Functions by name

`abbrev(`s`,`n`)`	name s, abbreviated to a length of n
`abs(`x`)`	the absolute value of x
`acos(`x`)`	the radian value of the arccosine of x
`acosh(`x`)`	the inverse hyperbolic cosine of x
`age(`$e_{d\,\mathrm{DOB}}$`,`e_d`[,`s_{nl}`])`	the age in integer years on e_d for date of birth $e_{d\,\mathrm{DOB}}$ with s_{nl} the nonleap-year birthday for 29feb birthdates
`age_frac(`$e_{d\,\mathrm{DOB}}$`,`e_d`[,`s_{nl}`])`	the age in years, including the fractional part, on e_d for date of birth $e_{d\,\mathrm{DOB}}$ with s_{nl} the nonleap-year birthday for 29feb birthdates
`asin(`x`)`	the radian value of the arcsine of x
`asinh(`x`)`	the inverse hyperbolic sine of x
`atan(`x`)`	the radian value of the arctangent of x
`atan2(`y`, `x`)`	the radian value of the arctangent of y/x, where the signs of the parameters y and x are used to determine the quadrant of the answer
`atanh(`x`)`	the inverse hyperbolic tangent of x
`autocode(`x`,`n`,`x_0`,`x_1`)`	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
`betaden(`a`,`b`,`x`)`	the probability density of the beta distribution, where a and b are the shape parameters; `0` if $x < 0$ or $x > 1$
`binomial(`n`,`k`,`θ`)`	the probability of observing `floor(`k`)` or fewer successes in `floor(`n`)` trials when the probability of a success on one trial is θ; `0` if $k < 0$; or `1` if $k > n$
`binomialp(`n`,`k`,`p`)`	the probability of observing `floor(`k`)` successes in `floor(`n`)` trials when the probability of a success on one trial is p
`binomialtail(`n`,`k`,`θ`)`	the probability of observing `floor(`k`)` or more successes in `floor(`n`)` trials when the probability of a success on one trial is θ; `1` if $k < 0$; or `0` if $k > n$
`binormal(`h`,`k`,`ρ`)`	the joint cumulative distribution $\Phi(h, k, \rho)$ of bivariate normal with correlation ρ
`birthday(`$e_{d\,\mathrm{DOB}}$`,`Y`[,`s_{nl}`])`	the e_d date of the birthday in year Y for date of birth $e_{d\,\mathrm{DOB}}$ with s_{nl} the nonleap-year birthday for 29feb birthdates
`bofd("`*cal*`",`e_d`)`	the e_b business date corresponding to e_d
`byteorder()`	`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
`c(`*name*`)`	the value of the system or constant result `c(`*name*`)` (see [P] **creturn**)
`_caller()`	`version` of the program or session that invoked the currently running program; see [P] **version**
`cauchy(`a`,`b`,`x`)`	the cumulative Cauchy distribution with location parameter a and scale parameter b
`cauchyden(`a`,`b`,`x`)`	the probability density of the Cauchy distribution with location parameter a and scale parameter b

`cauchytail(`a`,`b`,`x`)`	the reverse cumulative (upper tail or survivor) Cauchy distribution with location parameter a and scale parameter b
`Cdhms(`e_d`,`h`,`m`,`s`)`	the e_{tC} datetime (ms. with leap seconds since 01jan1960 00:00:00.000) corresponding to e_d, h, m, s
`ceil(`x`)`	the unique integer n such that $n-1 < x \leq n$; x (not "`.`") if x is missing, meaning that `ceil(.a)` = `.a`
`char(`n`)`	the character corresponding to ASCII or extended ASCII code n; `""` if n is not in the domain
`chi2(`df`,`x`)`	the cumulative χ^2 distribution with df degrees of freedom; `0` if $x < 0$
`chi2den(`df`,`x`)`	the probability density of the χ^2 distribution with df degrees of freedom; `0` if $x < 0$
`chi2tail(`df`,`x`)`	the reverse cumulative (upper tail or survivor) χ^2 distribution with df degrees of freedom; `1` if $x < 0$
`Chms(`h`,`m`,`s`)`	the e_{tC} datetime (ms. with leap seconds since 01jan1960 00:00:00.000) corresponding to h, m, s on 01jan1960
`chop(`x`, `ϵ`)`	`round(`x`)` if `abs(`x − `round(`x`))` $< \epsilon$; otherwise, x; or x if x is missing
`cholesky(`M`)`	the Cholesky decomposition of the matrix: if R = `cholesky(`S`)`, then $RR^T = S$
`clip(`x`,`a`,`b`)`	x if $a < x < b$, b if $x \geq b$, a if $x \leq a$, or *missing* if x is missing or if $a > b$; x if x is missing
`Clock(`s_1`,`s_2`[,`Y`])`	the e_{tC} datetime (ms. with leap seconds since 01jan1960 00:00:00.000) corresponding to s_1 based on s_2 and Y
`clock(`s_1`,`s_2`[,`Y`])`	the e_{tc} datetime (ms. since 01jan1960 00:00:00.000) corresponding to s_1 based on s_2 and Y
`Clockdiff(`e_{tC1}`, `e_{tC2}`,`s_u`)`	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
`clockdiff(`e_{tc1}`,`e_{tc2}`,`s_u`)`	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
`Clockdiff_frac(`e_{tC1}`,`e_{tC2}`,`s_u`)`	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
`clockdiff_frac(`e_{tc1}`,`e_{tc2}`,`s_u`)`	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
`Clockpart(`e_{tC}`,`s_u`)`	the integer year, month, day, hour, minute, second, or millisecond of e_{tC} with s_u specifying which time part
`clockpart(`e_{tc}`,`s_u`)`	the integer year, month, day, hour, minute, second, or millisecond of e_{tc} with s_u specifying which time part
`cloglog(`x`)`	the complementary log-log of x
`Cmdyhms(`M`,`D`,`Y`,`h`,`m`,`s`)`	the e_{tC} datetime (ms. with leap seconds since 01jan1960 00:00:00.000) corresponding to M, D, Y, h, m, s
`Cofc(`e_{tc}`)`	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)
`cofC(`e_{tC}`)`	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)

`Cofd(`e_d`)`	the e_{tC} datetime (ms. with leap seconds since 01jan1960 00:00:00.000) of date e_d at time 00:00:00.000
`cofd(`e_d`)`	the e_{tc} datetime (ms. since 01jan1960 00:00:00.000) of date e_d at time 00:00:00.000
`coleqnumb(`M`,`s`)`	the equation number of M associated with column equation s; *missing* if the column equation cannot be found
`collatorlocale(`*loc*`,`*type*`)`	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
`collatorversion(`*loc*`)`	the version string of a collator based on locale *loc*
`colnfreeparms(`M`)`	the number of free parameters in columns of M
`colnumb(`M`,`s`)`	the column number of M associated with column name s; *missing* if the column cannot be found
`colsof(`M`)`	the number of columns of M
`comb(`n`,`k`)`	the combinatorial function $n!/\{k!(n-k)!\}$
`cond(`x`,`a`,`b`[,`c`])`	a if x is *true* and nonmissing, b if x is *false*, and c if x is *missing*; a if c is not specified and x evaluates to *missing*
`corr(`M`)`	the correlation matrix of the variance matrix
`cos(`x`)`	the cosine of x, where x is in radians
`cosh(`x`)`	the hyperbolic cosine of x
`daily(`s_1`,`s_2`[,`Y`])`	a synonym for `date(`s_1`,`s_2`[,`Y`])`
`date(`s_1`,`s_2`[,`Y`])`	the e_d date (days since 01jan1960) corresponding to s_1 based on s_2 and Y
`datediff(`e_{d1}`,`e_{d2}`,`s_u`[,`s_{nl}`])`	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
`datediff_frac(`e_{d1}`,`e_{d2}`,`s_u`[,`s_{nl}`])`	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
`datepart(`e_d`,`s_u`)`	the integer year, month, or day of e_d with s_u specifying year, month, or day
`day(`e_d`)`	the numeric day of the month corresponding to e_d
`daysinmonth(`e_d`)`	the number of days in the month of e_d
`dayssincedow(`e_d`,`d`)`	a synonym for `dayssinceweekday(`e_d`,`d`)`
`dayssinceweekday(`e_d`,`d`)`	the number of days until e_d since previous day-of-week d
`daysuntildow(`e_d`,`d`)`	a synonym for `daysuntilweekday(`e_d`,`d`)`
`daysuntilweekday(`e_d`,`d`)`	the number of days from e_d until next day-of-week d
`det(`M`)`	the determinant of matrix M
`dgammapda(`a`,`x`)`	$\frac{\partial P(a,x)}{\partial a}$, where $P(a,x) =$ `gammap(`a`,`x`)`; 0 if $x<0$
`dgammapdada(`a`,`x`)`	$\frac{\partial^2 P(a,x)}{\partial a^2}$, where $P(a,x) =$ `gammap(`a`,`x`)`; 0 if $x<0$
`dgammapdadx(`a`,`x`)`	$\frac{\partial^2 P(a,x)}{\partial a \partial x}$, where $P(a,x) =$ `gammap(`a`,`x`)`; 0 if $x<0$
`dgammapdx(`a`,`x`)`	$\frac{\partial P(a,x)}{\partial x}$, where $P(a,x) =$ `gammap(`a`,`x`)`; 0 if $x<0$
`dgammapdxdx(`a`,`x`)`	$\frac{\partial^2 P(a,x)}{\partial x^2}$, where $P(a,x) =$ `gammap(`a`,`x`)`; 0 if $x<0$

dhms(e_d,h,m,s)	the e_{tc} datetime (ms. since 01jan1960 00:00:00.000) corresponding to e_d, h, m, and s
diag(M)	the square, diagonal matrix created from the row or column vector
diag0cnt(M)	the number of zeros on the diagonal of M
digamma(x)	the digamma() function, $d\ln\Gamma(x)/dx$
dmy(D,M,Y)	the e_d date (days since 01jan1960) corresponding to D, M, Y
dofb(e_b,"*cal*")	the e_d datetime corresponding to e_b
dofC(e_{tC})	the e_d date (days since 01jan1960) of datetime e_{tC} (ms. with leap seconds since 01jan1960 00:00:00.000)
dofc(e_{tc})	the e_d date (days since 01jan1960) of datetime e_{tc} (ms. since 01jan1960 00:00:00.000)
dofh(e_h)	the e_d date (days since 01jan1960) of the start of half-year e_h
dofm(e_m)	the e_d date (days since 01jan1960) of the start of month e_m
dofq(e_q)	the e_d date (days since 01jan1960) of the start of quarter e_q
dofw(e_w)	the e_d date (days since 01jan1960) of the start of week e_w
dofy(e_y)	the e_d date (days since 01jan1960) of 01jan in year e_y
dow(e_d)	the numeric day of the week corresponding to date e_d; $0 =$ Sunday, $1 =$ Monday, ..., $6 =$ Saturday
doy(e_d)	the numeric day of the year corresponding to date e_d
dunnettprob(k,df,x)	the cumulative multiple range distribution that is used in Dunnett's multiple-comparison method with k ranges and df degrees of freedom; 0 if $x < 0$
e(*name*)	the value of stored result e(*name*); see **[U] 18.8 Accessing results calculated by other programs**
el(s,i,j)	s[floor(i),floor(j)], the i, j element of the matrix named s; *missing* if i or j are out of range or if matrix s does not exist
e(sample)	1 if the observation is in the estimation sample and 0 otherwise
epsdouble()	the machine precision of a double-precision number
epsfloat()	the machine precision of a floating-point number
exp(x)	the exponential function e^x
expm1(x)	$e^x - 1$ with higher precision than exp(x) $- 1$ for small values of $\|x\|$
exponential(b,x)	the cumulative exponential distribution with scale b
exponentialden(b,x)	the probability density function of the exponential distribution with scale b
exponentialtail(b,x)	the reverse cumulative exponential distribution with scale b
F(df_1,df_2,f)	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 $f < 0$
Fden(df_1,df_2,f)	the probability density function of the F distribution with df_1 numerator and df_2 denominator degrees of freedom; 0 if $f < 0$
fileexists(f)	1 if the file specified by f exists; otherwise, 0
fileread(f)	the contents of the file specified by f
filereaderror(s)	0 or positive integer, said value having the interpretation of a return code

`filewrite(`f`,`s[`,`r]`)`	writes the string specified by s to the file specified by f and returns the number of bytes in the resulting file
`firstdayofmonth(`e_d`)`	the e_d date of the first day of the month of e_d
`firstdowofmonth(`M`,`Y`,`d`)`	a synonym for `firstweekdayofmonth(`M`,`Y`,`d`)`
`firstweekdayofmonth(`M`,`Y`,`d`)`	the e_d date of the first day-of-week d in month M of year Y
`float(`x`)`	the value of x rounded to `float` precision
`floor(`x`)`	the unique integer n such that $n \leq x < n + 1$; x (not "`.`") if x is missing, meaning that `floor(.a)` $=$ `.a`
`fmtwidth(`*fmtstr*`)`	the output length of the *%fmt* contained in *fmtstr*; *missing* if *fmtstr* does not contain a valid *%fmt*
`frval()`	returns values of variables stored in other frames
`_frval()`	programmer's version of `frval()`
`Ftail(`df_1`,`df_2`,`f`)`	the reverse cumulative (upper tail or survivor) F distribution with df_1 numerator and df_2 denominator degrees of freedom; `1` if $f < 0$
`gammaden(`a`,`b`,`g`,`x`)`	the probability density function of the gamma distribution; `0` if $x < g$
`gammap(`a`,`x`)`	the cumulative gamma distribution with shape parameter a; `0` if $x < 0$
`gammaptail(`a`,`x`)`	the reverse cumulative (upper tail or survivor) gamma distribution with shape parameter a; `1` if $x < 0$
`get(`*systemname*`)`	a copy of Stata internal system matrix *systemname*
`hadamard(`M`,`N`)`	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)
`halfyear(`e_d`)`	the numeric half of the year corresponding to date e_d
`halfyearly(`s_1`,`s_2[`,`Y]`)`	the e_h half-yearly date (half-years since 1960h1) corresponding to s_1 based on s_2 and Y; Y specifies *topyear*; see `date()`
`has_eprop(`*name*`)`	`1` if *name* appears as a word in `e(properties)`; otherwise, `0`
`hh(`e_{tc}`)`	the hour corresponding to datetime e_{tc} (ms. since 01jan1960 00:00:00.000)
`hhC(`e_{tC}`)`	the hour corresponding to datetime e_{tC} (ms. with leap seconds since 01jan1960 00:00:00.000)
`hms(`h`,`m`,`s`)`	the e_{tc} datetime (ms. since 01jan1960 00:00:00.000) corresponding to h, m, s on 01jan1960
`hofd(`e_d`)`	the e_h half-yearly date (half years since 1960h1) containing date e_d
`hours(`ms`)`	$ms/3{,}600{,}000$
`hypergeometric(`N`,`K`,`n`,`k`)`	the cumulative probability of the hypergeometric distribution
`hypergeometricp(`N`,`K`,`n`,`k`)`	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
`I(`n`)`	an $n \times n$ identity matrix if n is an integer; otherwise, a `round(`n`)` $\times$ `round(`n`)` identity matrix
`ibeta(`a`,`b`,`x`)`	the cumulative beta distribution with shape parameters a and b; `0` if $x < 0$; or `1` if $x > 1$
`ibetatail(`a`,`b`,`x`)`	the reverse cumulative (upper tail or survivor) beta distribution with shape parameters a and b; `1` if $x < 0$; or `0` if $x > 1$

igaussian(m,a,x)	the cumulative inverse Gaussian distribution with mean m and shape parameter a; 0 if $x \leq 0$
igaussianden(m,a,x)	the probability density of the inverse Gaussian distribution with mean m and shape parameter a; 0 if $x \leq 0$
igaussiantail(m,a,x)	the reverse cumulative (upper tail or survivor) inverse Gaussian distribution with mean m and shape parameter a; 1 if $x \leq 0$
indexnot(s_1,s_2)	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
inlist(z,a,b,...)	1 if z is a member of the remaining arguments; otherwise, 0
inrange(z,a,b)	1 if it is known that $a \leq z \leq b$; otherwise, 0
int(x)	the integer obtained by truncating x toward 0 (thus, int(5.2) $= 5$ and int(-5.8) $= -5$); x (not ".") if x is missing, meaning that int(.a) = .a
inv(M)	the inverse of the matrix M
invbinomial(n,k,p)	the inverse of the cumulative binomial; that is, θ ($\theta =$ probability of success on one trial) such that the probability of observing floor(k) or fewer successes in floor(n) trials is p
invbinomialtail(n,k,p)	the inverse of the right cumulative binomial; that is, θ ($\theta =$ probability of success on one trial) such that the probability of observing floor(k) or more successes in floor(n) trials is p
invcauchy(a,b,p)	the inverse of cauchy(): if cauchy(a,b,x) $= p$, then invcauchy(a,b,p) $= x$
invcauchytail(a,b,p)	the inverse of cauchytail(): if cauchytail(a,b,x) $= p$, then invcauchytail(a,b,p) $= x$
invchi2(df,p)	the inverse of chi2(): if chi2(df,x) $= p$, then invchi2(df,p) $= x$
invchi2tail(df,p)	the inverse of chi2tail(): if chi2tail(df,x) $= p$, then invchi2tail(df,p) $= x$
invcloglog(x)	the inverse of the complementary log-log function of x
invdunnettprob(k,df,p)	the inverse cumulative multiple range distribution that is used in Dunnett's multiple-comparison method with k ranges and df degrees of freedom
invexponential(b,p)	the inverse cumulative exponential distribution with scale b: if exponential(b,x) $= p$, then invexponential(b,p) $= x$
invexponentialtail(b,p)	the inverse reverse cumulative exponential distribution with scale b: if exponentialtail(b,x) $= p$, then invexponentialtail(b,p) $= x$
invF(df_1,df_2,p)	the inverse cumulative F distribution: if F(df_1,df_2,f) $= p$, then invF(df_1,df_2,p) $= f$
invFtail(df_1,df_2,p)	the inverse reverse cumulative (upper tail or survivor) F distribution: if Ftail(df_1,df_2,f) $= p$, then invFtail(df_1,df_2,p) $= f$
invgammap(a,p)	the inverse cumulative gamma distribution: if gammap(a,x) $= p$, then invgammap(a,p) $= x$
invgammaptail(a,p)	the inverse reverse cumulative (upper tail or survivor) gamma distribution: if gammaptail(a,x) $= p$, then invgammaptail(a,p) $= x$
invibeta(a,b,p)	the inverse cumulative beta distribution: if ibeta(a,b,x) $= p$, then invibeta(a,b,p) $= x$

invibetatail(a,b,p)	the inverse reverse cumulative (upper tail or survivor) beta distribution: if ibetatail(a,b,x) $= p$, then invibetatail(a,b,p) $= x$
invigaussian(m,a,p)	the inverse of igaussian(): if igaussian(m,a,x) $= p$, then invigaussian(m,a,p) $= x$
invigaussiantail(m,a,p)	the inverse of igaussiantail(): if igaussiantail(m,a,x) $= p$, then invigaussiantail(m,a,p) $= x$
invlaplace(m,b,p)	the inverse of laplace(): if laplace(m,b,x) $= p$, then invlaplace(m,b,p) $= x$
invlaplacetail(m,b,p)	the inverse of laplacetail(): if laplacetail(m,b,x) $= p$, then invlaplacetail(m,b,p) $= x$
invlogistic(p)	the inverse cumulative logistic distribution: if logistic(x) $= p$, then invlogistic(p) $= x$
invlogistic(s,p)	the inverse cumulative logistic distribution: if logistic(s,x) $= p$, then invlogistic(s,p) $= x$
invlogistic(m,s,p)	the inverse cumulative logistic distribution: if logistic(m,s,x) $= p$, then invlogistic(m,s,p) $= x$
invlogistictail(p)	the inverse reverse cumulative logistic distribution: if logistictail(x) $= p$, then invlogistictail(p) $= x$
invlogistictail(s,p)	the inverse cumulative logistic distribution: if logistic(s,x) $= p$, then invlogistic(s,p) $= x$
invlogistictail(m,s,p)	the inverse cumulative logistic distribution: if logistic(m,s,x) $= p$, then invlogistic(m,s,p) $= x$
invlogit(x)	the inverse of the logit function of x
invnbinomial(n,k,q)	the value of the negative binomial parameter, p, such that $q =$ nbinomial(n,k,p)
invnbinomialtail(n,k,q)	the value of the negative binomial parameter, p, such that $q =$ nbinomialtail(n,k,p)
invnchi2(df,np,p)	the inverse cumulative noncentral χ^2 distribution: if nchi2(df,np,x) $= p$, then invnchi2(df,np,p) $= x$
invnchi2tail(df,np,p)	the inverse reverse cumulative (upper tail or survivor) noncentral χ^2 distribution: if nchi2tail(df,np,x) $= p$, then invnchi2tail(df,np,p) $= x$
invnF(df_1,df_2,np,p)	the inverse cumulative noncentral F distribution: if nF(df_1,df_2,np,f) $= p$, then invnF(df_1,df_2,np,p) $= f$
invnFtail(df_1,df_2,np,p)	the inverse reverse cumulative (upper tail or survivor) noncentral F distribution: if nFtail(df_1,df_2,np,f) $= p$, then invnFtail(df_1,df_2,np,p) $= f$
invnibeta(a,b,np,p)	the inverse cumulative noncentral beta distribution: if nibeta(a,b,np,x) $= p$, then invibeta(a,b,np,p) $= x$
invnormal(p)	the inverse cumulative standard normal distribution: if normal(z) $= p$, then invnormal(p) $= z$
invnt(df,np,p)	the inverse cumulative noncentral Student's t distribution: if nt(df,np,t) $= p$, then invnt(df,np,p) $= t$
invnttail(df,np,p)	the inverse reverse cumulative (upper tail or survivor) noncentral Student's t distribution: if nttail(df,np,t) $= p$, then invnttail(df,np,p) $= t$

invpoisson(k,p)	the Poisson mean such that the cumulative Poisson distribution evaluated at k is p: if poisson(m,k) $= p$, then invpoisson(k,p) $= m$
invpoissontail(k,q)	the Poisson mean such that the reverse cumulative Poisson distribution evaluated at k is q: if poissontail(m,k) $= q$, then invpoissontail(k,q) $= m$
invsym(M)	the inverse of M if M is positive definite
invt(df,p)	the inverse cumulative Student's t distribution: if t(df,t) $= p$, then invt(df,p) $= t$
invttail(df,p)	the inverse reverse cumulative (upper tail or survivor) Student's t distribution: if ttail(df,t) $= p$, then invttail(df,p) $= t$
invtukeyprob(k,df,p)	the inverse cumulative Tukey's Studentized range distribution with k ranges and df degrees of freedom
invvech(M)	a symmetric matrix formed by filling in the columns of the lower triangle from a row or column vector
invvecp(M)	a symmetric matrix formed by filling in the columns of the upper triangle from a row or column vector
invweibull(a,b,p)	the inverse cumulative Weibull distribution with shape a and scale b: if weibull(a,b,x) $= p$, then invweibull(a,b,p) $= x$
invweibull(a,b,g,p)	the inverse cumulative Weibull distribution with shape a, scale b, and location g: if weibull(a,b,g,x) $= p$, then invweibull(a,b,g,p) $= x$
invweibullph(a,b,p)	the inverse cumulative Weibull (proportional hazards) distribution with shape a and scale b: if weibullph(a,b,x) $= p$, then invweibullph(a,b,p) $= x$
invweibullph(a,b,g,p)	the inverse cumulative Weibull (proportional hazards) distribution with shape a, scale b, and location g: if weibullph(a,b,g,x) $= p$, then invweibullph(a,b,g,p) $= x$
invweibullphtail(a,b,p)	the inverse reverse cumulative Weibull (proportional hazards) distribution with shape a and scale b: if weibullphtail(a,b,x)$=p$, then invweibullphtail(a,b,p)$=x$
invweibullphtail(a,b,g,p)	the inverse reverse cumulative Weibull (proportional hazards) distribution with shape a, scale b, and location g: if weibullphtail(a,b,g,x) $= p$, then invweibullphtail(a,b,g,p) $= x$
invweibulltail(a,b,p)	the inverse reverse cumulative Weibull distribution with shape a and scale b: if weibulltail(a,b,x) $= p$, then invweibulltail(a,b,p) $= x$
invweibulltail(a,b,g,p)	the inverse reverse cumulative Weibull distribution with shape a, scale b, and location g: if weibulltail(a,b,g,x) $= p$, then invweibulltail(a,b,g,p) $= x$
irecode(x,x_1,...,x_n)	*missing* if x is missing or $x_1, \ldots, x_n$ is not weakly increasing; 0 if $x \leq x_1$; 1 if $x_1 < x \leq x_2$; 2 if $x_2 < x \leq x_3$; $\ldots$; n if $x > x_n$
isleapsecond(e_{tC})	1 if e_{tC} is a leap second; otherwise, 0
isleapyear(Y)	1 if Y is a leap year; otherwise, 0
issymmetric(M)	1 if the matrix is symmetric; otherwise, 0

J(r,c,z)	the $r \times c$ matrix containing elements z
laplace(m,b,x)	the cumulative Laplace distribution with mean m and scale parameter b
laplaceden(m,b,x)	the probability density of the Laplace distribution with mean m and scale parameter b
laplacetail(m,b,x)	the reverse cumulative (upper tail or survivor) Laplace distribution with mean m and scale parameter b
lastdayofmonth(e_d)	the e_d date of the last day of the month of e_d
lastdowofmonth(M,Y,d)	a synonym for lastweekdayofmonth(M,Y,d)
lastweekdayofmonth(M,Y,d)	the e_d date of the last day-of-week d in month M of year Y
ln(x)	the natural logarithm, $\ln(x)$
ln1m(x)	the natural logarithm of $1-x$ with higher precision than ln($1-x$) for small values of $\lvert x\rvert$
ln1p(x)	the natural logarithm of $1+x$ with higher precision than ln($1+x$) for small values of $\lvert x\rvert$
lncauchyden(a,b,x)	the natural logarithm of the density of the Cauchy distribution with location parameter a and scale parameter b
lnfactorial(n)	the natural log of n factorial = $\ln(n!)$
lngamma(x)	$\ln\{\Gamma(x)\}$
lnigammaden(a,b,x)	the natural logarithm of the inverse gamma density, where a is the shape parameter and b is the scale parameter
lnigaussianden(m,a,x)	the natural logarithm of the inverse Gaussian density with mean m and shape parameter a
lniwishartden(df,V,X)	the natural logarithm of the density of the inverse Wishart distribution; missing if $df \leq n-1$
lnlaplaceden(m,b,x)	the natural logarithm of the density of the Laplace distribution with mean m and scale parameter b
lnmvnormalden(M,V,X)	the natural logarithm of the multivariate normal density
lnnormal(z)	the natural logarithm of the cumulative standard normal distribution
lnnormalden(z)	the natural logarithm of the standard normal density, $N(0,1)$
lnnormalden(x,σ)	the natural logarithm of the normal density with mean 0 and standard deviation σ
lnnormalden(x,μ,σ)	the natural logarithm of the normal density with mean μ and standard deviation σ, $N(\mu,\sigma^2)$
lnwishartden(df,V,X)	the natural logarithm of the density of the Wishart distribution; missing if $df \leq n-1$
log(x)	a synonym for ln(x)
log10(x)	the base-10 logarithm of x
log1m(x)	a synonym for ln1m(x)
log1p(x)	a synonym for ln1p(x)
logistic(x)	the cumulative logistic distribution with mean 0 and standard deviation $\pi/\sqrt{3}$
logistic(s,x)	the cumulative logistic distribution with mean 0, scale s, and standard deviation $s\pi/\sqrt{3}$

logistic(m,s,x)	the cumulative logistic distribution with mean m, scale s, and standard deviation $s\pi/\sqrt{3}$
logisticden(x)	the density of the logistic distribution with mean 0 and standard deviation $\pi/\sqrt{3}$
logisticden(s,x)	the density of the logistic distribution with mean 0, scale s, and standard deviation $s\pi/\sqrt{3}$
logisticden(m,s,x)	the density of the logistic distribution with mean m, scale s, and standard deviation $s\pi/\sqrt{3}$
logistictail(x)	the reverse cumulative logistic distribution with mean 0 and standard deviation $\pi/\sqrt{3}$
logistictail(s,x)	the reverse cumulative logistic distribution with mean 0, scale s, and standard deviation $s\pi/\sqrt{3}$
logistictail(m,s,x)	the reverse cumulative logistic distribution with mean m, scale s, and standard deviation $s\pi/\sqrt{3}$
logit(x)	the log of the odds ratio of x, logit(x) $= \ln\{x/(1-x)\}$
matmissing(M)	1 if any elements of the matrix are missing; otherwise, 0
matrix(*exp*)	restricts name interpretation to scalars and matrices; see scalar()
matuniform(r,c)	the $r \times c$ matrices containing uniformly distributed pseudorandom numbers on the interval $(0,1)$
max(x_1,x_2,...,x_n)	the maximum value of $x_1, x_2, \ldots, x_n$
maxbyte()	the largest value that can be stored in storage type byte
maxdouble()	the largest value that can be stored in storage type double
maxfloat()	the largest value that can be stored in storage type float
maxint()	the largest value that can be stored in storage type int
maxlong()	the largest value that can be stored in storage type long
mdy(M,D,Y)	the e_d date (days since 01jan1960) corresponding to M, D, Y
mdyhms(M,D,Y,h,m,s)	the e_{tc} datetime (ms. since 01jan1960 00:00:00.000) corresponding to M, D, Y, h, m, s
mi(x_1,x_2,...,x_n)	a synonym for missing(x_1,x_2,...,x_n)
min(x_1,x_2,...,x_n)	the minimum value of $x_1, x_2, \ldots, x_n$
minbyte()	the smallest value that can be stored in storage type byte
mindouble()	the smallest value that can be stored in storage type double
minfloat()	the smallest value that can be stored in storage type float
minint()	the smallest value that can be stored in storage type int
minlong()	the smallest value that can be stored in storage type long
minutes(ms)	$ms/60{,}000$
missing(x_1,x_2,...,x_n)	1 if any x_i evaluates to *missing*; otherwise, 0
mm(e_{tc})	the minute corresponding to datetime e_{tc} (ms. since 01jan1960 00:00:00.000)
mmC(e_{tC})	the minute corresponding to datetime e_{tC} (ms. with leap seconds since 01jan1960 00:00:00.000)
mod(x,y)	the modulus of x with respect to y
mofd(e_d)	the e_m monthly date (months since 1960m1) containing date e_d

month(e_d)	the numeric month corresponding to date e_d
monthly(s_1,s_2[,Y])	the e_m monthly date (months since 1960m1) corresponding to s_1 based on s_2 and Y; Y specifies *topyear*; see date()
mreldif(X,Y)	the relative difference of X and Y, where the relative difference is defined as $\max_{i,j}\{\lvert x_{ij} - y_{ij}\rvert / (\lvert y_{ij}\rvert + 1)\}$
msofhours(h)	$h \times$ 3,600,000
msofminutes(m)	$m \times$ 60,000
msofseconds(s)	$s \times$ 1,000
nbetaden(a,b,np,x)	the probability density function of the noncentral beta distribution; 0 if $x < 0$ or $x > 1$
nbinomial(n,k,p)	the cumulative probability of the negative binomial distribution
nbinomialp(n,k,p)	the negative binomial probability
nbinomialtail(n,k,p)	the reverse cumulative probability of the negative binomial distribution
nchi2(df,np,x)	the cumulative noncentral χ^2 distribution; 0 if $x < 0$
nchi2den(df,np,x)	the probability density of the noncentral χ^2 distribution; 0 if $x < 0$
nchi2tail(df,np,x)	the reverse cumulative (upper tail or survivor) noncentral χ^2 distribution; 1 if $x < 0$
nextbirthday($e_{d\text{DOB}}$,e_d[,s_{nl}])	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
nextdow(e_d,d)	a synonym for nextweekday(e_d,d)
nextleapyear(Y)	the first leap year after year Y
nextweekday(e_d,d)	the e_d date of the first day-of-week d after e_d
nF(df_1,df_2,np,f)	the cumulative noncentral F distribution with df_1 numerator and df_2 denominator degrees of freedom and noncentrality parameter np; 0 if $f < 0$
nFden(df_1,df_2,np,f)	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 $f < 0$
nFtail(df_1,df_2,np,f)	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 $f < 0$
nibeta(a,b,np,x)	the cumulative noncentral beta distribution; 0 if $x < 0$; or 1 if $x > 1$
normal(z)	the cumulative standard normal distribution
normalden(z)	the standard normal density, $N(0, 1)$
normalden(x,σ)	the normal density with mean 0 and standard deviation σ
normalden(x,μ,σ)	the normal density with mean μ and standard deviation σ, $N(\mu, \sigma^2)$
now()	the current e_{tc} datetime
npnchi2(df,x,p)	the noncentrality parameter, np, for noncentral χ^2: if nchi2(df,np,x) $= p$, then npnchi2(df,x,p) $= np$
npnF(df_1,df_2,f,p)	the noncentrality parameter, np, for the noncentral F: if nF(df_1,df_2,np,f) $= p$, then npnF(df_1,df_2,f,p) $= np$
npnt(df,t,p)	the noncentrality parameter, np, for the noncentral Student's t distribution: if nt(df,np,t) $= p$, then npnt(df,t,p) $= np$

`nt(`df`,`np`,`t`)`	the cumulative noncentral Student's t distribution with df degrees of freedom and noncentrality parameter np
`ntden(`df`,`np`,`t`)`	the probability density function of the noncentral Student's t distribution with df degrees of freedom and noncentrality parameter np
`nttail(`df`,`np`,`t`)`	the reverse cumulative (upper tail or survivor) noncentral Student's t distribution with df degrees of freedom and noncentrality parameter np
`nullmat(`*matname*`)`	use with the row-join (`,`) and column-join (`\\`) operators
`plural(`n`,`s`)`	the plural of s if $n \neq \pm 1$
`plural(`n`,`s_1`,`s_2`)`	the plural of s_1, as modified by or replaced with s_2, if $n \neq \pm 1$
`poisson(`m`,`k`)`	the probability of observing `floor(`k`)` or fewer outcomes that are distributed as Poisson with mean m
`poissonp(`m`,`k`)`	the probability of observing `floor(`k`)` outcomes that are distributed as Poisson with mean m
`poissontail(`m`,`k`)`	the probability of observing `floor(`k`)` or more outcomes that are distributed as Poisson with mean m
`previousbirthday(`$e_{d\,\mathrm{DOB}}$`,`e_d`[,`s_{nl}`])`	the e_d date of the birthday immediately before e_d for date of birth $e_{d\,\mathrm{DOB}}$ with s_{nl} the nonleap-year birthday for 29feb birthdates
`previousdow(`e_d`,`d`)`	a synonym for `previousweekday(`e_d`,`d`)`
`previousleapyear(`Y`)`	the leap year immediately before year Y
`previousweekday(`e_d`,`d`)`	the e_d date of the last day-of-week d before e_d
`qofd(`e_d`)`	the e_q quarterly date (quarters since 1960q1) containing date e_d
`quarter(`e_d`)`	the numeric quarter of the year corresponding to date e_d
`quarterly(`s_1`,`s_2`[,`Y`])`	the e_q quarterly date (quarters since 1960q1) corresponding to s_1 based on s_2 and Y; Y specifies *topyear*; see `date()`
`r(`*name*`)`	the value of the stored result `r(`*name*`)`; see **[U] 18.8 Accessing results calculated by other programs**
`rbeta(`a`,`b`)`	beta(a,b) random variates, where a and b are the beta distribution shape parameters
`rbinomial(`n`,`p`)`	binomial(n,p) random variates, where n is the number of trials and p is the success probability
`rcauchy(`a`,`b`)`	Cauchy(a,b) random variates, where a is the location parameter and b is the scale parameter
`rchi2(`df`)`	χ^2, with df degrees of freedom, random variates
`recode(`x`,`x_1`,...,`x_n`)`	*missing* if $x_1, x_2, \ldots, x_n$ is not weakly increasing; x if x is missing; x_1 if $x \leq x_1$; x_2 if $x \leq x_2, \ldots$; otherwise, x_n if $x > x_1, x_2, \ldots, x_{n-1}$. $x_i \geq .$ is interpreted as $x_i = +\infty$
`real(`s`)`	s converted to numeric or *missing*
`regexcapture(`n`)`	subexpression n from a previous `regexm()` or `regexmatch()` match
`regexcapturenamed(`*grp*`)`	subexpression corresponding to matching group named *grp* in regular expression from a previous `regexm()` or `regexmatch()` match
`regexm(`s`,`*re*`)`	a match of a regular expression, which evaluates to `1` if regular expression *re* is satisfied by the ASCII string s; otherwise, `0`

`regexmatch(`s`,`re`[,`noc`[,`std`[,`$nlalt$`]]])`
: a match of a regular expression, which evaluates to `1` if regular expression re is satisfied by the ASCII string s; otherwise, `0`

`regexr(`s_1`,`re`,`s_2`)` replaces the first substring within ASCII string s_1 that matches re with ASCII string s_2 and returns the resulting string

`regexreplace(`s_1`,`re`,`s_2`[,`noc`[,`fmt`[,`std`[,`$nlalt$`]]]])`
: replaces the first substring within ASCII string s_1 that matches re with ASCII string s_2 and returns the resulting string

`regexreplaceall(`s_1`,`re`,`s_2`[,`noc`[,`fmt`[,`std`[,`$nlalt$`]]]])`
: replaces all substrings within ASCII string s_1 that match re with ASCII string s_2 and returns the resulting string

`regexs(`n`)` subexpression n from a previous `regexm()` or `regexmatch()` match, where $0 \leq n < 10$

`reldif(`x`,`y`)` the "relative" difference $|x - y|/(|y| + 1)$; `0` if both arguments are the same type of extended missing value; *missing* if only one argument is missing or if the two arguments are two different types of *missing*

`replay()` `1` if the first nonblank character of local macro `` `0' `` is a comma, or if `` `0' `` is empty

`return(`*name*`)` the value of the to-be-stored result `r(`*name*`)`; see [P] **return**

`rexponential(`b`)` exponential random variates with scale b

`rgamma(`a`,`b`)` gamma(a,b) random variates, where a is the gamma shape parameter and b is the scale parameter

`rhypergeometric(`N`,`K`,`n`)` hypergeometric random variates

`rigaussian(`m`,`a`)` inverse Gaussian random variates with mean m and shape parameter a

`rlaplace(`m`,`b`)` Laplace(m,b) random variates with mean m and scale parameter b

`rlogistic()` logistic variates with mean 0 and standard deviation $\pi/\sqrt{3}$

`rlogistic(`s`)` logistic variates with mean 0, scale s, and standard deviation $s\pi/\sqrt{3}$

`rlogistic(`m`,`s`)` logistic variates with mean m, scale s, and standard deviation $s\pi/\sqrt{3}$

`rnbinomial(`n`,`p`)` negative binomial random variates

`rnormal()` standard normal (Gaussian) random variates, that is, variates from a normal distribution with a mean of 0 and a standard deviation of 1

`rnormal(`m`)` normal(m,1) (Gaussian) random variates, where m is the mean and the standard deviation is 1

`rnormal(`m`,`s`)` normal(m,s) (Gaussian) random variates, where m is the mean and s is the standard deviation

`round(`x`,`y`)` or `round(`x`)` 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 `round(.a)` = `.a` and that `round(.a,`y`)` = `.a` if y is not missing) and if y is missing, then "`.`" is returned

`roweqnumb(`M`,`s`)` the equation number of M associated with row equation s; *missing* if the row equation cannot be found

`rownfreeparms(`M`)` the number of free parameters in rows of M

`rownumb(`M`,`s`)` the row number of M associated with row name s; *missing* if the row cannot be found

`rowsof(`M`)`	the number of rows of M
`rpoisson(`m`)`	Poisson(m) random variates, where m is the distribution mean
`rt(`df`)`	Student's t random variates, where df is the degrees of freedom
`runiform()`	uniformly distributed random variates over the interval $(0, 1)$
`runiform(`a`,`b`)`	uniformly distributed random variates over the interval (a, b)
`runiformint(`a`,`b`)`	uniformly distributed random integer variates on the interval $[a, b]$
`rweibull(`a`,`b`)`	Weibull variates with shape a and scale b
`rweibull(`a`,`b`,`g`)`	Weibull variates with shape a, scale b, and location g
`rweibullph(`a`,`b`)`	Weibull (proportional hazards) variates with shape a and scale b
`rweibullph(`a`,`b`,`g`)`	Weibull (proportional hazards) variates with shape a, scale b, and location g
`s(`*name*`)`	the value of stored result `s(`*name*`)`; see **[U] 18.8 Accessing results calculated by other programs**
`scalar(`*exp*`)`	restricts name interpretation to scalars and matrices
`seconds(`ms`)`	ms/1,000
`sign(`x`)`	the sign of x: -1 if $x < 0$, 0 if $x = 0$, 1 if $x > 0$, or *missing* if x is missing
`sin(`x`)`	the sine of x, where x is in radians
`sinh(`x`)`	the hyperbolic sine of x
`smallestdouble()`	the smallest double-precision number greater than zero
`soundex(`s`)`	the soundex code for a string, s
`soundex_nara(`s`)`	the US Census soundex code for a string, s
`sqrt(`x`)`	the square root of x
`ss(`e_{tc}`)`	the second corresponding to datetime e_{tc} (ms. since 01jan1960 00:00:00.000)
`ssC(`e_{tC}`)`	the second corresponding to datetime e_{tC} (ms. with leap seconds since 01jan1960 00:00:00.000)
`strcat(`s_1`,`s_2`)`	there is no `strcat()` function; instead the addition operator is used to concatenate strings
`strdup(`s_1`,`n`)`	there is no `strdup()` function; instead the multiplication operator is used to create multiple copies of strings
`string(`n`)`	a synonym for `strofreal(`n`)`
`string(`n`,`s`)`	a synonym for `strofreal(`n`,`s`)`
`stritrim(`s`)`	s with multiple, consecutive internal blanks (ASCII space character `char(32)`) collapsed to one blank
`strlen(`s`)`	the number of characters in ASCII s or length in bytes
`strlower(`s`)`	lowercase ASCII characters in string s
`strltrim(`s`)`	s without leading blanks (ASCII space character `char(32)`)
`strmatch(`s_1`,`s_2`)`	`1` if s_1 matches the pattern s_2; otherwise, `0`
`strofreal(`n`)`	n converted to a string
`strofreal(`n`,`s`)`	n converted to a string using the specified display format
`strpos(`s_1`,`s_2`)`	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

strproper(s)	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
strreverse(s)	the reverse of ASCII string s
strrpos(s_1,s_2)	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
strrtrim(s)	s without trailing blanks (ASCII space character char(32))
strtoname(s[,p])	s translated into a Stata 13 compatible name
strtrim(s)	s without leading and trailing blanks (ASCII space character char(32)); equivalent to strltrim(strrtrim(s))
strupper(s)	uppercase ASCII characters in string s
subinstr(s_1,s_2,s_3,n)	s_1, where the first n occurrences in s_1 of s_2 have been replaced with s_3
subinword(s_1,s_2,s_3,n)	s_1, where the first n occurrences in s_1 of s_2 as a word have been replaced with s_3
substr(s,n_1,n_2)	the substring of s, starting at n_1, for a length of n_2
sum(x)	the running sum of x, treating missing values as zero
sweep(M,i)	matrix M with ith row/column swept
t(df,t)	the cumulative Student's t distribution with df degrees of freedom
tan(x)	the tangent of x, where x is in radians
tanh(x)	the hyperbolic tangent of x
tC(l)	convenience function to make typing dates and times in expressions easier
tc(l)	convenience function to make typing dates and times in expressions easier
td(l)	convenience function to make typing dates in expressions easier
tden(df,t)	the probability density function of Student's t distribution
th(l)	convenience function to make typing half-yearly dates in expressions easier
tin(d_1,d_2)	*true* if $d_1 \leq t \leq d_2$, where t is the time variable previously tsset
tm(l)	convenience function to make typing monthly dates in expressions easier
tobytes(s[,n])	escaped decimal or hex digit strings of up to 200 bytes of s
today()	today's e_d date
tq(l)	convenience function to make typing quarterly dates in expressions easier
trace(M)	the trace of matrix M
trigamma(x)	the second derivative of lngamma(x) $= d^2 \ln\Gamma(x)/dx^2$
trunc(x)	a synonym for int(x)
ttail(df,t)	the reverse cumulative (upper tail or survivor) Student's t distribution; the probability $T > t$
tukeyprob(k,df,x)	the cumulative Tukey's Studentized range distribution with k ranges and df degrees of freedom; 0 if $x < 0$
tw(l)	convenience function to make typing weekly dates in expressions easier

twithin(d_1,d_2)	*true* if $d_1 < t < d_2$, where t is the time variable previously tsset
uchar(n)	the Unicode character corresponding to Unicode code point n or an empty string if n is beyond the Unicode code-point range
udstrlen(s)	the number of display columns needed to display the Unicode string s in the Stata Results window
udsubstr(s,n_1,n_2)	the Unicode substring of s, starting at character n_1, for n_2 display columns
uisdigit(s)	1 if the first Unicode character in s is a Unicode decimal digit; otherwise, 0
uisletter(s)	1 if the first Unicode character in s is a Unicode letter; otherwise, 0
ustrcompare(s_1,s_2[,*loc*])	compares two Unicode strings
ustrcompareex(s_1,s_2,*loc*,*st*,*case*,*cslv*,*norm*,*num*,*alt*,*fr*)	compares two Unicode strings
ustrfix(s[,*rep*])	replaces each invalid UTF-8 sequence with a Unicode character
ustrfrom(s,*enc*,*mode*)	converts the string s in encoding *enc* to a UTF-8 encoded Unicode string
ustrinvalidcnt(s)	the number of invalid UTF-8 sequences in s
ustrleft(s,n)	the first n Unicode characters of the Unicode string s
ustrlen(s)	the number of characters in the Unicode string s
ustrlower(s[,*loc*])	lowercase all characters of Unicode string s under the given locale *loc*
ustrltrim(s)	removes the leading Unicode whitespace characters and blanks from the Unicode string s
ustrnormalize(s,*norm*)	normalizes Unicode string s to one of the five normalization forms specified by *norm*
ustrpos(s_1,s_2[,n])	the position in s_1 at which s_2 is first found; otherwise, 0
ustrregexm(s,*re*[,*noc*])	performs a match of a regular expression and evaluates to 1 if regular expression *re* is satisfied by the Unicode string s; otherwise, 0
ustrregexra(s_1,*re*,s_2[,*noc*])	replaces all substrings within the Unicode string s_1 that match *re* with s_2 and returns the resulting string
ustrregexrf(s_1,*re*,s_2[,*noc*])	replaces the first substring within the Unicode string s_1 that matches *re* with s_2 and returns the resulting string
ustrregexs(n)	subexpression n from a previous ustrregexm() match
ustrreverse(s)	the reverse of Unicode string s
ustrright(s,n)	the last n Unicode characters of the Unicode string s
ustrrpos(s_1,s_2[,n])	the position in s_1 at which s_2 is last found; otherwise, 0
ustrrtrim(s)	remove trailing Unicode whitespace characters and blanks from the Unicode string s
ustrsortkey(s[,*loc*])	generates a null-terminated byte array that can be used by the sort command to produce the same order as ustrcompare()
ustrsortkeyex(s,*loc*,*st*,*case*,*cslv*,*norm*,*num*,*alt*,*fr*)	generates a null-terminated byte array that can be used by the sort command to produce the same order as ustrcompare()
ustrtitle(s[,*loc*])	a string with the first characters of Unicode words titlecased and other characters lowercased

ustrto(s,enc,$mode$)	converts the Unicode string s in UTF-8 encoding to a string in encoding enc
ustrtohex(s[,n])	escaped hex digit string of s up to 200 Unicode characters
ustrtoname(s[,p])	string s translated into a Stata name
ustrtrim(s)	removes leading and trailing Unicode whitespace characters and blanks from the Unicode string s
ustrunescape(s)	the Unicode string corresponding to the escaped sequences of s
ustrupper(s[,loc])	uppercase all characters in string s under the given locale loc
ustrword(s,n[,loc])	the nth Unicode word in the Unicode string s
ustrwordcount(s[,loc])	the number of nonempty Unicode words in the Unicode string s
usubinstr(s_1,s_2,s_3,n)	replaces the first n occurrences of the Unicode string s_2 with the Unicode string s_3 in s_1
usubstr(s,n_1,n_2)	the Unicode substring of s, starting at n_1, for a length of n_2
vec(M)	a column vector formed by listing the elements of M, starting with the first column and proceeding column by column
vecdiag(M)	the row vector containing the diagonal of matrix M
vech(M)	a column vector formed by listing the lower triangle elements of M
vecp(M)	a column vector formed by listing the upper triangle elements of M
week(e_d)	the numeric week of the year corresponding to date e_d, the %td encoded date (days since 01jan1960)
weekly(s_1,s_2[,Y])	the e_w weekly date (weeks since 1960w1) corresponding to s_1 based on s_2 and Y; Y specifies *topyear*; see date()
weibull(a,b,x)	the cumulative Weibull distribution with shape a and scale b
weibull(a,b,g,x)	the cumulative Weibull distribution with shape a, scale b, and location g
weibullden(a,b,x)	the probability density function of the Weibull distribution with shape a and scale b
weibullden(a,b,g,x)	the probability density function of the Weibull distribution with shape a, scale b, and location g
weibullph(a,b,x)	the cumulative Weibull (proportional hazards) distribution with shape a and scale b
weibullph(a,b,g,x)	the cumulative Weibull (proportional hazards) distribution with shape a, scale b, and location g
weibullphden(a,b,x)	the probability density function of the Weibull (proportional hazards) distribution with shape a and scale b
weibullphden(a,b,g,x)	the probability density function of the Weibull (proportional hazards) distribution with shape a, scale b, and location g
weibullphtail(a,b,x)	the reverse cumulative Weibull (proportional hazards) distribution with shape a and scale b
weibullphtail(a,b,g,x)	the reverse cumulative Weibull (proportional hazards) distribution with shape a, scale b, and location g
weibulltail(a,b,x)	the reverse cumulative Weibull distribution with shape a and scale b
weibulltail(a,b,g,x)	the reverse cumulative Weibull distribution with shape a, scale b, and location g
wofd(e_d)	the e_w weekly date (weeks since 1960w1) containing date e_d

`word(`s`,`n`)`	the nth word in s; *missing* (`""`) if n is missing
`wordbreaklocale(`*loc*`,`*type*`)`	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*
`wordcount(`s`)`	the number of words in s
`year(`e_d`)`	the numeric year corresponding to date e_d
`yearly(`s_1`,`s_2[`,`Y]`)`	the e_y yearly date (year) corresponding to s_1 based on s_2 and Y; Y specifies *topyear*; see `date()`
`yh(`Y`,`H`)`	the e_h half-yearly date (half-years since 1960h1) corresponding to year Y, half-year H
`ym(`Y`,`M`)`	the e_m monthly date (months since 1960m1) corresponding to year Y, month M
`yofd(`e_d`)`	the e_y yearly date (year) containing date e_d
`yq(`Y`,`Q`)`	the e_q quarterly date (quarters since 1960q1) corresponding to year Y, quarter Q
`yw(`Y`,`W`)`	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