

Functions by name

<code>abbrev(<i>s</i>,<i>n</i>)</code>	name <i>s</i> , abbreviated to a length of <i>n</i>
<code>abs(<i>x</i>)</code>	the absolute value of <i>x</i>
<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>autocode(<i>x</i>,<i>n</i>,<i>x</i>₀,<i>x</i>₁)</code>	partitions the interval from <i>x</i> ₀ to <i>x</i> ₁ into <i>n</i> equal-length intervals and returns the upper bound of the interval that contains <i>x</i>
<code>betaden(<i>a</i>,<i>b</i>,<i>x</i>)</code>	the probability density of the beta distribution, where <i>a</i> and <i>b</i> are the shape parameters; 0 if <i>x</i> < 0 or <i>x</i> > 1
<code>binomial(<i>n</i>,<i>k</i>,θ)</code>	the probability of observing <code>floor(<i>k</i>)</code> or fewer successes in <code>floor(<i>n</i>)</code> trials when the probability of a success on one trial is θ ; 0 if <i>k</i> < 0; or 1 if <i>k</i> > <i>n</i>
<code>binomialp(<i>n</i>,<i>k</i>,<i>p</i>)</code>	the probability of observing <code>floor(<i>k</i>)</code> successes in <code>floor(<i>n</i>)</code> trials when the probability of a success on one trial is <i>p</i>
<code>binomialtail(<i>n</i>,<i>k</i>,θ)</code>	the probability of observing <code>floor(<i>k</i>)</code> or more successes in <code>floor(<i>n</i>)</code> trials when the probability of a success on one trial is θ ; 1 if <i>k</i> < 0; or 0 if <i>k</i> > <i>n</i>
<code>binormal(<i>h</i>,<i>k</i>,ρ)</code>	the joint cumulative distribution $\Phi(h, k, \rho)$ of bivariate normal with correlation ρ
<code>bofd("cal",<i>e</i>_d)</code>	the <i>e</i> _b business date corresponding to <i>e</i> _d
<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(<i>name</i>)</code>	the value of the system or constant result <i>c</i> (<i>name</i>) (see [P] creturn)
<code>_caller()</code>	version of the program or session that invoked the currently running program; see [P] version
<code>cauchy(<i>a</i>,<i>b</i>,<i>x</i>)</code>	the cumulative Cauchy distribution with location parameter <i>a</i> and scale parameter <i>b</i>
<code>cauchyden(<i>a</i>,<i>b</i>,<i>x</i>)</code>	the probability density of the Cauchy distribution with location parameter <i>a</i> and scale parameter <i>b</i>
<code>cauchytail(<i>a</i>,<i>b</i>,<i>x</i>)</code>	the reverse cumulative (upper tail or survivor) Cauchy distribution with location parameter <i>a</i> and scale parameter <i>b</i>
<code>Cdhms(<i>e</i>_d,<i>h</i>,<i>m</i>,<i>s</i>)</code>	the <i>e</i> _{tC} datetime (ms. with leap seconds since 01jan1960 00:00:00.000) corresponding to <i>e</i> _d , <i>h</i> , <i>m</i> , <i>s</i>
<code>ceil(<i>x</i>)</code>	the unique integer <i>n</i> such that $n - 1 < x \leq n$; <i>x</i> (not ".") if <i>x</i> is missing, meaning that <code>ceil(.a) = .a</code>

2 Functions by name

<code>char(<i>n</i>)</code>	the character corresponding to ASCII or extended ASCII code <i>n</i> ; "" if <i>n</i> is not in the domain
<code>chi2(<i>df</i>, <i>x</i>)</code>	the cumulative χ^2 distribution with <i>df</i> degrees of freedom; 0 if $x < 0$
<code>chi2den(<i>df</i>, <i>x</i>)</code>	the probability density of the chi-squared distribution with <i>df</i> degrees of freedom; 0 if $x < 0$
<code>chi2tail(<i>df</i>, <i>x</i>)</code>	the reverse cumulative (upper tail or survivor) χ^2 distribution with <i>df</i> degrees of freedom; 1 if $x < 0$
<code>Chms(<i>h</i>, <i>m</i>, <i>s</i>)</code>	the e_{tC} datetime (ms. with leap seconds since 01jan1960 00:00:00.000) corresponding to <i>h</i> , <i>m</i> , <i>s</i> on 01jan1960
<code>chop(<i>x</i>, ϵ)</code>	<code>round(<i>x</i>)</code> if $\text{abs}(x - \text{round}(x)) < \epsilon$; otherwise, <i>x</i> ; or <i>x</i> if <i>x</i> is missing
<code>cholesky(<i>M</i>)</code>	the Cholesky decomposition of the matrix: if $R = \text{cholesky}(S)$, then $RR^T = S$
<code>clip(<i>x</i>, <i>a</i>, <i>b</i>)</code>	<i>x</i> if $a < x < b$, <i>b</i> if $x \geq b$, <i>a</i> if $x \leq a$, or <i>missing</i> if <i>x</i> is missing or if $a > b$; <i>x</i> if <i>x</i> is missing
<code>Clock(<i>s</i>₁, <i>s</i>₂ [, <i>Y</i>])</code>	the e_{tC} datetime (ms. with leap seconds since 01jan1960 00:00:00.000) corresponding to <i>s</i> ₁ based on <i>s</i> ₂ and <i>Y</i>
<code>clock(<i>s</i>₁, <i>s</i>₂ [, <i>Y</i>])</code>	the e_{tc} datetime (ms. since 01jan1960 00:00:00.000) corresponding to <i>s</i> ₁ based on <i>s</i> ₂ and <i>Y</i>
<code>cloglog(<i>x</i>)</code>	the complementary log-log of <i>x</i>
<code>Cmdyhms(<i>M</i>, <i>D</i>, <i>Y</i>, <i>h</i>, <i>m</i>, <i>s</i>)</code>	the e_{tC} datetime (ms. with leap seconds since 01jan1960 00:00:00.000) corresponding to <i>M</i> , <i>D</i> , <i>Y</i> , <i>h</i> , <i>m</i> , <i>s</i>
<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>colegnumb(<i>M</i>, <i>s</i>)</code>	the equation number of <i>M</i> associated with column equation <i>s</i> ; <i>missing</i> if the column equation cannot be found
<code>collatorlocale(<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 collation data comes from if <i>type</i> is 2
<code>collatorversion(<i>loc</i>)</code>	the version string of a collator based on locale <i>loc</i>
<code>colnfreeparms(<i>M</i>, <i>s</i>)</code>	the number of free parameters in columns of <i>M</i>
<code>colnumb(<i>M</i>, <i>s</i>)</code>	the column number of <i>M</i> associated with column name <i>s</i> ; <i>missing</i> if the column cannot be found
<code>colsof(<i>M</i>)</code>	the number of columns of <i>M</i>
<code>comb(<i>n</i>, <i>k</i>)</code>	the combinatorial function $n! / \{k!(n - k)!\}$
<code>cond(<i>x</i>, <i>a</i>, <i>b</i> [, <i>c</i>])</code>	<i>a</i> if <i>x</i> is <i>true</i> and nonmissing, <i>b</i> if <i>x</i> is <i>false</i> , and <i>c</i> if <i>x</i> is <i>missing</i> ; <i>a</i> if <i>c</i> is not specified and <i>x</i> evaluates to <i>missing</i>
<code>corr(<i>M</i>)</code>	the correlation matrix of the variance matrix

<code>cos(x)</code>	the cosine of x , where x is in radians
<code>cosh(x)</code>	the hyperbolic cosine of x
<code>daily(s1, s2[, Y])</code>	a synonym for <code>date(s1, s2[, Y])</code>
<code>date(s1, s2[, Y])</code>	the e_d date (days since 01jan1960) corresponding to s_1 based on s_2 and Y
<code>day(e_d)</code>	the numeric day of the month corresponding to e_d
<code>det(M)</code>	the determinant of matrix M
<code>dgamma_pda(a, x)</code>	$\frac{\partial P(a, x)}{\partial a}$, where $P(a, x) = \text{gamma}(a, x)$; 0 if $x < 0$
<code>dgamma_pdda(a, x)</code>	$\frac{\partial^2 P(a, x)}{\partial a^2}$, where $P(a, x) = \text{gamma}(a, x)$; 0 if $x < 0$
<code>dgamma_pdadx(a, x)</code>	$\frac{\partial^2 P(a, x)}{\partial a \partial x}$, where $P(a, x) = \text{gamma}(a, x)$; 0 if $x < 0$
<code>dgamma_pdx(a, x)</code>	$\frac{\partial P(a, x)}{\partial x}$, where $P(a, x) = \text{gamma}(a, x)$; 0 if $x < 0$
<code>dgamma_pdxdx(a, x)</code>	$\frac{\partial^2 P(a, x)}{\partial x^2}$, where $P(a, x) = \text{gamma}(a, x)$; 0 if $x < 0$
<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 , and s
<code>diag(v)</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>digamma(x)</code>	the <code>digamma()</code> function, $d \ln \Gamma(x) / dx$
<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>doy(e_d)</code>	the numeric day of the year corresponding to date e_d
<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>e(name)</code>	the value of stored result <code>e(name)</code> ; see [U] 18.8 Accessing results calculated by other programs
<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>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>exp(x)</code>	the exponential function e^x
<code>exponential(b, x)</code>	the cumulative exponential distribution with scale b

4 Functions by name

<code>exponentialden(<i>b</i>,<i>x</i>)</code>	the probability density function of the exponential distribution with scale <i>b</i>
<code>exponentialtail(<i>b</i>,<i>x</i>)</code>	the reverse cumulative exponential distribution with scale <i>b</i>
<code>F(<i>df</i>₁,<i>df</i>₂,<i>f</i>)</code>	the cumulative <i>F</i> distribution with <i>df</i> ₁ numerator and <i>df</i> ₂ denominator degrees of freedom: $F(df_1, df_2, f) = \int_0^f Fden(df_1, df_2, t) dt$; 0 if <i>f</i> < 0
<code>Fden(<i>df</i>₁,<i>df</i>₂,<i>f</i>)</code>	the probability density function of the <i>F</i> distribution with <i>df</i> ₁ numerator and <i>df</i> ₂ denominator degrees of freedom; 0 if <i>f</i> < 0
<code>fileexists(<i>f</i>)</code>	1 if the file specified by <i>f</i> exists; otherwise, 0
<code>fileread(<i>f</i>)</code>	the contents of the file specified by <i>f</i>
<code>filereaderror(<i>f</i>)</code>	0 or positive integer, said value having the interpretation of a return code
<code>filewrite(<i>f</i>,<i>s</i>[,<i>r</i>])</code>	writes the string specified by <i>s</i> to the file specified by <i>f</i> and returns the number of bytes in the resulting file
<code>float(<i>x</i>)</code>	the value of <i>x</i> rounded to float precision
<code>floor(<i>x</i>)</code>	the unique integer <i>n</i> such that $n \leq x < n + 1$; <i>x</i> (not “.”) if <i>x</i> is missing, meaning that <code>floor(.a) = .a</code>
<code>fmtwidth(<i>fmtstr</i>)</code>	the output length of the <i>%fmt</i> contained in <i>fmtstr</i> ; <i>missing</i> if <i>fmtstr</i> does not contain a valid <i>%fmt</i>
<code>Ftail(<i>df</i>₁,<i>df</i>₂,<i>f</i>)</code>	the reverse cumulative (upper tail or survivor) <i>F</i> distribution with <i>df</i> ₁ numerator and <i>df</i> ₂ denominator degrees of freedom; 1 if <i>f</i> < 0
<code>gammaden(<i>a</i>,<i>b</i>,<i>g</i>,<i>x</i>)</code>	the probability density function of the gamma distribution; 0 if <i>x</i> < <i>g</i>
<code>gammap(<i>a</i>,<i>x</i>)</code>	the cumulative gamma distribution with shape parameter <i>a</i> ; 0 if <i>x</i> < 0
<code>gammaptail(<i>a</i>,<i>x</i>)</code>	the reverse cumulative (upper tail or survivor) gamma distribution with shape parameter <i>a</i> ; 1 if <i>x</i> < 0
<code>get(<i>systemname</i>)</code>	a copy of Stata internal system matrix <i>systemname</i>
<code>hadamard(<i>M</i>,<i>N</i>)</code>	a matrix whose <i>i</i> , <i>j</i> element is $M[i, j] \cdot N[i, j]$ (if <i>M</i> and <i>N</i> are not the same size, this function reports a conformability error)
<code>halfyear(<i>e</i>_{<i>d</i>})</code>	the numeric half of the year corresponding to date <i>e</i> _{<i>d</i>}
<code>halfyearly(<i>s</i>₁,<i>s</i>₂[,<i>Y</i>])</code>	the <i>e</i> _{<i>h</i>} half-yearly date (half-years since 1960h1) corresponding to <i>s</i> ₁ based on <i>s</i> ₂ and <i>Y</i> ; <i>Y</i> specifies <i>topyear</i> ; see <code>date()</code>
<code>has_eprop(<i>name</i>)</code>	1 if <i>name</i> appears as a word in <code>e(properties)</code> ; otherwise, 0
<code>hh(<i>e</i>_{<i>tc</i>})</code>	the hour corresponding to datetime <i>e</i> _{<i>tc</i>} (ms. since 01jan1960 00:00:00.000)
<code>hhC(<i>e</i>_{<i>tC</i>})</code>	the hour corresponding to datetime <i>e</i> _{<i>tC</i>} (ms. with leap seconds since 01jan1960 00:00:00.000)
<code>hms(<i>h</i>,<i>m</i>,<i>s</i>)</code>	the <i>e</i> _{<i>tc</i>} datetime (ms. since 01jan1960 00:00:00.000) corresponding to <i>h</i> , <i>m</i> , <i>s</i> on 01jan1960
<code>hofd(<i>e</i>_{<i>d</i>})</code>	the <i>e</i> _{<i>h</i>} half-yearly date (half years since 1960h1) containing date <i>e</i> _{<i>d</i>}
<code>hours(<i>ms</i>)</code>	$ms/3,600,000$
<code>hypergeometric(<i>N</i>,<i>K</i>,<i>n</i>,<i>k</i>)</code>	the cumulative probability of the hypergeometric distribution
<code>hypergeometricp(<i>N</i>,<i>K</i>,<i>n</i>,<i>k</i>)</code>	the hypergeometric probability of <i>k</i> successes out of a sample of size <i>n</i> , from a population of size <i>N</i> containing <i>K</i> elements that have the attribute of interest

<code>I(n)</code>	an $n \times n$ identity matrix if n is an integer; otherwise, a <code>round(n) × round(n)</code> identity matrix
<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>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>inlist(z,a,b,...)</code>	1 if z 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>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>inv(M)</code>	the inverse of the matrix M
<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>invcloglog(x)</code>	the inverse of the complementary log-log function of x
<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(df₁,df₂,p)</code>	the inverse cumulative F distribution: if <code>F(df₁,df₂,f) = p</code> , then <code>invF(df₁,df₂,p) = f</code>
<code>invFtail(df₁,df₂,p)</code>	the inverse reverse cumulative (upper tail or survivor) F distribution: if <code>Ftail(df₁,df₂,f) = p</code> , then <code>invFtail(df₁,df₂,p) = f</code>

6 Functions by name

<code>invgammap(<i>a,p</i>)</code>	the inverse cumulative gamma distribution: if <code>gammap(<i>a,x</i>) = <i>p</i></code> , then <code>invgammap(<i>a,p</i>) = <i>x</i></code>
<code>invgammaptail(<i>a,p</i>)</code>	the inverse reverse cumulative (upper tail or survivor) gamma distribution: if <code>gammaptail(<i>a,x</i>) = <i>p</i></code> , then <code>invgammaptail(<i>a,p</i>) = <i>x</i></code>
<code>invibeta(<i>a,b,p</i>)</code>	the inverse cumulative beta distribution: if <code>ibeta(<i>a,b,x</i>) = <i>p</i></code> , then <code>invibeta(<i>a,b,p</i>) = <i>x</i></code>
<code>invibetatail(<i>a,b,p</i>)</code>	the inverse reverse cumulative (upper tail or survivor) beta distribution: if <code>ibetatail(<i>a,b,x</i>) = <i>p</i></code> , then <code>invibetatail(<i>a,b,p</i>) = <i>x</i></code>
<code>invgaussian(<i>m,a,p</i>)</code>	the inverse of <code>igaussian()</code> : if <code>igaussian(<i>m,a,x</i>) = <i>p</i></code> , then <code>invgaussian(<i>m,a,p</i>) = <i>x</i></code>
<code>invgaussiantail(<i>m,a,p</i>)</code>	the inverse of <code>igaussiantail()</code> : if <code>igaussiantail(<i>m,a,x</i>) = <i>p</i></code> , then <code>invgaussiantail(<i>m,a,p</i>) = <i>x</i></code>
<code>invlaplace(<i>m,b,p</i>)</code>	the inverse of <code>laplace()</code> : if <code>laplace(<i>m,b,x</i>) = <i>p</i></code> , then <code>invlaplace(<i>m,b,p</i>) = <i>x</i></code>
<code>invlaplacetail(<i>m,b,p</i>)</code>	the inverse of <code>laplacetail()</code> : if <code>laplacetail(<i>m,b,x</i>) = <i>p</i></code> , then <code>invlaplacetail(<i>m,b,p</i>) = <i>x</i></code>
<code>invlogistic(<i>p</i>)</code>	the inverse cumulative logistic distribution: if <code>logistic(<i>x</i>) = <i>p</i></code> , then <code>invlogistic(<i>p</i>) = <i>x</i></code>
<code>invlogistic(<i>s,p</i>)</code>	the inverse cumulative logistic distribution: if <code>logistic(<i>s,x</i>) = <i>p</i></code> , then <code>invlogistic(<i>s,p</i>) = <i>x</i></code>
<code>invlogistic(<i>m,s,p</i>)</code>	the inverse cumulative logistic distribution: if <code>logistic(<i>m,s,x</i>) = <i>p</i></code> , then <code>invlogistic(<i>m,s,p</i>) = <i>x</i></code>
<code>invlogistictail(<i>p</i>)</code>	the inverse reverse cumulative logistic distribution: if <code>logistictail(<i>x</i>) = <i>p</i></code> , then <code>invlogistictail(<i>p</i>) = <i>x</i></code>
<code>invlogistictail(<i>s,p</i>)</code>	the inverse cumulative logistic distribution: if <code>logistic(<i>s,x</i>) = <i>p</i></code> , then <code>invlogistic(<i>s,p</i>) = <i>x</i></code>
<code>invlogistictail(<i>m,s,p</i>)</code>	the inverse cumulative logistic distribution: if <code>logistic(<i>m,s,x</i>) = <i>p</i></code> , then <code>invlogistic(<i>m,s,p</i>) = <i>x</i></code>
<code>invlogit(<i>x</i>)</code>	the inverse of the logit function of <i>x</i>
<code>invnbinomial(<i>n,k,q</i>)</code>	the value of the negative binomial parameter, <i>p</i> , such that <code>q = nbinomial(<i>n,k,p</i>)</code>
<code>invnbinomialtail(<i>n,k,q</i>)</code>	the value of the negative binomial parameter, <i>p</i> , such that <code>q = nbinomialtail(<i>n,k,p</i>)</code>
<code>invnchi2(<i>df,np,p</i>)</code>	the inverse cumulative noncentral χ^2 distribution: if <code>nchi2(<i>df,np,x</i>) = <i>p</i></code> , then <code>invnchi2(<i>df,np,p</i>) = <i>x</i></code>
<code>invnchi2tail(<i>df,np,p</i>)</code>	the inverse reverse cumulative (upper tail or survivor) non-central χ^2 distribution: if <code>nchi2tail(<i>df,np,x</i>) = <i>p</i></code> , then <code>invnchi2tail(<i>df,np,p</i>) = <i>x</i></code>
<code>invnF(<i>df₁,df₂,np,p</i>)</code>	the inverse cumulative noncentral <i>F</i> distribution: if <code>nF(<i>df₁,df₂,np,f</i>) = <i>p</i></code> , then <code>invnF(<i>df₁,df₂,np,p</i>) = <i>f</i></code>
<code>invnFtail(<i>df₁,df₂,np,p</i>)</code>	the inverse reverse cumulative (upper tail or survivor) non-central <i>F</i> distribution: if <code>nFtail(<i>df₁,df₂,np,x</i>) = <i>p</i></code> , then <code>invnFtail(<i>df₁,df₂,np,p</i>) = <i>x</i></code>
<code>invnibeta(<i>a,b,np,p</i>)</code>	the inverse cumulative noncentral beta distribution: if <code>nibeta(<i>a,b,np,x</i>) = <i>p</i></code> , then <code>invnibeta(<i>a,b,np,p</i>) = <i>x</i></code>

<code>invnormal(p)</code>	the inverse cumulative standard normal distribution: if <code>normal(z) = p</code> , then <code>invnormal(p) = z</code>
<code>invnt(df,np,p)</code>	the inverse cumulative noncentral Student's t distribution: if <code>nt(df,np,t) = p</code> , then <code>invnt(df,np,p) = t</code>
<code>invnttail(df,np,p)</code>	the inverse reverse cumulative (upper tail or survivor) noncentral Student's t distribution: if <code>nttail(df,np,t) = p</code> , then <code>invnttail(df,np,p) = t</code>
<code>invpoisson(k,p)</code>	the Poisson mean such that the cumulative Poisson distribution evaluated at k is p : if <code>poisson(m,k) = p</code> , then <code>invpoisson(k,p) = m</code>
<code>invpoisontail(k,q)</code>	the Poisson mean such that the reverse cumulative Poisson distribution evaluated at k is q : if <code>poisontail(m,k) = q</code> , then <code>invpoisontail(k,q) = m</code>
<code>invsym(M)</code>	the inverse of M if M is positive definite
<code>invt(df,p)</code>	the inverse cumulative Student's t distribution: if <code>t(df,t) = p</code> , then <code>invt(df,p) = t</code>
<code>invttail(df,p)</code>	the inverse reverse cumulative (upper tail or survivor) Student's t distribution: if <code>ttail(df,t) = p</code> , then <code>invttail(df,p) = t</code>
<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 <code>weibull(a,b,x) = p</code> , then <code>invweibull(a,b,p) = x</code>
<code>invweibull(a,b,g,p)</code>	the inverse cumulative Weibull distribution with shape a , scale b , and location g : if <code>weibull(a,b,g,x) = p</code> , then <code>invweibull(a,b,g,p) = x</code>
<code>invweibullph(a,b,p)</code>	the inverse cumulative Weibull (proportional hazards) distribution with shape a and scale b : if <code>weibullph(a,b,x) = p</code> , then <code>invweibullph(a,b,p) = x</code>
<code>invweibullph(a,b,g,p)</code>	the inverse cumulative Weibull (proportional hazards) distribution with shape a , scale b , and location g : if <code>weibullph(a,b,g,x) = p</code> , then <code>invweibullph(a,b,g,p) = x</code>
<code>invweibullphtail(a,b,p)</code>	the inverse reverse cumulative Weibull (proportional hazards) distribution with shape a and scale b : if <code>weibullphtail(a,b,x) = p</code> , then <code>invweibullphtail(a,b,p) = x</code>
<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>irecode(x,x₁,...,x_n)</code>	<i>missing</i> if x 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 $x_2 < x \leq x_3$; ...; n if $x > x_n$
<code>issymmetric(M)</code>	1 if the matrix is symmetric; otherwise, 0

8 Functions by name

<code>J(r, c, z)</code>	the $r \times c$ matrix containing elements z
<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>ln(x)</code>	the natural logarithm, $\ln(x)$
<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>lnfactorial(n)</code>	the natural log of n factorial = $\ln(n!)$
<code>lngamma(x)</code>	$\ln\{\Gamma(x)\}$
<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>log(x)</code>	the natural logarithm, $\ln(x)$; thus, a synonym for <code>ln(x)</code>
<code>log10(x)</code>	the base-10 logarithm of x
<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(<i>s</i>,<i>x</i>)</code>	the reverse cumulative logistic distribution with mean 0, scale <i>s</i> , and standard deviation $s\pi/\sqrt{3}$
<code>logistictail(<i>m</i>,<i>s</i>,<i>x</i>)</code>	the reverse cumulative logistic distribution with mean <i>m</i> , scale <i>s</i> , and standard deviation $s\pi/\sqrt{3}$
<code>logit(<i>x</i>)</code>	the log of the odds ratio of <i>x</i> , $\text{logit}(x) = \ln\{x/(1-x)\}$
<code>matmissing(<i>M</i>)</code>	1 if any elements of the matrix are missing; otherwise, 0
<code>matrix(<i>exp</i>)</code>	restricts name interpretation to scalars and matrices; see <code>scalar()</code>
<code>matuniform(<i>r</i>,<i>c</i>)</code>	the $r \times c$ matrices containing uniformly distributed pseudorandom numbers on the interval (0, 1)
<code>max(<i>x</i>₁,<i>x</i>₂,...,<i>x</i>_{<i>n</i>})</code>	the maximum value of x_1, x_2, \dots, x_n
<code>maxbyte()</code>	the largest value that can be stored in storage type byte
<code>maxdouble()</code>	the largest value that can be stored in storage type double
<code>maxfloat()</code>	the largest value that can be stored in storage type float
<code>maxint()</code>	the largest value that can be stored in storage type int
<code>maxlong()</code>	the largest value that can be stored in storage type long
<code>mdy(<i>M</i>,<i>D</i>,<i>Y</i>)</code>	the e_d date (days since 01jan1960) corresponding to <i>M</i> , <i>D</i> , <i>Y</i>
<code>mdyhms(<i>M</i>,<i>D</i>,<i>Y</i>,<i>h</i>,<i>m</i>,<i>s</i>)</code>	the e_{tc} datetime (ms. since 01jan1960 00:00:00.000) corresponding to <i>M</i> , <i>D</i> , <i>Y</i> , <i>h</i> , <i>m</i> , <i>s</i>
<code>mi(<i>x</i>₁,<i>x</i>₂,...,<i>x</i>_{<i>n</i>})</code>	a synonym for <code>missing(x_1, x_2, \dots, x_n)</code>
<code>min(<i>x</i>₁,<i>x</i>₂,...,<i>x</i>_{<i>n</i>})</code>	the minimum value of x_1, x_2, \dots, x_n
<code>minbyte()</code>	the smallest value that can be stored in storage type byte
<code>mindouble()</code>	the smallest value that can be stored in storage type double
<code>minfloat()</code>	the smallest value that can be stored in storage type float
<code>minint()</code>	the smallest value that can be stored in storage type int
<code>minlong()</code>	the smallest value that can be stored in storage type long
<code>minutes(<i>ms</i>)</code>	$ms/60,000$
<code>missing(<i>x</i>₁,<i>x</i>₂,...,<i>x</i>_{<i>n</i>})</code>	1 if any x_i evaluates to <i>missing</i> ; otherwise, 0
<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>mod(<i>x</i>,<i>y</i>)</code>	the modulus of <i>x</i> with respect to <i>y</i>
<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 [, <i>Y</i>])</code>	the e_m monthly date (months since 1960m1) corresponding to s_1 based on s_2 and <i>Y</i> ; <i>Y</i> specifies <i>topyear</i> ; see <code>date()</code>
<code>mreldif(<i>X</i>,<i>Y</i>)</code>	the relative difference of <i>X</i> and <i>Y</i> , where the relative difference is defined as $\max_{i,j} \{ x_{ij} - y_{ij} / (y_{ij} + 1)\}$
<code>msofhours(<i>h</i>)</code>	$h \times 3,600,000$
<code>msofminutes(<i>m</i>)</code>	$m \times 60,000$
<code>msofseconds(<i>s</i>)</code>	$s \times 1,000$
<code>nbetaden(<i>a</i>,<i>b</i>,<i>np</i>,<i>x</i>)</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>nullmat($matname$)</code>	use with the row-join (<code>,</code>) and column-join (<code>\</code>) operators in programming situations
<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>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>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>r($name$)</code>	the value of the stored result <code>r($name$)</code> ; see [U] 18.8 Accessing results calculated by other programs
<code>rbeta(a, b)</code>	beta(a, b) random variates, where a and b are the beta distribution shape parameters
<code>rbinomial(n, p)</code>	binomial(n, p) random variates, where n is the number of trials and p is the success probability
<code>rcauchy(a, b)</code>	Cauchy(a, b) random variates, where a is the location parameter and b is the scale parameter
<code>rchi2(df)</code>	chi-squared, with df degrees of freedom, random variates
<code>recode(x, x_1, \dots, x_n)</code>	<i>missing</i> if x_1, x_2, \dots, x_n is not weakly increasing; x if x 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>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>reldif(x, y)</code>	the “relative” difference $ x - y /(y + 1)$; 0 if both arguments are the same type of extended missing value; <i>missing</i> if only one argument is missing or if the two arguments are two different types of <i>missing</i>
<code>replay()</code>	1 if the first nonblank character of local macro ‘0’ is a comma, or if ‘0’ is empty
<code>return($name$)</code>	the value of the to-be-stored result <code>r($name$)</code> ; see [P] return
<code>rexponential(b)</code>	exponential random variates with scale b
<code>rgamma(a, b)</code>	gamma(a, b) random variates, where a is the gamma shape parameter and b is the scale parameter
<code>rhypergeometric(N, K, n)</code>	hypergeometric random variates
<code>rigaussian(m, a)</code>	inverse Gaussian random variates with mean m and shape parameter a
<code>rlaplace(m, b)</code>	Laplace(m, b) random variates with mean m and scale parameter b
<code>rlogistic()</code>	logistic variates with mean 0 and standard deviation $\pi/\sqrt{3}$
<code>rlogistic(s)</code>	logistic variates with mean 0, scale s , and standard deviation $s\pi/\sqrt{3}$
<code>rlogistic(m, s)</code>	logistic variates with mean m , scale s , 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($m, 1$) (Gaussian) random variates, where m is the mean and the standard deviation is 1

<code>rnormal(<i>m</i>,<i>s</i>)</code>	normal(<i>m</i> , <i>s</i>) (Gaussian) random variates, where <i>m</i> is the mean and <i>s</i> is the standard deviation
<code>round(<i>x</i>,<i>y</i>)</code> or <code>round(<i>x</i>)</code>	<i>x</i> rounded in units of <i>y</i> or <i>x</i> rounded to the nearest integer if the argument <i>y</i> is omitted; <i>x</i> (not “.”) if <i>x</i> is missing (meaning that <code>round(.a) = .a</code> and that <code>round(.a,<i>y</i>) = .a</code> if <i>y</i> is not missing) and if <i>y</i> is missing, then “.” is returned
<code>roweqnumb(<i>M</i>,<i>s</i>)</code>	the equation number of <i>M</i> associated with row equation <i>s</i> ; <i>missing</i> if the row equation cannot be found
<code>rowfreeparms(<i>M</i>,<i>s</i>)</code>	the number of free parameters in rows of <i>M</i>
<code>rownumb(<i>M</i>,<i>s</i>)</code>	the row number of <i>M</i> associated with row name <i>s</i> ; <i>missing</i> if the row cannot be found
<code>rowsof(<i>M</i>)</code>	the number of rows of <i>M</i>
<code>rpoisson(<i>m</i>)</code>	Poisson(<i>m</i>) random variates, where <i>m</i> is the distribution mean
<code>rt(<i>df</i>)</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(<i>a</i>,<i>b</i>)</code>	uniformly distributed random variates over the interval (<i>a</i> , <i>b</i>)
<code>runiformint(<i>a</i>,<i>b</i>)</code>	uniformly distributed random integer variates on the interval [<i>a</i> , <i>b</i>]
<code>rweibull(<i>a</i>,<i>b</i>)</code>	Weibull variates with shape <i>a</i> and scale <i>b</i>
<code>rweibull(<i>a</i>,<i>b</i>,<i>g</i>)</code>	Weibull variates with shape <i>a</i> , scale <i>b</i> , and location <i>g</i>
<code>rweibullph(<i>a</i>,<i>b</i>)</code>	Weibull (proportional hazards) variates with shape <i>a</i> and scale <i>b</i>
<code>rweibullph(<i>a</i>,<i>b</i>,<i>g</i>)</code>	Weibull (proportional hazards) variates with shape <i>a</i> , scale <i>b</i> , and location <i>g</i>
<code>s(<i>name</i>)</code>	the value of stored result <i>s</i> (<i>name</i>); see [U] 18.8 Accessing results calculated by other programs
<code>scalar(<i>exp</i>)</code>	restricts name interpretation to scalars and matrices
<code>seconds(<i>ms</i>)</code>	<i>ms</i> /1,000
<code>sign(<i>x</i>)</code>	the sign of <i>x</i> : -1 if <i>x</i> < 0, 0 if <i>x</i> = 0, 1 if <i>x</i> > 0, or <i>missing</i> if <i>x</i> is missing
<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>smallestdouble()</code>	the smallest double-precision number greater than zero
<code>soundex(<i>s</i>)</code>	the soundex code for a string, <i>s</i>
<code>soundex_nara(<i>s</i>)</code>	the U.S. Census soundex code for a string, <i>s</i>
<code>sqrt(<i>x</i>)</code>	the square root of <i>x</i>
<code>ss(<i>e_{tc}</i>)</code>	the second corresponding to datetime <i>e_{tc}</i> (ms. since 01jan1960 00:00:00.000)
<code>ssC(<i>e_{tC}</i>)</code>	the second corresponding to datetime <i>e_{tC}</i> (ms. with leap seconds since 01jan1960 00:00:00.000)
<code>strcat(<i>s</i>₁,<i>s</i>₂)</code>	there is no <code>strcat()</code> function; instead the addition operator is used to concatenate strings
<code>strdup(<i>s</i>₁,<i>n</i>)</code>	there is no <code>strdup()</code> function; instead the multiplication operator is used to create multiple copies of strings
<code>string(<i>n</i>)</code>	a synonym for <code>stroofreal(<i>n</i>)</code>
<code>string(<i>n</i>,<i>s</i>)</code>	a synonym for <code>stroofreal(<i>n</i>,<i>s</i>)</code>

<code>stritrim(<i>s</i>)</code>	<i>s</i> with multiple, consecutive internal blanks (ASCII space character <code>char(32)</code>) collapsed to one blank
<code>strlen(<i>s</i>)</code>	the number of characters in ASCII <i>s</i> or length in bytes
<code>strlower(<i>s</i>)</code>	lowercase ASCII characters in string <i>s</i>
<code>strltrim(<i>s</i>)</code>	<i>s</i> without leading blanks (ASCII space character <code>char(32)</code>)
<code>strmatch(<i>s</i>₁,<i>s</i>₂)</code>	1 if <i>s</i> ₁ matches the pattern <i>s</i> ₂ ; otherwise, 0
<code>stroofreal(<i>n</i>)</code>	<i>n</i> converted to a string
<code>stroofreal(<i>n</i>,<i>s</i>)</code>	<i>n</i> converted to a string using the specified display format
<code>strpos(<i>s</i>₁,<i>s</i>₂)</code>	the position in <i>s</i> ₁ at which <i>s</i> ₂ is first found; otherwise, 0
<code>strproper(<i>s</i>)</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(<i>s</i>)</code>	reverses the ASCII string <i>s</i>
<code>strrpos(<i>s</i>₁,<i>s</i>₂)</code>	the position in <i>s</i> ₁ at which <i>s</i> ₂ is last found; otherwise, 0
<code>strrtrim(<i>s</i>)</code>	<i>s</i> without trailing blanks (ASCII space character <code>char(32)</code>)
<code>strtoname(<i>s</i>[,<i>p</i>])</code>	<i>s</i> translated into a Stata 13 compatible name
<code>strtrim(<i>s</i>)</code>	<i>s</i> without leading and trailing blanks (ASCII space character <code>char(32)</code>); equivalent to <code>strltrim(strrtrim(<i>s</i>))</code>
<code>strupper(<i>s</i>)</code>	uppercase ASCII characters in string <i>s</i>
<code>subinstr(<i>s</i>₁,<i>s</i>₂,<i>s</i>₃,<i>n</i>)</code>	<i>s</i> ₁ , where the first <i>n</i> occurrences in <i>s</i> ₁ of <i>s</i> ₂ have been replaced with <i>s</i> ₃
<code>subinword(<i>s</i>₁,<i>s</i>₂,<i>s</i>₃,<i>n</i>)</code>	<i>s</i> ₁ , where the first <i>n</i> occurrences in <i>s</i> ₁ of <i>s</i> ₂ as a word have been replaced with <i>s</i> ₃
<code>substr(<i>s</i>,<i>n</i>₁,<i>n</i>₂)</code>	the substring of <i>s</i> , starting at <i>n</i> ₁ , for a length of <i>n</i> ₂
<code>sum(<i>x</i>)</code>	the running sum of <i>x</i> , treating missing values as zero
<code>sweep(<i>M</i>,<i>i</i>)</code>	matrix <i>M</i> with <i>i</i> th row/column swept
<code>t(<i>df</i>,<i>t</i>)</code>	the cumulative Student's <i>t</i> distribution with <i>df</i> degrees of freedom
<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>
<code>tC(<i>l</i>)</code>	convenience function to make typing dates and times in expressions easier
<code>tC(<i>l</i>)</code>	convenience function to make typing dates and times in expressions easier
<code>td(<i>l</i>)</code>	convenience function to make typing dates in expressions easier
<code>tden(<i>df</i>,<i>t</i>)</code>	the probability density function of Student's <i>t</i> distribution
<code>th(<i>l</i>)</code>	convenience function to make typing half-yearly dates in expressions easier
<code>tin(<i>d</i>₁,<i>d</i>₂)</code>	<i>true</i> if <i>d</i> ₁ ≤ <i>t</i> ≤ <i>d</i> ₂ , where <i>t</i> is the time variable previously <code>tsset</code>
<code>tm(<i>l</i>)</code>	convenience function to make typing monthly dates in expressions easier
<code>tobytes(<i>s</i>[,<i>n</i>])</code>	escaped decimal or hex digit strings of up to 200 bytes of <i>s</i>
<code>tq(<i>l</i>)</code>	convenience function to make typing quarterly dates in expressions easier
<code>trace(<i>M</i>)</code>	the trace of matrix <i>M</i>
<code>trigamma(<i>x</i>)</code>	the second derivative of <code>lngamma(<i>x</i>) = d² lnΓ(<i>x</i>)/d<i>x</i>²</code>

<code>trunc(<i>x</i>)</code>	a synonym for <code>int(<i>x</i>)</code>
<code>ttail(<i>df</i>, <i>t</i>)</code>	the reverse cumulative (upper tail or survivor) Student's <i>t</i> distribution; the probability $T > t$
<code>tukeyprob(<i>k</i>, <i>df</i>, <i>x</i>)</code>	the cumulative Tukey's Studentized range distribution with <i>k</i> ranges and <i>df</i> degrees of freedom; 0 if $x < 0$
<code>tw(<i>l</i>)</code>	convenience function to make typing weekly dates in expressions easier
<code>twithin(<i>d</i>₁, <i>d</i>₂)</code>	<i>true</i> if $d_1 < t < d_2$, where <i>t</i> is the time variable previously <code>tset</code>
<code>uchar(<i>n</i>)</code>	the Unicode character corresponding to Unicode code point <i>n</i> or an empty string if <i>n</i> 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,loc,st,case,cslv,norm,num,alt,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,enc,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,n</i>[,<i>noc</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>s₁,s₂,s₃,n</i>)</code>	replaces the first <i>n</i> occurrences of the Unicode string <i>s₂</i> with the Unicode string <i>s₃</i> in <i>s₁</i>
<code>usubstr(<i>s,n₁,n₂</i>)</code>	the Unicode substring of <i>s</i> , starting at <i>n₁</i> , for a length of <i>n₂</i>
<code>vec(<i>M</i>)</code>	a column vector formed by listing the elements of <i>M</i> , starting with the first column and proceeding column by column
<code>vecdiag(<i>M</i>)</code>	the row vector containing the diagonal of matrix <i>M</i>
<code>week(<i>e_d</i>)</code>	the numeric week of the year corresponding to date <i>e_d</i> , the %td encoded date (days since 01jan1960)
<code>weekly(<i>s₁,s₂</i>[,<i>Y</i>])</code>	the <i>e_w</i> weekly date (weeks since 1960w1) corresponding to <i>s₁</i> based on <i>s₂</i> and <i>Y</i> ; <i>Y</i> specifies <i>topyear</i> ; see <code>date()</code>
<code>weibull(<i>a,b,x</i>)</code>	the cumulative Weibull distribution with shape <i>a</i> and scale <i>b</i>
<code>weibull(<i>a,b,g,x</i>)</code>	the cumulative Weibull distribution with shape <i>a</i> , scale <i>b</i> , and location <i>g</i>
<code>weibullden(<i>a,b,x</i>)</code>	the probability density function of the Weibull distribution with shape <i>a</i> and scale <i>b</i>
<code>weibullden(<i>a,b,g,x</i>)</code>	the probability density function of the Weibull distribution with shape <i>a</i> , scale <i>b</i> , and location <i>g</i>
<code>weibullph(<i>a,b,x</i>)</code>	the cumulative Weibull (proportional hazards) distribution with shape <i>a</i> and scale <i>b</i>
<code>weibullph(<i>a,b,g,x</i>)</code>	the cumulative Weibull (proportional hazards) distribution with shape <i>a</i> , scale <i>b</i> , and location <i>g</i>
<code>weibullphden(<i>a,b,x</i>)</code>	the probability density function of the Weibull (proportional hazards) distribution with shape <i>a</i> and scale <i>b</i>
<code>weibullphden(<i>a,b,g,x</i>)</code>	the probability density function of the Weibull (proportional hazards) distribution with shape <i>a</i> , scale <i>b</i> , and location <i>g</i>
<code>weibullphtail(<i>a,b,x</i>)</code>	the reverse cumulative Weibull (proportional hazards) distribution with shape <i>a</i> and scale <i>b</i>
<code>weibullphtail(<i>a,b,g,x</i>)</code>	the reverse cumulative Weibull (proportional hazards) distribution with shape <i>a</i> , scale <i>b</i> , and location <i>g</i>

<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
<code>wofd(e_d)</code>	the e_w weekly date (weeks since 1960w1) containing date e_d
<code>word(s,n)</code>	the n th word in s ; <i>missing</i> ("") if n is missing
<code>wordbreaklocale(loc,type)</code>	the most closely related locale supported by ICU from loc if $type$ is 1, the actual locale where the word-boundary analysis data come from if $type$ is 2; or an empty string is returned for any other $type$
<code>wordcount(s)</code>	the number of words in s
<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

Also see

[FN] **Functions by category**

[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**