	the e_{tc} datetime (ms. since 01jan1960 00:00:00.000) corresponding to e_d, h, m, s
<code>dofb($e_b, "cal"$)</code>	the e_d datetime corresponding to e_b
<code>dofC(e_{tC})</code>	the e_d date (days since 01jan1960) of datetime e_{tC} (ms. with leap seconds since 01jan1960 00:00:00.000)
<code>dofc(e_{tC})</code>	the e_d date (days since 01jan1960) of datetime e_{tc} (ms. since 01jan1960 00:00:00.000)
<code>dofh(e_h)</code>	the e_d date (days since 01jan1960) of the start of half-year e_h
<code>dofm(e_m)</code>	the e_d date (days since 01jan1960) of the start of month e_m
<code>dofq(e_q)</code>	the e_d date (days since 01jan1960) of the start of quarter e_q
<code>dofw(e_w)</code>	the e_d date (days since 01jan1960) of the start of week e_w
<code>dofy(e_y)</code>	the e_d date (days since 01jan1960) of 01jan in year e_y
<code>dow(e_d)</code>	the numeric day of the week corresponding to date e_d ; 0 = Sunday, 1 = Monday, ..., 6 = Saturday
<code>day(e_d)</code>	the numeric day of the year corresponding to date e_d
<code>firstdayofmonth(e_d)</code>	the e_d date of the first day of the month of e_d
<code>halfyear(e_d)</code>	the numeric half of the year corresponding to date e_d
<code>halfyearly(s_1, s_2 [, Y])</code>	the e_h half-yearly date (half-years since 1960h1) corresponding to s_1 based on s_2 and Y ; Y specifies <i>topyear</i> ; see <code>date()</code>

<code>hh(e_{tc})</code>	the hour corresponding to datetime e_{tc} (ms. since 01jan1960 00:00:00.000)
<code>hhC(e_{tC})</code>	the hour corresponding to datetime e_{tC} (ms. with leap seconds since 01jan1960 00:00:00.000)
<code>hms(h, m, s)</code>	the e_{tc} datetime (ms. since 01jan1960 00:00:00.000) corresponding to h, m, s on 01jan1960
<code>hofd(e_d)</code>	the e_h half-yearly date (half years since 1960h1) containing date e_d
<code>hours(ms)</code>	$ms/3,600,000$
<code>isleapsecond(e_{tC})</code>	1 if e_{tC} is a leap second; otherwise, 0
<code>isleapyear(Y)</code>	1 if Y is a leap year; otherwise, 0
<code>lastdayofmonth(e_d)</code>	the e_d date of the last day of the month of e_d
<code>mdy(M, D, Y)</code>	the e_d date (days since 01jan1960) corresponding to M, D, Y
<code>mdyhms(M, D, Y, h, m, s)</code>	the e_{tc} datetime (ms. since 01jan1960 00:00:00.000) corresponding to M, D, Y, h, m, s
<code>minutes(ms)</code>	$ms/60,000$
<code>mm(e_{tc})</code>	the minute corresponding to datetime e_{tc} (ms. since 01jan1960 00:00:00.000)
<code>mmC(e_{tC})</code>	the minute corresponding to datetime e_{tC} (ms. with leap seconds since 01jan1960 00:00:00.000)
<code>mofd(e_d)</code>	the e_m monthly date (months since 1960m1) containing date e_d
<code>month(e_d)</code>	the numeric month corresponding to date e_d
<code>monthly(s_1, s_2 [, Y])</code>	the e_m monthly date (months since 1960m1) corresponding to s_1 based on s_2 and Y ; Y specifies <i>topyear</i> ; see <code>date()</code>
<code>msofhours(h)</code>	$h \times 3,600,000$
<code>msofminutes(m)</code>	$m \times 60,000$
<code>msofseconds(s)</code>	$s \times 1,000$
<code>nextbirthday($e_{d\text{DOB}}, e_d$ [, s_{nl}])</code>	the e_d date of the first birthday after e_d for date of birth $e_{d\text{DOB}}$ with s_{nl} the nonleap-year birthday for 29feb birthdates
<code>nextleapyear(Y)</code>	the first leap year after year Y
<code>now()</code>	the current e_{tc} datetime
<code>previousbirthday($e_{d\text{DOB}}, e_d$ [, s_{nl}])</code>	the e_d date of the birthday immediately before e_d for date of birth $e_{d\text{DOB}}$ with s_{nl} the nonleap-year birthday for 29feb birthdates
<code>previousleapyear(Y)</code>	the leap year immediately before year Y
<code>qofd(e_d)</code>	the e_q quarterly date (quarters since 1960q1) containing date e_d
<code>quarter(e_d)</code>	the numeric quarter of the year corresponding to date e_d
<code>quarterly(s_1, s_2 [, Y])</code>	the e_q quarterly date (quarters since 1960q1) corresponding to s_1 based on s_2 and Y ; Y specifies <i>topyear</i> ; see <code>date()</code>
<code>seconds(ms)</code>	$ms/1,000$
<code>ss(e_{tc})</code>	the second corresponding to datetime e_{tc} (ms. since 01jan1960 00:00:00.000)
<code>ssC(e_{tC})</code>	the second corresponding to datetime e_{tC} (ms. with leap seconds since 01jan1960 00:00:00.000)

<code>tC(l)</code>	convenience function to make typing dates and times in expressions easier
<code>tC(l)</code>	convenience function to make typing dates and times in expressions easier
<code>tD(l)</code>	convenience function to make typing dates in expressions easier
<code>tH(l)</code>	convenience function to make typing half-yearly dates in expressions easier
<code>tM(l)</code>	convenience function to make typing monthly dates in expressions easier
<code>today()</code>	today's e_d date
<code>tQ(l)</code>	convenience function to make typing quarterly dates in expressions easier
<code>tW(l)</code>	convenience function to make typing weekly dates in expressions easier
<code>week(e_d)</code>	the numeric week of the year corresponding to date e_d , the %td encoded date (days since 01jan1960)
<code>weekly(s₁, s₂ [, Y])</code>	the e_w weekly date (weeks since 1960w1) corresponding to s_1 based on s_2 and Y ; Y specifies <i>topyear</i> ; see <code>date()</code>
<code>wofd(e_d)</code>	the e_w weekly date (weeks since 1960w1) containing date e_d
<code>year(e_d)</code>	the numeric year corresponding to date e_d
<code>yearly(s₁, s₂ [, Y])</code>	the e_y yearly date (year) corresponding to s_1 based on s_2 and Y ; Y specifies <i>topyear</i> ; see <code>date()</code>
<code>yH(Y, H)</code>	the e_h half-yearly date (half-years since 1960h1) corresponding to year Y , half-year H
<code>yM(Y, M)</code>	the e_m monthly date (months since 1960m1) corresponding to year Y , month M
<code>yofd(e_d)</code>	the e_y yearly date (year) containing date e_d
<code>yQ(Y, Q)</code>	the e_q quarterly date (quarters since 1960q1) corresponding to year Y , quarter Q
<code>yW(Y, W)</code>	the e_w weekly date (weeks since 1960w1) corresponding to year Y , week W

Mathematical functions

<code>abs(x)</code>	the absolute value of x
<code>ceil(x)</code>	the unique integer n such that $n - 1 < x \leq n$; x (not “.”) if x is missing, meaning that <code>ceil(.a) = .a</code>
<code>cloglog(x)</code>	the complementary log-log of x
<code>comb(n, k)</code>	the combinatorial function $n! / \{k!(n - k)!\}$
<code>digamma(x)</code>	the <code>digamma()</code> function, $d \ln \Gamma(x) / dx$
<code>exp(x)</code>	the exponential function e^x
<code>expm1(x)</code>	$e^x - 1$ with higher precision than <code>exp(x) - 1</code> for small values of $ x $
<code>floor(x)</code>	the unique integer n such that $n \leq x < n + 1$; x (not “.”) if x is missing, meaning that <code>floor(.a) = .a</code>

<code>int(x)</code>	the integer obtained by truncating x toward 0 (thus, <code>int(5.2) = 5</code> and <code>int(-5.8) = -5</code>); x (not “.”) if x is missing, meaning that <code>int(.a) = .a</code>
<code>invcloglog(x)</code>	the inverse of the complementary log-log function of x
<code>invlogit(x)</code>	the inverse of the logit function of x
<code>ln(x)</code>	the natural logarithm, $\ln(x)$
<code>ln1m(x)</code>	the natural logarithm of $1 - x$ with higher precision than <code>ln(1 - x)</code> for small values of $ x $
<code>ln1p(x)</code>	the natural logarithm of $1 + x$ with higher precision than <code>ln(1 + x)</code> for small values of $ x $
<code>lnfactorial(n)</code>	the natural log of n factorial = $\ln(n!)$
<code>lngamma(x)</code>	$\ln\{\Gamma(x)\}$
<code>log(x)</code>	a synonym for <code>ln(x)</code>
<code>log10(x)</code>	the base-10 logarithm of x
<code>log1m(x)</code>	a synonym for <code>ln1m(x)</code>
<code>log1p(x)</code>	a synonym for <code>ln1p(x)</code>
<code>logit(x)</code>	the log of the odds ratio of x , $\text{logit}(x) = \ln\{x/(1 - x)\}$
<code>max(x₁, x₂, ..., x_n)</code>	the maximum value of x_1, x_2, \dots, x_n
<code>min(x₁, x₂, ..., x_n)</code>	the minimum value of x_1, x_2, \dots, x_n
<code>mod(x, y)</code>	the modulus of x with respect to y
<code>reldif(x, y)</code>	the “relative” difference $ x - y /(y + 1)$; 0 if both arguments are the same type of extended missing value; <i>missing</i> if only one argument is missing or if the two arguments are two different types of <i>missing</i>
<code>round(x, y)</code> or <code>round(x)</code>	x rounded in units of y or x rounded to the nearest integer if the argument y is omitted; x (not “.”) if x is missing (meaning that <code>round(.a) = .a</code> and that <code>round(.a, y) = .a</code> if y is not missing) and if y is missing, then “.” is returned
<code>sign(x)</code>	the sign of x : -1 if $x < 0$, 0 if $x = 0$, 1 if $x > 0$, or <i>missing</i> if x is missing
<code>sqrt(x)</code>	the square root of x
<code>sum(x)</code>	the running sum of x , treating missing values as zero
<code>trigamma(x)</code>	the second derivative of $\text{lngamma}(x) = d^2 \ln\Gamma(x)/dx^2$
<code>trunc(x)</code>	a synonym for <code>int(x)</code>

Matrix functions

<code>cholesky(M)</code>	the Cholesky decomposition of the matrix: if $R = \text{cholesky}(S)$, then $RR^T = S$
<code>coleqnumb(M, s)</code>	the equation number of M associated with column equation s ; <i>missing</i> if the column equation cannot be found
<code>colnfreeparms(M)</code>	the number of free parameters in columns of M
<code>colnumb(M, s)</code>	the column number of M associated with column name s ; <i>missing</i> if the column cannot be found

<code>colsof(M)</code>	the number of columns of M
<code>corr(M)</code>	the correlation matrix of the variance matrix
<code>det(M)</code>	the determinant of matrix M
<code>diag(M)</code>	the square, diagonal matrix created from the row or column vector
<code>diag0cnt(M)</code>	the number of zeros on the diagonal of M
<code>el(s,i,j)</code>	$s[\text{floor}(i), \text{floor}(j)]$, the i, j element of the matrix named s ; <i>missing</i> if i or j are out of range or if matrix s does not exist
<code>get(systemname)</code>	a copy of Stata internal system matrix <i>systemname</i>
<code>hadamard(M,N)</code>	a matrix whose i, j element is $M[i, j] \cdot N[i, j]$ (if M and N are not the same size, this function reports a conformability error)
<code>I(n)</code>	an $n \times n$ identity matrix if n is an integer; otherwise, a $\text{round}(n) \times \text{round}(n)$ identity matrix
<code>inv(M)</code>	the inverse of the matrix M
<code>invsym(M)</code>	the inverse of M if M is positive definite
<code>issymmetric(M)</code>	1 if the matrix is symmetric; otherwise, 0
<code>J(r,c,z)</code>	the $r \times c$ matrix containing elements z
<code>matmissing(M)</code>	1 if any elements of the matrix are missing; otherwise, 0
<code>matuniform(r,c)</code>	the $r \times c$ matrices containing uniformly distributed pseudorandom numbers on the interval (0, 1)
<code>mreldif(X,Y)</code>	the relative difference of X and Y , where the relative difference is defined as $\max_{i,j} \{ x_{ij} - y_{ij} / (y_{ij} + 1)\}$
<code>nullmat(matname)</code>	use with the row-join (,) and column-join (\) operators
<code>roweqnumb(M,s)</code>	the equation number of M associated with row equation s ; <i>missing</i> if the row equation cannot be found
<code>rowfreeparms(M)</code>	the number of free parameters in rows of M
<code>rownumb(M,s)</code>	the row number of M associated with row name s ; <i>missing</i> if the row cannot be found
<code>rowsof(M)</code>	the number of rows of M
<code>sweep(M,i)</code>	matrix M with i th row/column swept
<code>trace(M)</code>	the trace of matrix M
<code>vec(M)</code>	a column vector formed by listing the elements of M , starting with the first column and proceeding column by column
<code>vecdiag(M)</code>	the row vector containing the diagonal of matrix M

Programming functions

<code>autocode(x,n,x0,x1)</code>	partitions the interval from x_0 to x_1 into n equal-length intervals and returns the upper bound of the interval that contains x or the upper bound of the first or last interval if $x < x_0$ or $x > x_1$, respectively
<code>byteorder()</code>	1 if your computer stores numbers by using a hilo byte order and evaluates to 2 if your computer stores numbers by using a lohi byte order
<code>c(name)</code>	the value of the system or constant result $c(\text{name})$ (see [P] creturn)

<code>_caller()</code>	version of the program or session that invoked the currently running program; see [P] version
<code>chop(x, ϵ)</code>	<code>round(x)</code> if $\text{abs}(x - \text{round}(x)) < \epsilon$; otherwise, <code>x</code> ; or <code>x</code> if <code>x</code> is missing
<code>clip(x, a, b)</code>	<code>x</code> if $a < x < b$, <code>b</code> if $x \geq b$, <code>a</code> if $x \leq a$, or <i>missing</i> if <code>x</code> is missing or if $a > b$; <code>x</code> if <code>x</code> is missing
<code>cond(x, a, b[, c])</code>	<code>a</code> if <code>x</code> is <i>true</i> and nonmissing, <code>b</code> if <code>x</code> is <i>false</i> , and <code>c</code> if <code>x</code> is <i>missing</i> ; <code>a</code> if <code>c</code> is not specified and <code>x</code> evaluates to <i>missing</i>
<code>e(name)</code>	the value of stored result <code>e(name)</code> ; see [U] 18.8 Accessing results calculated by other programs
<code>e(sample)</code>	1 if the observation is in the estimation sample and 0 otherwise
<code>epsdouble()</code>	the machine precision of a double-precision number
<code>epsfloat()</code>	the machine precision of a floating-point number
<code>fileexists(f)</code>	1 if the file specified by <code>f</code> exists; otherwise, 0
<code>fileread(f)</code>	the contents of the file specified by <code>f</code>
<code>filereaderror(s)</code>	0 or positive integer, said value having the interpretation of a return code
<code>filewrite(f, s[, r])</code>	writes the string specified by <code>s</code> to the file specified by <code>f</code> and returns the number of bytes in the resulting file
<code>float(x)</code>	the value of <code>x</code> rounded to <code>float</code> precision
<code>fmtwidth(fmtstr)</code>	the output length of the <code>%fmt</code> contained in <code>fmtstr</code> ; <i>missing</i> if <code>fmtstr</code> does not contain a valid <code>%fmt</code>
<code>frval()</code>	returns values of variables stored in other frames
<code>_frval()</code>	programmer's version of <code>frval()</code>
<code>has_eprop(name)</code>	1 if <code>name</code> appears as a word in <code>e(properties)</code> ; otherwise, 0
<code>inlist(z, a, b, ...)</code>	1 if <code>z</code> is a member of the remaining arguments; otherwise, 0
<code>inrange(z, a, b)</code>	1 if it is known that $a \leq z \leq b$; otherwise, 0
<code>irecode(x, x₁, ..., x_n)</code>	<i>missing</i> if <code>x</code> is missing or x_1, \dots, x_n is not weakly increasing; 0 if $x \leq x_1$; 1 if $x_1 < x \leq x_2$; 2 if $x_2 < x \leq x_3$; ...; <code>n</code> if $x > x_n$
<code>matrix(exp)</code>	restricts name interpretation to scalars and matrices; see <code>scalar()</code>
<code>maxbyte()</code>	the largest value that can be stored in storage type <code>byte</code>
<code>maxdouble()</code>	the largest value that can be stored in storage type <code>double</code>
<code>maxfloat()</code>	the largest value that can be stored in storage type <code>float</code>
<code>maxint()</code>	the largest value that can be stored in storage type <code>int</code>
<code>maxlong()</code>	the largest value that can be stored in storage type <code>long</code>
<code>mi(x₁, x₂, ..., x_n)</code>	a synonym for <code>missing(x₁, x₂, ..., x_n)</code>
<code>minbyte()</code>	the smallest value that can be stored in storage type <code>byte</code>
<code>mindouble()</code>	the smallest value that can be stored in storage type <code>double</code>
<code>minfloat()</code>	the smallest value that can be stored in storage type <code>float</code>
<code>minint()</code>	the smallest value that can be stored in storage type <code>int</code>
<code>minlong()</code>	the smallest value that can be stored in storage type <code>long</code>
<code>missing(x₁, x₂, ..., x_n)</code>	1 if any <code>x_i</code> evaluates to <i>missing</i> ; otherwise, 0

<code>r(name)</code>	the value of the stored result <code>r(name)</code> ; see [U] 18.8 Accessing results calculated by other programs
<code>recode(x, x₁, ..., x_n)</code>	<i>missing</i> if x_1, x_2, \dots, x_n is not weakly increasing; <i>x</i> if <i>x</i> is missing; x_1 if $x \leq x_1$; x_2 if $x \leq x_2, \dots$; otherwise, x_n if $x > x_1, x_2, \dots, x_{n-1}$. $x_i \geq .$ is interpreted as $x_i = +\infty$
<code>replay()</code>	1 if the first nonblank character of local macro '0' is a comma, or if '0' is empty
<code>return(name)</code>	the value of the to-be-stored result <code>r(name)</code> ; see [P] return
<code>s(name)</code>	the value of stored result <code>s(name)</code> ; see [U] 18.8 Accessing results calculated by other programs
<code>scalar(exp)</code>	restricts name interpretation to scalars and matrices
<code>smallestdouble()</code>	the smallest double-precision number greater than zero

Random-number functions

<code>rbeta(a, b)</code>	beta(<i>a, b</i>) random variates, where <i>a</i> and <i>b</i> are the beta distribution shape parameters
<code>rbinomial(n, p)</code>	binomial(<i>n, p</i>) random variates, where <i>n</i> is the number of trials and <i>p</i> is the success probability
<code>rcauchy(a, b)</code>	Cauchy(<i>a, b</i>) random variates, where <i>a</i> is the location parameter and <i>b</i> is the scale parameter
<code>rchi2(df)</code>	χ^2 , with <i>df</i> degrees of freedom, random variates
<code>rexponential(b)</code>	exponential random variates with scale <i>b</i>
<code>rgamma(a, b)</code>	gamma(<i>a, b</i>) random variates, where <i>a</i> is the gamma shape parameter and <i>b</i> is the scale parameter
<code>rhypergeometric(N, K, n)</code>	hypergeometric random variates
<code>rigaussian(m, a)</code>	inverse Gaussian random variates with mean <i>m</i> and shape parameter <i>a</i>
<code>rlaplace(m, b)</code>	Laplace(<i>m, b</i>) random variates with mean <i>m</i> and scale parameter <i>b</i>
<code>rlogistic()</code>	logistic variates with mean 0 and standard deviation $\pi/\sqrt{3}$
<code>rlogistic(s)</code>	logistic variates with mean 0, scale <i>s</i> , and standard deviation $s\pi/\sqrt{3}$
<code>rlogistic(m, s)</code>	logistic variates with mean <i>m</i> , scale <i>s</i> , and standard deviation $s\pi/\sqrt{3}$
<code>rnbinomial(n, p)</code>	negative binomial random variates
<code>rnormal()</code>	standard normal (Gaussian) random variates, that is, variates from a normal distribution with a mean of 0 and a standard deviation of 1
<code>rnormal(m)</code>	normal(<i>m, 1</i>) (Gaussian) random variates, where <i>m</i> is the mean and the standard deviation is 1
<code>rnormal(m, s)</code>	normal(<i>m, s</i>) (Gaussian) random variates, where <i>m</i> is the mean and <i>s</i> is the standard deviation
<code>rpoisson(m)</code>	Poisson(<i>m</i>) random variates, where <i>m</i> is the distribution mean
<code>rt(df)</code>	Student's <i>t</i> random variates, where <i>df</i> is the degrees of freedom
<code>runiform()</code>	uniformly distributed random variates over the interval (0, 1)
<code>runiform(a, b)</code>	uniformly distributed random variates over the interval (<i>a, b</i>)

<code>runiformint(a,b)</code>	uniformly distributed random integer variates on the interval $[a, b]$
<code>rweibull(a,b)</code>	Weibull variates with shape a and scale b
<code>rweibull(a,b,g)</code>	Weibull variates with shape a , scale b , and location g
<code>rweibullph(a,b)</code>	Weibull (proportional hazards) variates with shape a and scale b
<code>rweibullph(a,b,g)</code>	Weibull (proportional hazards) variates with shape a , scale b , and location g

Selecting time-span functions

<code>tin(d1,d2)</code>	true if $d_1 \leq t \leq d_2$, where t is the time variable previously <code>tsset</code>
<code>twithin(d1,d2)</code>	true if $d_1 < t < d_2$, where t is the time variable previously <code>tsset</code>

Statistical functions

<code>betaden(a,b,x)</code>	the probability density of the beta distribution, where a and b are the shape parameters; 0 if $x < 0$ or $x > 1$
<code>binomial(n,k,theta)</code>	the probability of observing <code>floor(k)</code> or fewer successes in <code>floor(n)</code> trials when the probability of a success on one trial is θ ; 0 if $k < 0$; or 1 if $k > n$
<code>binomialp(n,k,p)</code>	the probability of observing <code>floor(k)</code> successes in <code>floor(n)</code> trials when the probability of a success on one trial is p
<code>binomialtail(n,k,theta)</code>	the probability of observing <code>floor(k)</code> or more successes in <code>floor(n)</code> trials when the probability of a success on one trial is θ ; 1 if $k < 0$; or 0 if $k > n$
<code>binormal(h,k,rho)</code>	the joint cumulative distribution $\Phi(h, k, \rho)$ of bivariate normal with correlation ρ
<code>cauchy(a,b,x)</code>	the cumulative Cauchy distribution with location parameter a and scale parameter b
<code>cauchyden(a,b,x)</code>	the probability density of the Cauchy distribution with location parameter a and scale parameter b
<code>cauchytail(a,b,x)</code>	the reverse cumulative (upper tail or survivor) Cauchy distribution with location parameter a and scale parameter b
<code>chi2(df,x)</code>	the cumulative χ^2 distribution with df degrees of freedom; 0 if $x < 0$
<code>chi2den(df,x)</code>	the probability density of the χ^2 distribution with df degrees of freedom; 0 if $x < 0$
<code>chi2tail(df,x)</code>	the reverse cumulative (upper tail or survivor) χ^2 distribution with df degrees of freedom; 1 if $x < 0$
<code>dgammapda(a,x)</code>	$\frac{\partial P(a,x)}{\partial a}$, where $P(a,x) = \text{gammap}(a,x)$; 0 if $x < 0$
<code>dgammapdada(a,x)</code>	$\frac{\partial^2 P(a,x)}{\partial a^2}$, where $P(a,x) = \text{gammap}(a,x)$; 0 if $x < 0$
<code>dgammapdadx(a,x)</code>	$\frac{\partial^2 P(a,x)}{\partial a \partial x}$, where $P(a,x) = \text{gammap}(a,x)$; 0 if $x < 0$
<code>dgammapdx(a,x)</code>	$\frac{\partial P(a,x)}{\partial x}$, where $P(a,x) = \text{gammap}(a,x)$; 0 if $x < 0$
<code>dgammapdxdx(a,x)</code>	$\frac{\partial^2 P(a,x)}{\partial x^2}$, where $P(a,x) = \text{gammap}(a,x)$; 0 if $x < 0$

<code>dunnettprob(k, df, x)</code>	the cumulative multiple range distribution that is used in Dunnett's multiple-comparison method with k ranges and df degrees of freedom; 0 if $x < 0$
<code>exponential(b, x)</code>	the cumulative exponential distribution with scale b
<code>exponentialden(b, x)</code>	the probability density function of the exponential distribution with scale b
<code>exponentialtail(b, x)</code>	the reverse cumulative exponential distribution with scale b
<code>F(df1, df2, f)</code>	the cumulative F distribution with df_1 numerator and df_2 denominator degrees of freedom: $F(df_1, df_2, f) = \int_0^f Fden(df_1, df_2, t) dt$; 0 if $f < 0$
<code>Fden(df1, df2, f)</code>	the probability density function of the F distribution with df_1 numerator and df_2 denominator degrees of freedom; 0 if $f < 0$
<code>Ftail(df1, df2, f)</code>	the reverse cumulative (upper tail or survivor) F distribution with df_1 numerator and df_2 denominator degrees of freedom; 1 if $f < 0$
<code>gammaden(a, b, g, x)</code>	the probability density function of the gamma distribution; 0 if $x < g$
<code>gammap(a, x)</code>	the cumulative gamma distribution with shape parameter a ; 0 if $x < 0$
<code>gammatail(a, x)</code>	the reverse cumulative (upper tail or survivor) gamma distribution with shape parameter a ; 1 if $x < 0$
<code>hypergeometric(N, K, n, k)</code>	the cumulative probability of the hypergeometric distribution
<code>hypergeometricp(N, K, n, k)</code>	the hypergeometric probability of k successes out of a sample of size n , from a population of size N containing K elements that have the attribute of interest
<code>ibeta(a, b, x)</code>	the cumulative beta distribution with shape parameters a and b ; 0 if $x < 0$; or 1 if $x > 1$
<code>ibetatail(a, b, x)</code>	the reverse cumulative (upper tail or survivor) beta distribution with shape parameters a and b ; 1 if $x < 0$; or 0 if $x > 1$
<code>igaussian(m, a, x)</code>	the cumulative inverse Gaussian distribution with mean m and shape parameter a ; 0 if $x \leq 0$
<code>igaussianden(m, a, x)</code>	the probability density of the inverse Gaussian distribution with mean m and shape parameter a ; 0 if $x \leq 0$
<code>igaussiantail(m, a, x)</code>	the reverse cumulative (upper tail or survivor) inverse Gaussian distribution with mean m and shape parameter a ; 1 if $x \leq 0$
<code>invbinomial(n, k, p)</code>	the inverse of the cumulative binomial; that is, θ (θ = probability of success on one trial) such that the probability of observing <code>floor(k)</code> or fewer successes in <code>floor(n)</code> trials is p
<code>invbinomialtail(n, k, p)</code>	the inverse of the right cumulative binomial; that is, θ (θ = probability of success on one trial) such that the probability of observing <code>floor(k)</code> or more successes in <code>floor(n)</code> trials is p
<code>invcauchy(a, b, p)</code>	the inverse of <code>cauchy()</code> : if <code>cauchy(a, b, x) = p</code> , then <code>invcauchy(a, b, p) = x</code>
<code>invcauchytail(a, b, p)</code>	the inverse of <code>cauchytail()</code> : if <code>cauchytail(a, b, x) = p</code> , then <code>invcauchytail(a, b, p) = x</code>
<code>invchi2(df, p)</code>	the inverse of <code>chi2()</code> : if <code>chi2(df, x) = p</code> , then <code>invchi2(df, p) = x</code>
<code>invchi2tail(df, p)</code>	the inverse of <code>chi2tail()</code> : if <code>chi2tail(df, x) = p</code> , then <code>invchi2tail(df, p) = x</code>

<code>invdunnettprob(k, df, p)</code>	the inverse cumulative multiple range distribution that is used in Dunnett's multiple-comparison method with k ranges and df degrees of freedom
<code>invexponential(b, p)</code>	the inverse cumulative exponential distribution with scale b : if <code>exponential(b, x) = p</code> , then <code>invexponential(b, p) = x</code>
<code>invexponentialtail(b, p)</code>	the inverse reverse cumulative exponential distribution with scale b : if <code>exponentialtail(b, x) = p</code> , then <code>invexponentialtail(b, p) = x</code>
<code>invF(df_1, df_2, p)</code>	the inverse cumulative F distribution: if <code>F(df_1, df_2, f) = p</code> , then <code>invF(df_1, df_2, p) = f</code>
<code>invFtail(df_1, df_2, p)</code>	the inverse reverse cumulative (upper tail or survivor) F distribution: if <code>Ftail(df_1, df_2, f) = p</code> , then <code>invFtail(df_1, df_2, p) = f</code>
<code>invgammap(a, p)</code>	the inverse cumulative gamma distribution: if <code>gammap(a, x) = p</code> , then <code>invgammap(a, p) = x</code>
<code>invgammaptail(a, p)</code>	the inverse reverse cumulative (upper tail or survivor) gamma distribution: if <code>gammaptail(a, x) = p</code> , then <code>invgammaptail(a, p) = x</code>
<code>invibeta(a, b, p)</code>	the inverse cumulative beta distribution: if <code>ibeta(a, b, x) = p</code> , then <code>invibeta(a, b, p) = x</code>
<code>invibetatail(a, b, p)</code>	the inverse reverse cumulative (upper tail or survivor) beta distribution: if <code>ibetatail(a, b, x) = p</code> , then <code>invibetatail(a, b, p) = x</code>
<code>invigaussian(m, a, p)</code>	the inverse of <code>igaussian()</code> : if <code>igaussian(m, a, x) = p</code> , then <code>invigaussian(m, a, p) = x</code>
<code>invigaussiantail(m, a, p)</code>	the inverse of <code>igaussiantail()</code> : if <code>igaussiantail(m, a, x) = p</code> , then <code>invigaussiantail(m, a, p) = x</code>
<code>invlaplace(m, b, p)</code>	the inverse of <code>laplace()</code> : if <code>laplace(m, b, x) = p</code> , then <code>invlaplace(m, b, p) = x</code>
<code>invlaplacetail(m, b, p)</code>	the inverse of <code>laplacetail()</code> : if <code>laplacetail(m, b, x) = p</code> , then <code>invlaplacetail(m, b, p) = x</code>
<code>invlogistic(p)</code>	the inverse cumulative logistic distribution: if <code>logistic(x) = p</code> , then <code>invlogistic(p) = x</code>
<code>invlogistic(s, p)</code>	the inverse cumulative logistic distribution: if <code>logistic(s, x) = p</code> , then <code>invlogistic(s, p) = x</code>
<code>invlogistic(m, s, p)</code>	the inverse cumulative logistic distribution: if <code>logistic(m, s, x) = p</code> , then <code>invlogistic(m, s, p) = x</code>
<code>invlogistictail(p)</code>	the inverse reverse cumulative logistic distribution: if <code>logistictail(x) = p</code> , then <code>invlogistictail(p) = x</code>
<code>invlogistictail(s, p)</code>	the inverse reverse cumulative logistic distribution: if <code>logistictail(s, x) = p</code> , then <code>invlogistictail(s, p) = x</code>
<code>invlogistictail(m, s, p)</code>	the inverse reverse cumulative logistic distribution: if <code>logistictail(m, s, x) = p</code> , then <code>invlogistictail(m, s, p) = x</code>
<code>invnbinomial(n, k, q)</code>	the value of the negative binomial parameter, p , such that $q = \text{nbinomial}(n, k, p)$
<code>invnbinomialtail(n, k, q)</code>	the value of the negative binomial parameter, p , such that $q = \text{nbinomialtail}(n, k, p)$

<code>invnchi2(df, np, p)</code>	the inverse cumulative noncentral χ^2 distribution: if $\text{nchi2}(df, np, x) = p$, then $\text{invnchi2}(df, np, p) = x$
<code>invnchi2tail(df, np, p)</code>	the inverse reverse cumulative (upper tail or survivor) noncentral χ^2 distribution: if $\text{nchi2tail}(df, np, x) = p$, then $\text{invnchi2tail}(df, np, p) = x$
<code>invnF(df1, df2, np, p)</code>	the inverse cumulative noncentral F distribution: if $\text{nF}(df_1, df_2, np, f) = p$, then $\text{invnF}(df_1, df_2, np, p) = f$
<code>invnFtail(df1, df2, np, p)</code>	the inverse reverse cumulative (upper tail or survivor) noncentral F distribution: if $\text{nFtail}(df_1, df_2, np, f) = p$, then $\text{invnFtail}(df_1, df_2, np, p) = f$
<code>invnibeta(a, b, np, p)</code>	the inverse cumulative noncentral beta distribution: if $\text{nibeta}(a, b, np, x) = p$, then $\text{invnibeta}(a, b, np, p) = x$
<code>invnormal(p)</code>	the inverse cumulative standard normal distribution: if $\text{normal}(z) = p$, then $\text{invnormal}(p) = z$
<code>invnt(df, np, p)</code>	the inverse cumulative noncentral Student's t distribution: if $\text{nt}(df, np, t) = p$, then $\text{invnt}(df, np, p) = t$
<code>invnttail(df, np, p)</code>	the inverse reverse cumulative (upper tail or survivor) noncentral Student's t distribution: if $\text{nttail}(df, np, t) = p$, then $\text{invnttail}(df, np, p) = t$
<code>invpoisson(k, p)</code>	the Poisson mean such that the cumulative Poisson distribution evaluated at k is p : if $\text{poisson}(m, k) = p$, then $\text{invpoisson}(k, p) = m$
<code>invpoisontail(k, q)</code>	the Poisson mean such that the reverse cumulative Poisson distribution evaluated at k is q : if $\text{poisontail}(m, k) = q$, then $\text{invpoisontail}(k, q) = m$
<code>invt(df, p)</code>	the inverse cumulative Student's t distribution: if $\text{t}(df, t) = p$, then $\text{invt}(df, p) = t$
<code>invttail(df, p)</code>	the inverse reverse cumulative (upper tail or survivor) Student's t distribution: if $\text{ttail}(df, t) = p$, then $\text{invttail}(df, p) = t$
<code>invtukeyprob(k, df, p)</code>	the inverse cumulative Tukey's Studentized range distribution with k ranges and df degrees of freedom
<code>invweibull(a, b, p)</code>	the inverse cumulative Weibull distribution with shape a and scale b : if $\text{weibull}(a, b, x) = p$, then $\text{invweibull}(a, b, p) = x$
<code>invweibull(a, b, g, p)</code>	the inverse cumulative Weibull distribution with shape a , scale b , and location g : if $\text{weibull}(a, b, g, x) = p$, then $\text{invweibull}(a, b, g, p) = x$
<code>invweibullph(a, b, p)</code>	the inverse cumulative Weibull (proportional hazards) distribution with shape a and scale b : if $\text{weibullph}(a, b, x) = p$, then $\text{invweibullph}(a, b, p) = x$
<code>invweibullph(a, b, g, p)</code>	the inverse cumulative Weibull (proportional hazards) distribution with shape a , scale b , and location g : if $\text{weibullph}(a, b, g, x) = p$, then $\text{invweibullph}(a, b, g, p) = x$
<code>invweibullphtail(a, b, p)</code>	the inverse reverse cumulative Weibull (proportional hazards) distribution with shape a and scale b : if $\text{weibullphtail}(a, b, x) = p$, then $\text{invweibullphtail}(a, b, p) = x$

<code>invweibullphtail(a, b, g, p)</code>	the inverse reverse cumulative Weibull (proportional hazards) distribution with shape a , scale b , and location g : if <code>weibullphtail(a, b, g, x) = p</code> , then <code>invweibullphtail(a, b, g, p) = x</code>
<code>invweibulltail(a, b, p)</code>	the inverse reverse cumulative Weibull distribution with shape a and scale b : if <code>weibulltail(a, b, x) = p</code> , then <code>invweibulltail(a, b, p) = x</code>
<code>invweibulltail(a, b, g, p)</code>	the inverse reverse cumulative Weibull distribution with shape a , scale b , and location g : if <code>weibulltail(a, b, g, x) = p</code> , then <code>invweibulltail(a, b, g, p) = x</code>
<code>laplace(m, b, x)</code>	the cumulative Laplace distribution with mean m and scale parameter b
<code>laplaceden(m, b, x)</code>	the probability density of the Laplace distribution with mean m and scale parameter b
<code>laplacetail(m, b, x)</code>	the reverse cumulative (upper tail or survivor) Laplace distribution with mean m and scale parameter b
<code>lncauchyden(a, b, x)</code>	the natural logarithm of the density of the Cauchy distribution with location parameter a and scale parameter b
<code>lnigammaden(a, b, x)</code>	the natural logarithm of the inverse gamma density, where a is the shape parameter and b is the scale parameter
<code>lnigaussianden(m, a, x)</code>	the natural logarithm of the inverse Gaussian density with mean m and shape parameter a
<code>lniwishartden(df, V, X)</code>	the natural logarithm of the density of the inverse Wishart distribution; missing if $df \leq n - 1$
<code>lnlaplaceden(m, b, x)</code>	the natural logarithm of the density of the Laplace distribution with mean m and scale parameter b
<code>lnmvnormalden(M, V, X)</code>	the natural logarithm of the multivariate normal density
<code>lnnormal(z)</code>	the natural logarithm of the cumulative standard normal distribution
<code>lnnormalden(z)</code>	the natural logarithm of the standard normal density, $N(0, 1)$
<code>lnnormalden(x, σ)</code>	the natural logarithm of the normal density with mean 0 and standard deviation σ
<code>lnnormalden(x, μ, σ)</code>	the natural logarithm of the normal density with mean μ and standard deviation σ , $N(\mu, \sigma^2)$
<code>lnwishartden(df, V, X)</code>	the natural logarithm of the density of the Wishart distribution; missing if $df \leq n - 1$
<code>logistic(x)</code>	the cumulative logistic distribution with mean 0 and standard deviation $\pi/\sqrt{3}$
<code>logistic(s, x)</code>	the cumulative logistic distribution with mean 0, scale s , and standard deviation $s\pi/\sqrt{3}$
<code>logistic(m, s, x)</code>	the cumulative logistic distribution with mean m , scale s , and standard deviation $s\pi/\sqrt{3}$
<code>logisticden(x)</code>	the density of the logistic distribution with mean 0 and standard deviation $\pi/\sqrt{3}$
<code>logisticden(s, x)</code>	the density of the logistic distribution with mean 0, scale s , and standard deviation $s\pi/\sqrt{3}$
<code>logisticden(m, s, x)</code>	the density of the logistic distribution with mean m , scale s , and standard deviation $s\pi/\sqrt{3}$

<code>logistictail(x)</code>	the reverse cumulative logistic distribution with mean 0 and standard deviation $\pi/\sqrt{3}$
<code>logistictail(s, x)</code>	the reverse cumulative logistic distribution with mean 0, scale s , and standard deviation $s\pi/\sqrt{3}$
<code>logistictail(m, s, x)</code>	the reverse cumulative logistic distribution with mean m , scale s , and standard deviation $s\pi/\sqrt{3}$
<code>nbetaden(a, b, np, x)</code>	the probability density function of the noncentral beta distribution; 0 if $x < 0$ or $x > 1$
<code>nbinomial(n, k, p)</code>	the cumulative probability of the negative binomial distribution
<code>nbinomialp(n, k, p)</code>	the negative binomial probability
<code>nbinomialtail(n, k, p)</code>	the reverse cumulative probability of the negative binomial distribution
<code>nchi2(df, np, x)</code>	the cumulative noncentral χ^2 distribution; 0 if $x < 0$
<code>nchi2den(df, np, x)</code>	the probability density of the noncentral χ^2 distribution; 0 if $x < 0$
<code>nchi2tail(df, np, x)</code>	the reverse cumulative (upper tail or survivor) noncentral χ^2 distribution; 1 if $x < 0$
<code>nF(df_1, df_2, np, f)</code>	the cumulative noncentral F distribution with df_1 numerator and df_2 denominator degrees of freedom and noncentrality parameter np ; 0 if $f < 0$
<code>nFden(df_1, df_2, np, f)</code>	the probability density function of the noncentral F distribution with df_1 numerator and df_2 denominator degrees of freedom and noncentrality parameter np ; 0 if $f < 0$
<code>nFtail(df_1, df_2, np, f)</code>	the reverse cumulative (upper tail or survivor) noncentral F distribution with df_1 numerator and df_2 denominator degrees of freedom and noncentrality parameter np ; 1 if $f < 0$
<code>nibeta(a, b, np, x)</code>	the cumulative noncentral beta distribution; 0 if $x < 0$; or 1 if $x > 1$
<code>normal(z)</code>	the cumulative standard normal distribution
<code>normalden(z)</code>	the standard normal density, $N(0, 1)$
<code>normalden(x, σ)</code>	the normal density with mean 0 and standard deviation σ
<code>normalden(x, μ, σ)</code>	the normal density with mean μ and standard deviation σ , $N(\mu, \sigma^2)$
<code>npnchi2(df, x, p)</code>	the noncentrality parameter, np , for noncentral χ^2 : if <code>nchi2(df, np, x) = p</code> , then <code>npnchi2(df, x, p) = np</code>
<code>npnF(df_1, df_2, f, p)</code>	the noncentrality parameter, np , for the noncentral F : if <code>nF(df_1, df_2, np, f) = p</code> , then <code>npnF(df_1, df_2, f, p) = np</code>
<code>npnt(df, t, p)</code>	the noncentrality parameter, np , for the noncentral Student's t distribution: if <code>nt(df, np, t) = p</code> , then <code>npnt(df, t, p) = np</code>
<code>nt(df, np, t)</code>	the cumulative noncentral Student's t distribution with df degrees of freedom and noncentrality parameter np
<code>ntden(df, np, t)</code>	the probability density function of the noncentral Student's t distribution with df degrees of freedom and noncentrality parameter np
<code>nttail(df, np, t)</code>	the reverse cumulative (upper tail or survivor) noncentral Student's t distribution with df degrees of freedom and noncentrality parameter np
<code>poisson(m, k)</code>	the probability of observing <code>floor(k)</code> or fewer outcomes that are distributed as Poisson with mean m

<code>poissonp(m,k)</code>	the probability of observing <code>floor(k)</code> outcomes that are distributed as Poisson with mean m
<code>poisontail(m,k)</code>	the probability of observing <code>floor(k)</code> or more outcomes that are distributed as Poisson with mean m
<code>t(df,t)</code>	the cumulative Student's t distribution with df degrees of freedom
<code>tden(df,t)</code>	the probability density function of Student's t distribution
<code>ttail(df,t)</code>	the reverse cumulative (upper tail or survivor) Student's t distribution; the probability $T > t$
<code>tukeyprob(k,df,x)</code>	the cumulative Tukey's Studentized range distribution with k ranges and df degrees of freedom; 0 if $x < 0$
<code>weibull(a,b,x)</code>	the cumulative Weibull distribution with shape a and scale b
<code>weibull(a,b,g,x)</code>	the cumulative Weibull distribution with shape a , scale b , and location g
<code>weibullden(a,b,x)</code>	the probability density function of the Weibull distribution with shape a and scale b
<code>weibullden(a,b,g,x)</code>	the probability density function of the Weibull distribution with shape a , scale b , and location g
<code>weibullph(a,b,x)</code>	the cumulative Weibull (proportional hazards) distribution with shape a and scale b
<code>weibullph(a,b,g,x)</code>	the cumulative Weibull (proportional hazards) distribution with shape a , scale b , and location g
<code>weibullphden(a,b,x)</code>	the probability density function of the Weibull (proportional hazards) distribution with shape a and scale b
<code>weibullphden(a,b,g,x)</code>	the probability density function of the Weibull (proportional hazards) distribution with shape a , scale b , and location g
<code>weibullphtail(a,b,x)</code>	the reverse cumulative Weibull (proportional hazards) distribution with shape a and scale b
<code>weibullphtail(a,b,g,x)</code>	the reverse cumulative Weibull (proportional hazards) distribution with shape a , scale b , and location g
<code>weibulltail(a,b,x)</code>	the reverse cumulative Weibull distribution with shape a and scale b
<code>weibulltail(a,b,g,x)</code>	the reverse cumulative Weibull distribution with shape a , scale b , and location g

String functions

<code>abbrev(s,n)</code>	name s , abbreviated to a length of n
<code>char(n)</code>	the character corresponding to ASCII or extended ASCII code n ; "" if n is not in the domain
<code>collatorlocale(loc,type)</code>	the most closely related locale supported by ICU from loc if $type$ is 1; the actual locale where the collation data comes from if $type$ is 2
<code>collatorversion(loc)</code>	the version string of a collator based on locale loc
<code>indexnot(s₁,s₂)</code>	the position in ASCII string s_1 of the first character of s_1 not found in ASCII string s_2 , or 0 if all characters of s_1 are found in s_2
<code>plural(n,s)</code>	the plural of s if $n \neq \pm 1$

<code>plural(n, s_1, s_2)</code>	the plural of s_1 , as modified by or replaced with s_2 , if $n \neq \pm 1$
<code>real(s)</code>	s converted to numeric or <i>missing</i>
<code>regexm(s, re)</code>	performs a match of a regular expression and evaluates to 1 if regular expression re is satisfied by the ASCII string s ; otherwise, 0
<code>regexpr(s_1, re, s_2)</code>	replaces the first substring within ASCII string s_1 that matches re with ASCII string s_2 and returns the resulting string
<code>regexs(n)</code>	subexpression n from a previous <code>regexm()</code> match, where $0 \leq n < 10$
<code>soundex(s)</code>	the soundex code for a string, s
<code>soundex_nara(s)</code>	the U.S. Census soundex code for a string, s
<code>strcat(s_1, s_2)</code>	there is no <code>strcat()</code> function; instead the addition operator is used to concatenate strings
<code>strdup(s_1, n)</code>	there is no <code>strdup()</code> function; instead the multiplication operator is used to create multiple copies of strings
<code>string(n)</code>	a synonym for <code>stroofreal(n)</code>
<code>string(n, s)</code>	a synonym for <code>stroofreal(n, s)</code>
<code>stritrim(s)</code>	s with multiple, consecutive internal blanks (ASCII space character <code>char(32)</code>) collapsed to one blank
<code>strlen(s)</code>	the number of characters in ASCII s or length in bytes
<code>strlower(s)</code>	lowercase ASCII characters in string s
<code>strltrim(s)</code>	s without leading blanks (ASCII space character <code>char(32)</code>)
<code>strmatch(s_1, s_2)</code>	1 if s_1 matches the pattern s_2 ; otherwise, 0
<code>stroofreal(n)</code>	n converted to a string
<code>stroofreal(n, s)</code>	n converted to a string using the specified display format
<code>strpos(s_1, s_2)</code>	the position in s_1 at which s_2 is first found, 0 if s_2 does not occur, and 1 if s_2 is empty
<code>strproper(s)</code>	a string with the first ASCII letter and any other letters immediately following characters that are not letters capitalized; all other ASCII letters converted to lowercase
<code>strreverse(s)</code>	reverses the ASCII string s
<code>strrpos(s_1, s_2)</code>	the position in s_1 at which s_2 is last found, 0 if s_2 does not occur, and 1 if s_2 is empty
<code>strrtrim(s)</code>	s without trailing blanks (ASCII space character <code>char(32)</code>)
<code>strtoname($s[, p]$)</code>	s translated into a Stata 13 compatible name
<code>strtrim(s)</code>	s without leading and trailing blanks (ASCII space character <code>char(32)</code>); equivalent to <code>strltrim(strrtrim(s))</code>
<code>strupper(s)</code>	uppercase ASCII characters in string s
<code>subinstr(s_1, s_2, s_3, n)</code>	s_1 , where the first n occurrences in s_1 of s_2 have been replaced with s_3
<code>subinword(s_1, s_2, s_3, n)</code>	s_1 , where the first n occurrences in s_1 of s_2 as a word have been replaced with s_3
<code>substr(s, n_1, n_2)</code>	the substring of s , starting at n_1 , for a length of n_2
<code>tobytes($s[, n]$)</code>	escaped decimal or hex digit strings of up to 200 bytes of s
<code>uchar(n)</code>	the Unicode character corresponding to Unicode code point n or an empty string if n is beyond the Unicode code-point range

<code>udstrlen(<i>s</i>)</code>	the number of display columns needed to display the Unicode string <i>s</i> in the Stata Results window
<code>udsubstr(<i>s</i>,<i>n</i>₁,<i>n</i>₂)</code>	the Unicode substring of <i>s</i> , starting at character <i>n</i> ₁ , for <i>n</i> ₂ display columns
<code>uisdigit(<i>s</i>)</code>	1 if the first Unicode character in <i>s</i> is a Unicode decimal digit; otherwise, 0
<code>uisletter(<i>s</i>)</code>	1 if the first Unicode character in <i>s</i> is a Unicode letter; otherwise, 0
<code>ustrcompare(<i>s</i>₁,<i>s</i>₂[,<i>loc</i>])</code>	compares two Unicode strings
<code>ustrcompareex(<i>s</i>₁,<i>s</i>₂,<i>loc</i>,<i>st</i>,<i>case</i>,<i>cslv</i>,<i>norm</i>,<i>num</i>,<i>alt</i>,<i>fr</i>)</code>	compares two Unicode strings
<code>ustrfix(<i>s</i>[,<i>rep</i>])</code>	replaces each invalid UTF-8 sequence with a Unicode character
<code>ustrfrom(<i>s</i>,<i>enc</i>,<i>mode</i>)</code>	converts the string <i>s</i> in encoding <i>enc</i> to a UTF-8 encoded Unicode string
<code>ustrinvalidcnt(<i>s</i>)</code>	the number of invalid UTF-8 sequences in <i>s</i>
<code>ustrleft(<i>s</i>,<i>n</i>)</code>	the first <i>n</i> Unicode characters of the Unicode string <i>s</i>
<code>ustrlen(<i>s</i>)</code>	the number of characters in the Unicode string <i>s</i>
<code>ustrlower(<i>s</i>[,<i>loc</i>])</code>	lowercase all characters of Unicode string <i>s</i> under the given locale <i>loc</i>
<code>ustrltrim(<i>s</i>)</code>	removes the leading Unicode whitespace characters and blanks from the Unicode string <i>s</i>
<code>ustrnormalize(<i>s</i>,<i>norm</i>)</code>	normalizes Unicode string <i>s</i> to one of the five normalization forms specified by <i>norm</i>
<code>ustrpos(<i>s</i>₁,<i>s</i>₂[,<i>n</i>])</code>	the position in <i>s</i> ₁ at which <i>s</i> ₂ is first found; otherwise, 0
<code>ustrregexm(<i>s</i>,<i>re</i>[,<i>noc</i>])</code>	performs a match of a regular expression and evaluates to 1 if regular expression <i>re</i> is satisfied by the Unicode string <i>s</i> ; otherwise, 0
<code>ustrregexra(<i>s</i>₁,<i>re</i>,<i>s</i>₂[,<i>noc</i>])</code>	replaces all substrings within the Unicode string <i>s</i> ₁ that match <i>re</i> with <i>s</i> ₂ and returns the resulting string
<code>ustrregexrf(<i>s</i>₁,<i>re</i>,<i>s</i>₂[,<i>noc</i>])</code>	replaces the first substring within the Unicode string <i>s</i> ₁ that matches <i>re</i> with <i>s</i> ₂ and returns the resulting string
<code>ustrregexs(<i>n</i>)</code>	subexpression <i>n</i> from a previous <code>ustrregexm()</code> match
<code>ustrreverse(<i>s</i>)</code>	reverses the Unicode string <i>s</i>
<code>ustrright(<i>s</i>,<i>n</i>)</code>	the last <i>n</i> Unicode characters of the Unicode string <i>s</i>
<code>ustrrpos(<i>s</i>₁,<i>s</i>₂[,<i>n</i>])</code>	the position in <i>s</i> ₁ at which <i>s</i> ₂ is last found; otherwise, 0
<code>ustrrtrim(<i>s</i>)</code>	remove trailing Unicode whitespace characters and blanks from the Unicode string <i>s</i>
<code>ustrsortkey(<i>s</i>[,<i>loc</i>])</code>	generates a null-terminated byte array that can be used by the <code>sort</code> command to produce the same order as <code>ustrcompare()</code>
<code>ustrsortkeyex(<i>s</i>,<i>loc</i>,<i>st</i>,<i>case</i>,<i>cslv</i>,<i>norm</i>,<i>num</i>,<i>alt</i>,<i>fr</i>)</code>	generates a null-terminated byte array that can be used by the <code>sort</code> command to produce the same order as <code>ustrcompare()</code>
<code>ustrtitle(<i>s</i>[,<i>loc</i>])</code>	a string with the first characters of Unicode words titlecased and other characters lowercased
<code>ustrto(<i>s</i>,<i>enc</i>,<i>mode</i>)</code>	converts the Unicode string <i>s</i> in UTF-8 encoding to a string in encoding <i>enc</i>
<code>ustrtohex(<i>s</i>[,<i>n</i>])</code>	escaped hex digit string of <i>s</i> up to 200 Unicode characters

<code>ustrtoname(<i>s</i> [, <i>p</i>])</code>	string <i>s</i> translated into a Stata name
<code>ustrtrim(<i>s</i>)</code>	removes leading and trailing Unicode whitespace characters and blanks from the Unicode string <i>s</i>
<code>ustrunescape(<i>s</i>)</code>	the Unicode string corresponding to the escaped sequences of <i>s</i>
<code>ustrupper(<i>s</i> [, <i>loc</i>])</code>	uppercase all characters in string <i>s</i> under the given locale <i>loc</i>
<code>ustrword(<i>s</i>, <i>n</i> [, <i>loc</i>])</code>	the <i>n</i> th Unicode word in the Unicode string <i>s</i>
<code>ustrwordcount(<i>s</i> [, <i>loc</i>])</code>	the number of nonempty Unicode words in the Unicode string <i>s</i>
<code>usubinstr(<i>s</i>₁, <i>s</i>₂, <i>s</i>₃, <i>n</i>)</code>	replaces the first <i>n</i> occurrences of the Unicode string <i>s</i> ₂ with the Unicode string <i>s</i> ₃ in <i>s</i> ₁
<code>usubstr(<i>s</i>, <i>n</i>₁, <i>n</i>₂)</code>	the Unicode substring of <i>s</i> , starting at <i>n</i> ₁ , for a length of <i>n</i> ₂
<code>word(<i>s</i>, <i>n</i>)</code>	the <i>n</i> th word in <i>s</i> ; missing ("") if <i>n</i> is missing
<code>wordbreaklocale(<i>loc</i>, <i>type</i>)</code>	the most closely related locale supported by ICU from <i>loc</i> if <i>type</i> is 1, the actual locale where the word-boundary analysis data come from if <i>type</i> is 2; or an empty string is returned for any other <i>type</i>
<code>wordcount(<i>s</i>)</code>	the number of words in <i>s</i>

Trigonometric functions

<code>acos(<i>x</i>)</code>	the radian value of the arccosine of <i>x</i>
<code>acosh(<i>x</i>)</code>	the inverse hyperbolic cosine of <i>x</i>
<code>asin(<i>x</i>)</code>	the radian value of the arcsine of <i>x</i>
<code>asinh(<i>x</i>)</code>	the inverse hyperbolic sine of <i>x</i>
<code>atan(<i>x</i>)</code>	the radian value of the arctangent of <i>x</i>
<code>atan2(<i>y</i>, <i>x</i>)</code>	the radian value of the arctangent of <i>y/x</i> , where the signs of the parameters <i>y</i> and <i>x</i> are used to determine the quadrant of the answer
<code>atanh(<i>x</i>)</code>	the inverse hyperbolic tangent of <i>x</i>
<code>cos(<i>x</i>)</code>	the cosine of <i>x</i> , where <i>x</i> is in radians
<code>cosh(<i>x</i>)</code>	the hyperbolic cosine of <i>x</i>
<code>sin(<i>x</i>)</code>	the sine of <i>x</i> , where <i>x</i> is in radians
<code>sinh(<i>x</i>)</code>	the hyperbolic sine of <i>x</i>
<code>tan(<i>x</i>)</code>	the tangent of <i>x</i> , where <i>x</i> is in radians
<code>tanh(<i>x</i>)</code>	the hyperbolic tangent of <i>x</i>

Also see

[FN] **Functions by name**

[D] **egen** — Extensions to generate

[D] **generate** — Create or change contents of variable

[M-4] **Intro** — Categorical guide to Mata functions

[U] **13.3 Functions**